Effiziente Kapazitätsplanung durch dynamische Erweiterung einer lokalen Ressourcenumgebung um Grid- und Cloud-Ressourcen: algorithmische und technische Betrachtungen.

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von
Alexander Fölling

Dortmund
Mai 2014

Tag der mündlichen Prüfung:
05. 05. 2014

Dekan:
Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr.-Ing. Uwe Schwiegelshohn, TU Dortmund
Prof. Dr. Dieter Kranzlmüller, LMU München

Danksagung

Mit dem Ende eines beruflichen Lebensabschnitts blickt man auch zurück auf Dinge, die man erlebt und Leistungen, die man erbracht hat. Dabei erinnert man sich sowohl an schwierige Abschnitte als auch an mitreißende und spannende Passagen auf der Reise zum Ziel. Darüber hinaus wird man sich der Privilegiertheit der eigenen Situation bewusst.

Ich bin privilegiert durch mein soziales Umfeld aus Eltern, Geschwistern, Verwandten und Freunden, die mich im Leben begleiten, stets unterstützen und die Grundlage für mein berufliches und privates Glück gelegt haben. Ich bin privilegiert mit meiner eigenen kleinen Familie — meiner lieben Frau Melanie und meinem Sohn Felix, die mir das größte Glück sind.

Als hauptverantwortlich für meinen wissenschaftlichen Werdegang hingegen sehe ich meine ehemaligen Kollegen Joachim Lepping, Alexander Papaspyrou und Christian Grimme, die mich nicht nur für die wissenschaftliche Arbeit begeistert haben, sondern mir über viele Jahre als Freunde, Berater und Kritiker den Mut zugesprochen haben, meinen Weg zu gehen.

Zudem möchte ich ganz besonders Prof. Dr.-Ing. Uwe Schwiegelshohn danken, der es mir ermöglicht hat, mich in einem so vielseitigen Beruf aus Lehre, Projektarbeit und freier Forschung zu entfalten, denn gerade Grundlagenforschung braucht Luft zum Atmen und die Freiheit, Irrwege zu gehen. Jungen Forschern diese Möglichkeit einzuräumen, rechne ich ihm hoch an.
Darüber hinaus bedanke ich mich bei Prof. Dr. Dieter Kranzlmüller für sein Interesse an meiner Arbeit und seine Bereitschaft zur Zweibegutachtung.

Zum Abschluss geht mein Dank auch an die vielen Kollegen des IRF, die meinen Alltag nicht nur produktiv gemacht, sondern auch mit viel Humor und frischen Ideen bereichert haben: Daniel Hauschildt, Nicolaj Kirchhof, Oliver Urbann, Matthias Hofmann, Tanja Burda, Sabine Winterhoff, Lajos Herpay, Tassilo Galonska, Markus Kemmerling, Stefan Tasse, Sonja Neweling, Stefan Freitag, Sören Kerner, Lars Schley, Jürgen Kemper, Jörg Platte und für die vielen technischen kleinen Hilfen sowie bedingungslose Hilfsbereitschaft René Schubert und Peter Resch.

Abstract

The increasing complexity of scientific problems and industrial challenges leads to significant demands on computing infrastructures. Today, many scientific institutions and also most companies are still hosting their own computing infrastructures as commodity clusters or parallel machines to handle the upcoming load. Usually, these infrastructures are shared between many users with various applications.

Caused by new problems and a fluctuating user community, changing load leads to phases with overloaded resources and undesired delays in job calculation on the first hand, and to idle phases with inefficent use of the available performance on the other hand.

Instead of adapting the physical resources to the varying load, a resource operator is able to make use of algorithms and technologies of Grid- and Cloud-Computing to dynamically exchange load with delegation partners or a public cloud.

This work investigates algorithmic approaches for the dynamic load exchange between autonomous resource environments in Computational Grids. The bilateral exchange between equitable partners is optimized using computational intelligence methods and subsequently is evaluated concerning improvements in service quality for the respective user communities.

Usually, in Computational Grids the participating resource centers do not charge accepted workload. However, a cloud provider does charge unilateral load exchange. Thus, from the delegating centers point of view the charge represent an additional objective. This leads to a Multi-Objective Optimization Problem with the contradicting objectives of minimizing charge and maximizing service quality. In addition to common offline optimization algorithms, we introduce an adaptive reinforcement learning based policy to establish a beneficial relation of charge and gain for a considered resource centre.

All presented algorithms are evaluated with real workload traces and examined regarding their robustness on foreign workload data or different local scheduling systems. The achieved results motivate the usage of dynamic load exchange algorithms for the improvement in service quality. Further, they serve as alternative to an extension of physical resources, which would be unfavorable from the economical and ecological perspective.

Zusammenfassung

Mit der Komplexität der Probleme aus Forschung und Industrie steigen auch die Anforderungen an Rechnerinfrastrukturen zur ihrer effizienten Berechnung. Dabei greifen viele wissenschaftliche Einrichtungen und Unternehmen bisher noch auf den Betrieb eigener Rechnerinfrastruktur in Form von Commodity-Clustern bzw. Parallelrechnern zurück, deren Zugriff oftmals unter einer Vielzahl von Nutzern mit unterschiedlichen Anwendungen geteilt wird.

Die aufgrund neuer Problemstellungen und der fluktuierenden Nutzergemeinschaft schwankende Rechenlast führt jedoch dazu, dass in Phasen mit Überlast eine unerwünschte Verzögerung der Rechenjobs stattfindet, während das Leistungspotential der Ressourcen in Phasen mit Unterlast nicht voll ausgeschöpft wird.

Anstatt seine Ressourcen physisch den Lastschwankungen anzupassen, kann ein Ressourcenbetreiber auf Algorithmen und Technologien des Grid- und Cloud-Computings zurückgreifen, um einen dynamischen Lastaustausch mit einem Delegationspartner bzw. in eine öffentliche Cloud vorzunehmen.

Diese Arbeit untersucht algorithmische Lösungen für den dynamischen Lastaustausch zwischen autonomen Ressourcenumgebungen in Computational Grids. Der wechselseitige Lastaustausch zwischen gleichberechtigten Partnern wird dabei hinsichtlich einer verbesserten Servicequalität für ihre jeweiligen Nutzergemeinschaften evaluiert und mit Methoden der Computational Intelligence optimiert.

Während für den beidseitigen Lastaustausch innerhalb eines Computational Grids in der Regel kein (monetärer) Ausgleich verlangt wird, ist dies bei einseitiger Auslagerung von Last in eine Cloud stets der Fall. Die durch die Auslagerung entstehenden Kosten stellen neben der Steigerung der Servicequalität einen weiteren Optimierungsfaktor dar, welcher im Rahmen eines mehrkriteriellen Optimierungsproblems aus Kosten und Steigerung der Servicequalität untersucht wird. Neben Offline-Optimierungsverfahren wird für diesen Auslagerungsansatz eine adaptive Online-Lernstrategie auf Basis von Reinforcement Learning umgesetzt, mit dem Ziel ein gutes Kosten-Nutzen-Verhältnis für das auslagernde Ressourcenzentrum zu erreichen.

Alle vorgestellten Algorithmen werden dabei mit realen Lastaufzeichnungen und in Bezug auf ihre Robustheit bei Übertragung auf Änderungen der Eingabedaten sowie des lokalen Scheduling-Systems untersucht. Die gewonnenen Erkenntnisse motivieren den Einsatz dynamischer Lastverteilungsalgorithmen zur Steigerung der Servicequalität als Alternative zu einer aus ökonomischen und ökologischen Gründen unvorteilhaften Erweiterung der physischen Ressourcen.

Eigene Veröffentlichungen

Die vorliegende Arbeit ist auf Basis gemeinsamer Forschungsarbeiten und Veröffentlichungen entstanden, an denen der Autor maßgeblich beteiligt war. Alle im Folgenden genannten Veröffentlichungen werden ebenfalls an entsprechender Stelle im Text angeführt.

Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2009). Decentralized Grid Scheduling with Evolutionary Fuzzy Systems. In Frachtenberg, E. & Schwiegelshohn, U. (Hrsg.), *Proceedings of the 14th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 5798 von *Lecture Notes in Computer Science*, (S. 16–36). Springer.

Fölling, A., Grimme, C., Lepping, J., Papaspyrou, A., & Schwiegelshohn, U. (2009). Competitive Co-evolutionary Learning of Fuzzy Systems for Job Exchange in Computational Grids. *Evolutionary Computation,* 17*(4), 545–560.*

Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2009). *Co-evolving Fuzzy Rule Sets for Job Exchange in Computational Grids. In* Proceedings of the IEEE International Conference on Fuzzy Systems*, (S. 1683–1688). IEEE Computer Society Press.*

Birkenheuer, G., Carlson, A., Fölling, A., Högqvist, M., Hoheisel, A., Papaspyrou, A., Rieger, K., Schott, B., & Ziegler, W. (2009). *Connecting Communities on the Meta-Scheduling Level: The DGSI Approach! In* Proceedings of the Cracow Grid Workshop (CGW).

Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2010). *Robust Load Delegation in Service Grid Environments.* IEEE Transactions on Parallel and Distributed Systems*,* 21*(9), 1304–1316.*

Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2010). *The Gain of Resource Delegation in Distributed Computing Environments. In Schwiegelshohn, U. & Frachtenberg, E. (Hrsg.),* Proceedings of the 15th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 6253 von* Lecture Notes in Computer Science*, (S. 77–92). Springer.*

Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2011).
Connecting Community-Grids by supporting job negotiation with coevolutionary Fuzzy-Systems. Soft Computing, 15*(12), 2375–2387.*

Fölling, A. & Lepping, J. (2012).
Knowledge discovery for scheduling in computational grids. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2, *287–297.*

Fölling, A. & Hofmann, M. (2012).
Improving Scheduling Performance using a Q-Learning-based Leasing Policy for Clouds. In Kaklamanis, C., Papatheodorou, T., & Spirakis, P. (Hrsg.), Euro-Par 2012 Parallel Processing, *Band 7484 von* Lecture Notes in Computer Science, *(S. 337–349)., Berlin, Heidelberg. Springer.*

Inhaltsverzeichnis

1 Einleitung **1**

I Grundlagen **5**

2 Scheduling in HPC-Cluster-Systemen **7**
2.1 Basismodell eines MPP-Systems 10
2.1.1 First Come First Serve 12
2.1.2 EASY 12
2.2 Ein homogenes MPP-System auf heterogenen Ressourcen 14
2.3 Möglichkeiten zur Erweiterung des Ressourcenraums 16

3 Modellannahmen für Job- und Maschinenmodell **19**
3.1 Formale Einführung von Job und Maschinenmodell 21
3.2 Bewertungsgrößen zur Feststellung der Scheduling-Qualität 22
3.2.1 Produktionsspanne und Abarbeitungszeit eines Schedules . 22
3.2.2 Ressourcenprodukt (RP) 23
3.2.3 Auslastung (U) 23
3.2.4 Durchschnittliche gewichtete Antwortzeit (AWRT) 24
3.2.5 Gesamtwartezeit (TWT) 25

4 Simulation von parallelen Systemen **27**
4.1 Auswahl von Eingabedaten 27
4.2 Auswahl eines Frameworks zur Simulation paralleler Systeme . . . 31
4.3 Referenzergebnisse bei der Simulation realer Arbeitslastaufzeichnungen 33

5 Optimierungsverfahren für die Erweiterung lokaler Ressourcen **35**
5.1 Evolutionäre Algorithmen 36
5.2 Evolutionäre Algorithmen für die mehrkriterielle Optimierung . . . 40
5.3 Reinforcement Learning 44

II Ressourcenerweiterung mit Computational Grids 49

6 Grundlagen von Computational Grids 51
6.1 Betrachtete Grid-Scheduling-Architektur 52
6.2 Technische Umsetzung eines dezentralen Computational Grids am Beispiel von DGSI 55

7 Aktivitätendelegation in dezentralen Computational Grids 57
7.1 Existierende algorithmische Ansätze für die Aktivitätendelegation . 57
7.2 Eigene Ansätze für die Aktivitätendelegation 62
7.2.1 Erweiterung von Modell und Bewertungsgrößen 62
7.2.2 Aufbau von Strategien zur Aktivitätendelegation 64
7.2.3 Eine einfache Heuristik zur Realisierung einer Transferstrategie 67
7.2.4 Lernbasierte Realisierung einer Transferstrategie 70
7.2.5 Effiziente Implementierung einer Lokationsstrategie 86
7.2.6 Coevolutionäres Lernen von Regelbasen 92
7.3 Abschluss der Aktivitätendelegation 97

8 Ressourcendelegation in dezentralen Computational Grids 99
8.1 Algorithmen für die Ressourcendelegation 100
8.1.1 Modell und Bewertungsgrößen 101
8.1.2 Eine konfigurierbare Strategie zur Ressourcendelegation . . 102
8.1.3 Evaluation der RD-Strategie 108
8.1.4 Auswahlstrategie bei mehreren Partnern 115
8.2 Abschluss der Ressourcendelegation 121

III Hybrides Cloud Scheduling 123

9 Einführung in das Cloud Computing 125

10 Grundlagen des hybriden Cloud Schedulings 129
10.1 Technische Voraussetzungen 129
10.2 Wissenschaftliches Rechnen in der Cloud 131
10.3 Existierende Ansätze des Cloud Schedulings 132

11 Eigene Ansätze für das hybride Cloud Scheduling 135
11.1 Modell und Bewertungsgrößen für das hybride Cloud Scheduling . 135
11.1.1 Kostenmodell 137
11.1.2 Die Bilanz als Kosten-Nutzen-Funktion 138
11.1.3 Diskretisierung der Cloud Scheduling Entscheidungen . . . 140
11.1.4 Diskussion und Einführung verschiedener Strategieparameter 142
11.2 Feststellung des Optimierungspotentials mit mehrkriteriellen Optimierungsverfahren 148
11.2.1 Kodierung des Optimierungsproblems für NSGA-II 148

11.2.2 Evaluation 150
11.3 Dynamisches Online-Lernen von Aktionssequenzen 154
11.3.1 Grundlegende Modellierung des Problems 155
11.3.2 Konfiguration der Lernparameter 158
11.3.3 Evaluation der Leistung des Q-Learning 162
11.3.4 Vereinfachung des Lernverfahrens 164
11.3.5 Untersuchung der Robustheit beider Online-Verfahren im Fehlerfall 168
11.3.6 Übertragbarkeitsanalyse der unterschiedlichen Konzepte . . 173

12 Abschluss von Teil III **181**

Zusammenfassung und Ausblick **183**

Anhang: Inhalt der Daten-CD **189**

Anhang: Architektur einer Cluster-Simulation **197**

Literaturverzeichnis **199**

Kapitel 1

Einleitung

Ein Großteil des heutigen technologischen und wissenschaftlichen Fortschritts wäre ohne die gleichzeitige Fortentwicklung moderner Rechnerinfrastrukturen und Algorithmen nicht möglich gewesen.

Davon sind in erster Linie naturwissenschaftliche Problemstellungen betroffen. So müssen bei einigen Problemen große Datenmengen verarbeitet werden, um beispielsweise Muster in DNA-Sequenzen zu entschlüsseln oder Teleskopaufnahmen und Klimadaten auszuwerten. Gleichzeitig werden Entwicklungen in vielen technischen Einsatzgebieten von aufwendigen Simulationen begleitet, die physikalische Prozesse nachbilden, um beispielsweise geräuscharme Flugzeuge zu konzipieren, die möglichst viele Passagiere bei niedrigem Treibstoffverbrauch transportieren können, oder stabile Brücken mit minimalem Materialaufwand zu bauen.

Eines ist allen vorangehenden Beispielen gemein: Neben dem eigentlichen Speicherbedarf für die Speicherung von Eingabe- und Ergebnisdaten, benötigen sowohl Auswertungen als auch Simulationen in der Regel große Rechenleistung, die sich nicht mehr auf Basis eines einzelnen Desktop-Systems bereitstellen lässt. Vielmehr können sie erst auf Basis paralleler und verteilter Infrastrukturen wie Rechencluster effizient gelöst werden.

Gewöhnlicherweise lohnt sich die Anschaffung und der Betrieb eines solchen Rechenclusters nur, wenn dieses eine größere Nutzergemeinschaft besitzt und somit gleichzeitig zur Lösung vieler verschiedener Problemstellungen dient. Die Zuteilung der geteilten Ressourcen zu den einzelnen Rechenaufgaben der Nutzer findet dabei über Scheduling-Systeme statt.

Durch eine wechselnde Nutzergemeinschaft, neue interessante wissenschaftliche Fragestellungen oder der schwankenden Auftragslage einer Firma (bei Nutzung von Rechenkapazitäten für kommerzielle Zwecke) kommt es zwangsläufig auch zu schwankender Rechenlast im System. Dabei führt eine hohe Last zu Verzögerungen in der Abarbeitung der Rechenaufgaben und gefährdet damit unter Umständen die Einhaltung von Terminen (wissenschaftliche Veröffentlichung, Fertigstellung eines Auftrags). Eine niedrige Last hingegen geht mit einer geringeren Wirtschaftlichkeit

der Ressourcen einher, da diese in der Regel zu erheblichen Kosten für Betrieb und Wartung führen, denen in dem entsprechenden Zeitraum weder wissenschaftlicher noch kommerzieller Gewinn gegenüberstehen.

In der Praxis setzt der Betrieb von Rechenressourcen daher eine ständige Anpassung der Kapazitäten voraus, die dem Wechselspiel aus Wirtschaftlichkeit und Anforderungen seitens der Nutzergemeinschaft unterworfen ist.

Eine Möglichkeit zur Rekonfiguration der Ressourcenlandschaft besteht dabei je nach Lastsituation in der Anschaffung neuer oder Abschaltung bereits vorhandener physischer Hardware. Allerdings sind die erwähnten Lastschwankungen üblicherweise so stark, dass diese durch eine statische Rekonfiguration der Ressourcen nicht aufgefangen werden können, ohne sich an Hochlastzeiten zu orientieren.

Neben der Wirtschaftlichkeit rückt mit dem Schlagwort GreenIT zunehmend auch der ökologische Aspekt des Ressourcenbetriebs in den Mittelpunkt und motiviert die Effizienzsteigerung in der Nutzung (eher weniger) lokaler Ressourcen.

Zu diesem Zweck existieren zwei unterschiedliche Infrastruktur-Paradigmen, die es einem Ressourcenbetreiber erlauben, entweder seine lokale Ressourcenkonfiguration dynamisch zur Laufzeit zu verändern oder Teile seiner Arbeitslast auf Fremdressourcen auszulagern. Auf diese Art und Weise ist er in der Lage, die Abarbeitungsgeschwindigkeit der lokalen Rechenaufgaben auch in Hochlastsituationen zu steigern ohne dafür gleichzeitig neue physische Ressourcen anschaffen zu müssen.

Mit dem Grid-Computing ist (vornehmlich im wissenschaftlichen Umfeld) eine verteilte Infrastruktur gegeben, die es ermöglicht, die Ressourcen unterschiedlicher Betreiber oder wissenschaftlicher Communities zusammenzuschließen und so einen wechselseitigen Lastaustausch zwischen den (global verteilten) Ressourcenumgebungen zu etablieren. Die Teilnahme jeder einzelnen Partei am Grid ist in erster Linie davon abhängig, ob sie auf die Weise den Service für ihre lokale Nutzergemeinschaft und die Auslastung ihrer lokalen Ressourcen steigern kann.

Allerdings findet der Zusammenschluss unabhängiger Parteien zu einem Grid nicht zwangsläufig ohne Nebenbedingungen statt. So existieren für gewöhnlich bei jeder Partei bereits etablierte Verfahren zur lokalen Verteilung der Arbeitslast, die spezifisch an die eigenen Anforderungen angepasst sind und nach Möglichkeit unangetastet bleiben sollen.

Auch wenn jede Partei des Grids grundsätzlich zur Kooperation mit den übrigen Parteien bereit ist und auch sein muss, so ist die Bereitschaft zum Austausch dynamischer Lastinformationen meist gering.

Das kann zum einen politische Gründe haben, da diese Informationen Aufschluss darüber geben könnten, wie effizient die lokale Ressourcenlandschaft eines Betreibers verwaltet wird. Zum anderen führt ein vermehrter Austausch von Lastinformationen auch zu einer Erhöhung des Nachrichtenaustauschs zwischen den Parteien und damit auch zu einer zunehmenden Belastung der verbundenen Netze.

Daher empfiehlt es sich, Austauschalgorithmen zu entwickeln, die auch ohne den Austausch dynamischer Informationen zu einer möglichst fairen Performanz-

steigerung aller Parteien des Grids führen. Hierbei kann der Lastaustausch über die direkte Abgabe von Rechenaufgaben, der sogenannten Aktivitätendelegation, oder über die temporäre „Vermietung“ der Ressourcen einer Partei an eine andere durch Ressourcendelegation erfolgen.

Neben dem wechselseitigen Austausch der Arbeitslast in Form eines Grids bietet das Cloud Computing ein (in erster Linie für kommerzielle Betreiber) interessantes Infrastruktur-Paradigma für die einseitige Auslagerung von Arbeitslast.

In der Regel ist der wechselseitige Austausch von Arbeitslast lediglich durch die Verbesserung der Scheduling-Qualität aller Parteien motiviert und findet ohne monetären Ausgleich statt. Beim einseitigen Austausch der Last in eine Compute Cloud stehen der höheren Scheduling-Performanz monetäre Kosten für die Nutzung der geliehenen Ressourcen gegenüber. Somit entsteht ein mehrkriterielles Optimierungsproblem, welches entweder direkt mit mehrkriteriellen Optimierungsverfahren oder durch eine vorangehende Aggregation der gegensätzlichen Ziele zu einer Art Kosten-Nutzen-Funktion gelöst werden muss.

Um den Lastaustausch praktisch umzusetzen, existieren je nach Infrastruktur-Paradigma und Austauschform (Ressourcen oder Rechenaufgaben) unterschiedliche technische Lösungen, die in Form offener Standards, wissenschaftlicher Projekte oder kommerzieller Produkte fortwährend weiterentwickelt werden.

Abgesehen von der technischen Lösung einer jeden Methode zur Ressourcenerweiterung, bedarf es auch der Entwicklung neuer Algorithmen, die den Austausch von Arbeitslast steuern, da dieser aufgrund der Komplexität des Problems (viele Jobs und Maschinen) nicht von Hand vorgenommen werden kann.

Zur Lösung dieses Problems stehen neben dem einfachen Heuristikentwurf auch Methoden der Computational Intelligence (Fuzzy-Systeme, Evolutionäre Optimierung) und des Maschinellen Lernens (Reinforcement Learning) zur Verfügung, um die Effizienz des Lastaustauschs und damit auch die Servicequalität der entsprechenden Ressourcenumgebungen zu steigern.

Die vorliegende Arbeit eröffnet einen tieferen Einblick in die technischen und algorithmischen Aspekte für alle drei Formen des Lastaustauschs. Dabei werden bereits vorhandene Vorarbeiten im Bezug auf eigene Lösungen diskutiert. Zusätzlich werden die im Rahmen der Promotion erarbeiteten Strategien für den Lastaustausch detailliert beschrieben und evaluiert, um einen Vergleich zwischen den einzelnen Verfahren zu ermöglichen.

Aufbau der Arbeit

Die Arbeit ist in drei Teile gegliedert, wobei der erste Teil sich mit Grundlagen der Ressourcenerweiterung und der zur Optimierung des Austauschs verwendeten Verfahren befasst. Während der zweite Teil im Anschluss den wechselseitigen Lastaustausch über Grid-Computing behandelt, gibt der dritte Teil Aufschluss über Probleme der einseitigen Ressourcenerweiterung mit Hilfe von Cloud-Computing.

Teil I beginnt zunächst mit den Grundlagen des parallelen Computings und skizziert nachfolgend das vorliegende lokale Online-Scheduling-Problem für parallele Maschinen. Ausgehend von diesem Basismodell werden die Möglichkeiten zur Ressourcenerweiterung und das angenommene Job- und Maschinenmodell detailliert dargestellt. Im Zuge dessen werden ebenso gängige Notationen eingeführt wie auch Größen zur Bewertung der Scheduling-Qualität. Da die in dieser Arbeit diskutierten Strategien zum Lastaustausch auf Basis von Simulationen mit realen Lastaufzeichnungsdaten vorgenommen werden, sind sowohl die Software zur Simulation als auch die Lastaufzeichnungen an sich Bestandteile von Teil I. Abgeschlossen wird der Teil mit einer Einführung in Ein- und Mehrkriterielle Evolutionäre Optimierung sowie dem Reinforcement Learning.

Nach einer näheren Erläuterung von Computational Grids, werden in Teil II zunächst eine Grid-spezifische Erweiterung des Basismodells paralleler Maschinen vorgenommen und unterschiedliche Grid-Scheduling-Architekturen erläutert. Die exemplarische Beschreibung der praktischen Umsetzung einer Ressourcenerweiterung anhand eines geförderten Grid-Projekts liefert die technische Grundlage für die nachfolgenden algorithmischen Untersuchungen.

Im Anschluss teilt sich das Grid-Scheduling in die zwei besagten Austauschmethoden Aktivitäten- und Ressourcendelegation. Jede der beiden Methoden wird im Hinblick auf bereits bestehende algorithmischen Ansätzen und schließlich eigenen Strategien für den Lastaustausch diskutiert. Für die Aktivitätendelegation werden in Kapitel 7 neben einer einfachen Heuristik auch komplexere Strategien eingeführt, welche evolutionär optimierte Fuzzy-Regelbasen für die Entscheidungsfindung nutzen, die erst mit einer Evolutionsstrategie und später auch coevolutionär optimiert werden. Für die Ressourcendelegation wird in Kapitel 8 nach einer vorteilhaften Heuristik für alle Parteien des Grids gesucht und ein Vergleich mit der Performanz der Aktivitätendelegationsverfahren vorgenommen.

Der abschließende Teil III erläutert zunächst Begrifflichkeiten sowie technische bzw. algorithmische Voraussetzungen, die nötig sind, um Lastauslagerungen in eine Cloud vorzunehmen. Mit der anschließenden mehrkriteriellen Optimierung hinsichtlich Kosten und Scheduling-Qualität wird das Verbesserungspotential durch Cloud Scheduling festgestellt, welches in nachfolgenden Online-Lernverfahren mit Reinforcement Learning angenähert werden soll. Eine zusätzliche Vereinfachung des Lernverfahrens und eine Analyse beider Verfahren im Fehlerfall (temporärer Ausfall von Maschinen) runden das Thema Cloud Scheduling ab.

Teil I

Grundlagen

Kapitel 2

Scheduling in HPC-Cluster-Systemen

Wie die Vielzahl von Anwendungen im parallelen Rechnen, so unterscheiden sich auch die Begrifflichkeiten, die in diesem Kontext verwendet werden.

Beim parallelen Rechnen geht es grundsätzlich darum, ein zu berechnendes Gesamtproblem in mehrere Teilprobleme aufzuteilen. Hierbei können die parallelen Einzelprozesse aus Teilproblemen mit unterschiedlichem Zweck und damit auch unterschiedlichen Algorithmen bestehen. Unter Umständen ist ein Gesamtproblem auch über identische Teilprozesse parallelisiert, die jeweils auf einer Teilmenge der Eingabedaten arbeiten.

Der etwas vage Begriff *hochparalleles Rechnen* betont in diesem Kontext, dass ein zu berechnendes Gesamtproblem in sehr viele gleichzeitig laufende Teilberechnungen unterteilt wird.

Im wissenschaftlichen und industriellen Kontext war die Motivation für hochparalleles Rechnen schon lange gegeben. Diese lag vor allem in der Art der zu lösenden Probleme. Schließlich zeichnen sich komplexe Modellberechnungen zu Wetter, Physik oder Molekularbiologie — um nur einige zu nennen — dadurch aus, dass ihre Durchführung einen sehr großen Bedarf an Rechenleistung hat. Eine sequentielle Berechnung könnte nur unter großem Zeitaufwand geschehen.

Um daher die Leistungsfähigkeit von Computing-Architekturen für solche Anwendungsarten zu steigern, hat sich die Parallelisierung von Hardware auf allen Ebenen der Computerarchitektur durchgesetzt. So bestehen die vermeintlich kleinsten Bauteile zur Abarbeitung einzelner Befehle — die Prozessoren — bereits aus mehreren separaten Kernen (engl. *Core*).

Diese wiederum setzen mit dem internen *Pipelining* der Befehlsabarbeitung ebenfalls eine Parallelisierungsmethode ein, welches darüber hinaus unter Einsatz multipler Ausführungseinheiten *superskalar* sein kann. Die tatsächliche Größe der einzelnen Kerne hängt (abgesehen natürlich von der internen Struktur) am Ende maßgeblich von der Größe der atomaren Bausteine des Rechenwerks — den Transistoren — ab.

Bereits jetzt zeichnet sich mit der Entwicklung von Transistoren atomarer Größe [54] ab, dass die Integrationsdichte von Transistoren pro Chipfläche mittelfristig nicht weiter erhöht werden kann.

Deswegen und aufgrund der Komplexität der zu berechnenden Probleme hat sich gleichzeitig auch auf Schichten oberhalb der Prozessoren die Parallelität erhöht, z.B. über mehrere Prozessoren innerhalb eines Rechners/auf einem *Serverblade*, mehrere Rechner in einem Netzwerk/Blades innerhalb eines *Racks*, mehrere Netzwerke/Racks innerhalb eines lokalen Rechenzentrums bis zur Weitverkehrsvernetzung mehrerer Rechenzentren.

Neben den aufeinander aufbauenden Ebenen der Parallelisierung wurde — in Abhängigkeit der Anwendungen — spezielle parallele Hardware für die präzise Verarbeitung großer Fließkommazahlen erschaffen, wie beispielsweise Grafikkarten oder Co-Prozessoren.

Bei einer größeren Zahl vernetzter Rechnerknoten[1] spricht man allgemein von einem *Massively-Parallel-Processing (MPP)*-System. Die zumeist homogenen (baugleichen) Rechenknoten sind dabei über ein schnelles Netzwerk (*Interconnect*) miteinander verbunden. Im Gegensatz zu den in der TOP500-Liste[2] weltweit schnellsten Systemen, setzen sich die meisten MPP-Systeme aus handelsüblicher (Commodity-) Hardware zusammen und werden innerhalb von wissenschaftlichen Einrichtungen bzw. Unternehmen in Form von Rechenclustern betrieben.

Gleichzeitig kommen bei der Nutzung dieser parallelen Ressourcen im Grunde genommen zwei unterschiedliche Sichtweisen zum Tragen, die sich durch unterschiedliche Anforderungen an die Hardware auszeichnen. So steht bei der Verwendung von Supercomputern vor allem die enge Kopplung der Rechenknoten und die volle Ausnutzung der Leistungsfähigkeit der Hardware im Mittelpunkt. Dies erfordert bei der Entwicklung von Applikationen ein hohes Maß an Wissen über die Hardwarearchitektur und führt darüber hinaus zu einer starken Anpassung der Software an die unterliegende Hardware.

Die Entwicklung dieser angepassten Applikationen erfolgt meist über Jahre hinweg und beginnt oft schon gleichzeitig mit der Planung eines neuen Supercomputers, lange vor dessen Installation. Die Leistungsbewertung dieser Systeme findet über Größen wie Giga-Flops (Milliarden Fließkommaoperationen pro Sekunde) statt und bestimmt somit den Rang im internationalen Vergleich.

Beim Clustercomputing steht hingegen eher ein hoher Durchsatz von Applikationen bzw. komplexen Berechnungen gemessen über einen verhältnismäßig langen Zeitraum (Monate/Jahre) und zu geringeren Kosten als beim Supercomputing im Vordergrund. Um dies zu erreichen, wird in erster Linie auf eine möglichst breite Nutzung der zugrundeliegenden Standard-Hardware gezielt. Von Interesse sind Aspekte wie eine dynamische Skalierbarkeit des unterliegenden Ressourcenraums, beispielsweise das Einbinden von sich im Leerlauf befindenden Arbeitsplatzrechnern

[1] Damit ist ein Verbund aus Prozessor mit eigenem Speicher (Hauptspeicher/Festplatte) gemeint, wie dies z.B. bei einem einzelnen PC oder innerhalb eines Blades der Fall ist.

[2] `www.top500.org`, Zugriff: 16. Januar 2013.

oder die effiziente Planung (Scheduling) unterschiedlicher Applikationen mehrerer Nutzer innerhalb eines geteilten Ressourcenraums.

Innerhalb der Planung von Jobs auf MPP-Systemen (*MPP-Scheduling*) gibt es noch zwei verschiedene Nutzungsparadigmen, die sich vor allem in der Art der Jobs unterscheiden. Beim *High-Performance-Computing (HPC)* handelt es sich um Jobs unterschiedlicher Nutzer mit einer limitierten Eingabe und Rechenaufgabe, so dass die einzelnen Jobs nach einer bestimmten Zeit auch beendet sind und die ihnen zugeteilten Ressourcen für die Abarbeitung anderer Jobs freigeben. Beim *High-Throughput-Computing (HTC)* hingegen ist die Länge der Eingabe für eine bestimmte Applikation praktisch unbegrenzt. Hierbei suchen speziell für diese Anwendung ausgerichtete Planungssysteme wie Condor [75] im gesamten Ressourcenraum nach ungenutzten Ressourcen und füllen diese mit Anwendungen wie beispielsweise SETI@HOME[3] oder Einstein@Home[4] auf Basis der BOINC [104]-Plattform aus, die solange rechnen, bis sie von außen zu einer Freigabe der Ressourcen gezwungen werden.

Aufgrund ihrer Ausrichtung in Hinblick auf Skalierbarkeit und das Scheduling von Anwendungen mit dem Fokus auf eine möglichst rasche Abarbeitung der gesamten limitierten Arbeitslast, stellen die im Rahmen dieser Arbeit betrachteten MPP-Systeme Cluster-Systeme für das High-Performance-Computing dar.

Bei den Anwendungen innerhalb eines MPP-Systems handelt es sich gewöhnlicherweise um sogenannte Batch-Jobs (im Folgenden nur Jobs). Im Gegensatz zu interaktiven Anwendungen (Webservices, Hosting von Gastbetriebssystemen) erfordern diese nach Eingabe in das System keine weitere Interaktion seitens des Nutzers, sondern können mit Hilfe automatischer Abläufe innerhalb des Systems geplant und zur Ausführung gebracht werden. Der folgende Abschnitt erläutert zum einen das im weiteren Verlauf genutzte Basismodell für die Ressourcenplanung in MPP-Systemen und zum anderen wichtige Modellannahmen bzw. Algorithmen zur Planung.

[3] Software zur Auswertung von Radioteleskopdaten auf der Suche nach extraterrestrischer Intelligenz (http://setiathome.berkeley.edu, Zugriff: 16. Januar 2013).

[4] http://einstein.phys.uwm.edu, Zugriff: 16. Januar 2013.

2.1 Basismodell eines MPP-Systems

Das Basismodell eines MPP-Systems gliedert sich in drei elementare Bestandteile. Der erste Bestandteil ist eine lokale Nutzergemeinschaft. Diese kann sich prinzipiell auch aus diversen Untergruppen zusammensetzen und umfasst z.B. Studenten, Mitarbeiter einer Firma/Abteilung oder eine *Virtuelle Organisation (VO)*[5]. Diese Nutzergemeinschaft reicht — wie in Abbildung 2.1 dargestellt — Jobs in das System ein, die von einem Scheduling-System verarbeitet und an die unterliegenden Ressourcen weitergegeben werden.

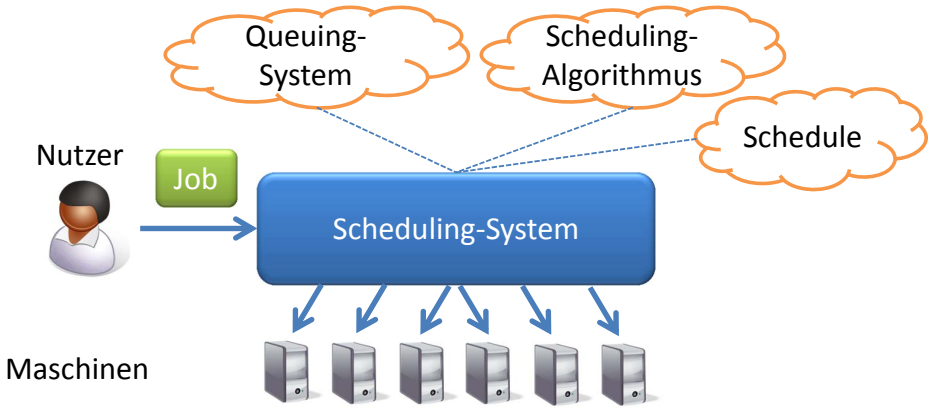


Abbildung 2.1: *Darstellung eines lokalen Scheduling-Systems (Basismodell)*

Im einfachsten Fall besteht dieses Scheduling-System[6] aus drei Komponenten, einem Queuing-System, dem eigentlichen Scheduling-Algorithmus und einem Schedule, welcher als Datenstruktur die momentane Belegung der unterliegenden Maschinen verwaltet. Moderne Batch-Systeme wie beispielsweise LoadLeveler [115], PBS [61] und LSF [137] dienen der Nutzergemeinschaft nicht nur als zentrale Einreichschnittstelle für ihre Jobs, sondern verfügen über eine Vielzahl von Konfigurationsmöglichkeiten. Hierzu zählen z.B. die Verwendung mehrerer Queues für unterschiedliche Nutzergruppen oder zur Sortierung von Jobs nach erwarteter Ressourcennutzung.

Ebenfalls möglich ist eine Priorisierung von Jobs einzelner Nutzer oder Queues. Dies erlaubt eine sehr feingranulare Konfiguration der Ressourcenzuteilung, die auch insgesamt zur einer besseren Planungsqualität führen. So haben beispielsweise Franke et al. [53] ein Verfahren diskutiert, welches unterschiedliche Nutzergruppen anhand von frei definierbaren gewichteten Zielfunktionen priorisiert und über eine Umsortierung von Jobs in der Queue realisiert.

[5] Eine Virtuelle Organisation wird in dieser Arbeit betrachtet als ein Zusammenschluss von Ressourcenbetreibern und Forschern (unter Umständen unterschiedlicher Einrichtungen oder Fachrichtungen), mit dem Zweck, ein gemeinsames Forschungsziel zu erreichen.

[6] In der Fachliteratur wird ein solches Scheduling-System auch häufig als Lokales Ressourcen Management System (LRMS) bezeichnet.

Für die eigentliche Entnahme von Jobs aus der Queue und Erstellung des Ausführungsplans (Schedule) wird ein Scheduling-Algorithmus konfiguriert.

In der Theorie lassen sich Scheduling-Algorithmen in Abhängigkeit des unterliegenden Modells grob in deterministische und probabilistische Scheduling-Algorithmen unterteilen.

Bei der Analyse von deterministischen Scheduling-Algorithmen geht es beispielsweise um die theoretische Untersuchung der Komplexität dieser Algorithmen [55, 24], die unter sehr restriktiven Modellannahmen agieren. So sind beim hierbei alle Charakteristika für die Ermittlung einer optimalen Lösung gegeben wie die Anzahl von Jobs und ihre Laufzeit. Alternativ existieren auch Ansätze wie beispielsweise die Arbeit von Xhafa et al. [127] bei denen Jobs mit einer absoluten Zahl von Instruktionen sowie eine feste Anzahl betrachteter Maschinen mit den von ihnen berechenbaren Instruktionen pro Sekunde (*MIPS*) gegeben sind, mit denen sich die Abarbeitungszeit der Jobs berechnen lässt.

In der Praxis handelt es sich bei dem Scheduling-Problem allerdings um ein probabilistisches Online-Scheduling-Problem, welches entsprechend mit Online-Scheduling-Algorithmen gelöst wird. In einem solchen Problem sind die Charakteristika von Jobs nur teilweise vorhanden. So werden diese hier z.B. über die Zeit in das System eingegeben und sowohl der Einreichungszeitpunkt als auch die Ausführungszeiten sind unbekannt (non-clairvoyant, vgl. Motwani et al. [88]). Letztere sind im besten Fall durch Laufzeitschätzungen beschränkt [5]. Darüber hinaus können in der Praxis eine Vielzahl von Fehlern [107] (z.B. Hardware, Software, Netzwerk) auftreten und nach Zhang et al. [135] zu einer Beeinflussung der Scheduling-Performanz eines MPP-Systems führen.

Beim Online-Scheduling stehen vor allem *Heuristiken* im Mittelpunkt, die ungeachtet der fehlenden Informationen sowie tolerant gegenüber Fehlern innerhalb des unterliegenden Systems zu vorteilhaften Lösungen gelangen — und dies möglichst unter Echtzeitbedingungen. Aufgrund der fehlenden Informationen kommt eine solche Heuristik höchstens zufällig zu einem „optimalen“ Schedule[7]. Aus diesem Grund wird bei der theoretischen Betrachtung eines Online-Algorithmus auf die kompetitive Analyse zurückgegriffen, bei welcher dessen Lösung im ungünstigsten Fall (engl. *worst case*) mit der optimalen Lösung eines Offline-Algorithmus verglichen wird.

In dem in dieser Arbeit verwendeten Basismodell und allen vorgestellten Algorithmen wird von einem Online-Scheduling-Problem ausgegangen, da dieses der Realität am meisten entspricht und damit die tatsächliche Anwendbarkeit der entwickelten Algorithmen erhöht. Nachfolgend werden zur Lösung des Onlineproblems zwei Standardheuristiken vorgestellt, die im weiteren Verlauf der Arbeit die Grundlage für das Online-Scheduling bilden.

[7]Optimalität bedarf stets auch der Definition eines Kriteriums, welches minimiert oder maximiert werden soll. Dies können z.B. Metriken zur Leistungsbewertung sein, wie sie später in Abschnitt 3.2 diskutiert werden.

2.1.1 First Come First Serve

First Come First Serve (FCFS) verläuft wie von Yahyapour [128] erläutert nach einem einfachen Muster. Zunächst werden neu eingereichte Jobs am Ende der Queue eingefügt, so dass die Jobs innerhalb der Queue stets nach ihrem Einreichungszeitpunkt sortiert sind. Stehen genügend Ressourcen für den ersten Job in der Queue zur Verfügung, wird dieser sofort zur Ausführung gebracht. Sollten nicht genügend Ressourcen zur Verfügung stehen, passiert nichts und erst die Freigabe von Ressourcen oder — in der Praxis oft eingesetzt — ein zeitlicher Auslöser sorgen für eine Neuevaluation des ersten Jobs.

Neben der Tatsache, dass FCFS in dem Sinne fair ist, dass die Jobs in ihrer ursprünglichen Einreichungsreihenfolge abgearbeitet werden (keine Priorisierung bestimmter Nutzergruppen oder Sortierung nach Parallelität), bedarf die Heuristik auch keiner Angaben von Ausführungszeiten und ist über einen einfachen Abgleich des derzeitigen Schedules und der Parallelität des ersten Jobs in der Queue zu realisieren.

Da FCFS stets nur den ersten Job in der Queue betrachtet, und daher nicht auf die Charakteristika der nachfolgenden Jobs eingeht, führt die Heuristik zu schlechten Resultaten für Jobsequenzen, bei denen sich Jobs mit einer hohen Parallelität abwechseln mit Jobs, die eine niedrige Parallelität besitzen. In einer solchen Situation kann es zu der so genannten Blockade-Situation [121] kommen, in der ein Job, der für seine Ausführung viele Prozessoren benötigt, während seiner Wartezeit nachfolgende Jobs in der Queue blockiert.

2.1.2 EASY

Das von Lifka [74] vorgeschlagene Extensible Argonne Scheduling System (EASY) versucht die in FCFS bestehende Fairness zu erhalten und gleichzeitig eine Lösung für die zuvor genannte Blockade-Situation zu bieten. Für den Fall, dass der vorderste Job in der Queue nicht ausführbar ist und somit die nachfolgenden Jobs blockiert, greift die Heuristik auf das Backfilling zurück, welches eine von den Nutzern angegebene Schätzung der Laufzeit mit in den Scheduling-Prozess einbindet.

Hierbei werden die nachfolgenden Jobs in ihrer bestehenden Reihenfolge durchsucht und auf ihre Ausführbarkeit überprüft. Ein so gefundener Job muss dann noch die Bedingung erfüllen, dass er bei seiner augenblicklichen Ausführung und dem aktuellen Stand der Informationen nicht zu einer Verzögerung des ersten Jobs führen darf.

Dies erfordert die Vorausplanung des vordersten Jobs, die nur möglich ist, da durch die gegebenen Laufzeitschätzungen der momentan ausgeführten Jobs ersichtlich ist, wann jener Job spätestens gestartet werden kann. Der Zeitraum zwischen der vorausberechneten Startzeit des vordersten Jobs und dem aktuellen Zeitpunkt wird in der Literatur auch *shadow time* [33] genannt. Ist die erwartete Laufzeit des betrachteten Jobs kleiner oder gleich diesem Zeitraum, so ist auch dieses Kriterium erfüllt und der Job kann vorgezogen werden. Ist sie dies nicht, wird die

Suche fortgesetzt, bis entweder ein Job gefunden wird, der die beiden Kriterien erfüllt oder das Ende der Queue erreicht ist. EASY wird neben FCFS häufig auf bestehenden Parallelrechnern eingesetzt (vgl. Skovira et al. [115]) und hat sich in Untersuchungen als sehr effizient erwiesen (vgl. Ernemann et al. [29]).

Der Performanz-Vorteil dieser Heuristik gegenüber FCFS hängt von der Qualität der Laufzeitschätzungen durch die Nutzer ab. Diese sind in der Realität oft mit der „walltime“ eines Jobs gleichgesetzt, also der Zeit, nach der ein vermeintlich fehlerhafter Job, der sich z.B. in einer Endlosschleife befindet, abgebrochen wird. Aus diesem Grund geben Nutzer oft zu große Schätzungen an, was nach Tsafrir et al. [125] wiederum die Qualität des Schedules beeinträchtigt.

2.2 Ein homogenes MPP-System auf heterogenen Ressourcen

Aus ökonomischen und administrativen Gründen handelt es sich bei heutigen Cluster-Systemen im Gegensatz zu Supercomputern in erster Linie um homogene Ressourcen (z.B. 100 baugleiche Bladeserver oder PCs der gleichen Produktgeneration). Mit der Erweiterung eines vorhandenen Clusters um Spezialressourcen (GPUs, Many Integrated Cores (MIC) [113]) bzw. neuer Commodity-Hardware mit anderen Charakteristiken (Architektur, Grad der internen Parallelisierung, Taktung) kann aber auch ein heterogener Cluster aus homogenen Subclustern entstehen.

Unter solchen Bedingungen sollte bei dem Entwurf von Scheduling-Algorithmen entweder darauf geachtet werden, dass diese nicht durch die systemweite Heterogenität beeinflusst werden (Partitionierung des Schedules in unterschiedliche Ressourcenarten) oder dass sie gezielt auf die Heterogenität eingehen (z.B. Jobs erst auf vermeintlich leistungsfähigeren Ressourcen ausführen und erst bei hoher Last auf weniger leistungsfähigen).

In dieser Arbeit wird von einem homogenen Ressourcenraum ausgegangen. Damit wird zum einen der Fokus auf die Optimierung des Job- bzw. Ressourcenaustauschs — weg vom lokalen Scheduling — gelegt und zum anderen einem in der IT-Welt vorliegenden Trend gefolgt — nämlich der Virtualisierung von Clustern, die nachfolgend genauer beschrieben werden soll.

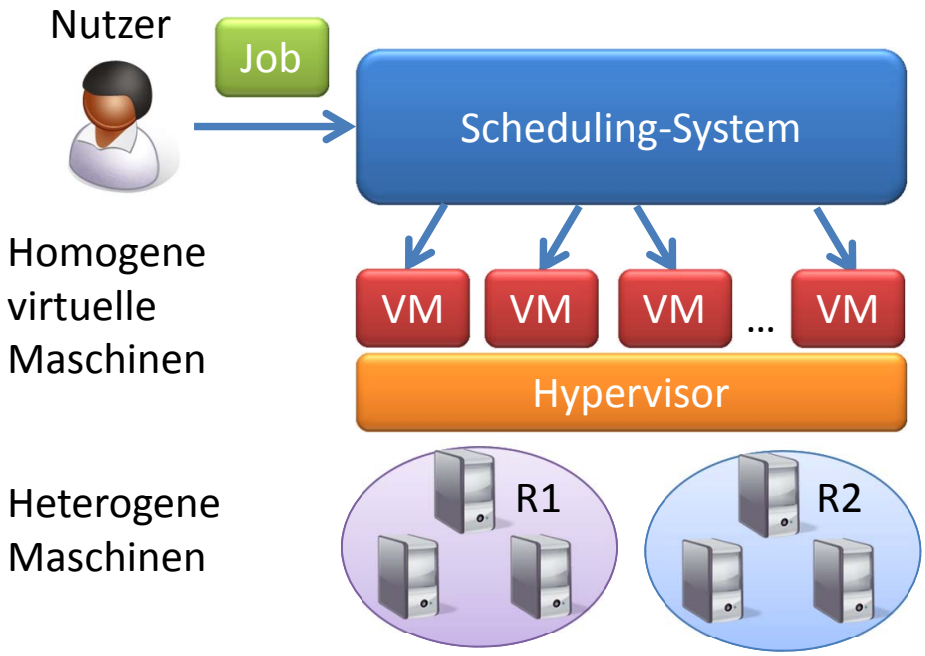


Abbildung 2.2: *Virtualisiertes lokales Scheduling-System*

Hierbei abstrahiert wie in Abbildung 2.2 dargestellt eine Virtualisierungsschicht von den physischen Ressourcen (Host-Systeme), so dass aus Sicht des Scheduling-Systems lediglich virtuelle Maschinen (VM) für die Verteilung der Arbeitslast angesprochen werden. Beispiele für die Virtualisierung von Clustern können in den Arbeiten von Fallenbeck et al. [31] und Nishimura et al. [93] gefunden werden. Vorteile einer solchen Abstraktionsschicht sind beispielsweise:

- eine bessere Ausnutzung der unterliegenden physischen Hardware. Manche Applikationen profitieren per se nicht von der internen Parallelität der Hardware (Mehrkernprozessoren). Die Ausführung mehrerer virtueller Maschinen auf einem Host-System (eine VM pro Kern) kann so zu einer höheren Gesamtleistung führen. Dies konnte unter anderem von Nussbaum et al. [94] nachgewiesen werden, die verschiedene Virtualisierungslösungen mit Hilfe des — im HPC-Bereich häufig zur Bewertung genutzten — LINPACK-Benchmarks auf ihre Performanz untersucht haben.

- eine anpassbare Ausführungsumgebung. Gerade im Bereich des Clustercomputing existiert neben der Ressourcenheterogenität auch eine große Heterogenität der Arbeitslast. Da die Anwender oft aus verschiedenen Bereichen kommen und somit auch verschiedene Programmiersprachen und Softwarebibliotheken einsetzen ist der Vorteil einer virtualisierten Plattform auch in der Anpassbarkeit der Abbilder virtueller Maschinen der sogenannten „Images“ gegeben. So könnte der Entwickler eines Rechenjobs ausgehend von einem durch den Betreiber der Hardware definierten Template einer virtuellen Maschine Anpassung an der darin installierten Software (Bibliotheken, Lizenzsoftware) vornehmen und sich somit eine eigene Ausführungsumgebung definieren, die unabhängig von der unterliegenden Hardware ist [95].

- die erhöhte Sicherheit durch Isolation der virtuellen Maschinen. Ein wichtiger Punkt bei der koordinierten Nutzung geteilter Ressourcen stellt die Isolation der Jobbearbeitung unterschiedlicher Nutzer dar. Durch die Kapselung der Jobbearbeitung innerhalb von virtuellen Maschinen wird eine zusätzliche Sicherheitsschicht eingeführt, die den Übergriff von Prozessen unterschiedlicher Nutzer auf die Daten und Prozesse der jeweils anderen Nutzer verhindert. Neben dem Schutz der in der VM-liegenden Daten und Prozesse ist hierbei gleichzeitig auch der Schutz des Host-Systems (Installation der physischen Hardware) vor schadhaftem Code innerhalb der VM gegeben. Dies macht unter anderem auch eine gezielte wissenschaftliche Untersuchung von schadhaftem Code auf einem MPP-System praktikabel, ohne das unterliegende System zu korrumpieren [56].

Neben den Vorteilen, die in erster Linie aus dem Anwendungsbereich der Serverkonsolidierung bei Hosting-Unternehmen kommen, existieren im HPC-Bereich auch Zweifel an der Performanz einer virtualisierten Infrastruktur gegenüber dem Einsatz nativer Komponenten. Der bei der Virtualisierung durch zusätzliche Schichten entstehende diskutierte Overhead teilt sich hierbei in einen CPU-Overhead

(Computational Overhead) und den Kommunikations-Overhead (I/O-Overhead) paralleler Anwendungen, die oft auch getrennt voneinander untersucht werden.

Neueste Entwicklungen konnten den CPU-Overhead weitestgehend auf unter 2% senken [94], was zum einen durch Paravirtualisierung [129] und zum anderen durch Prozessorvirtualisierung ermöglicht wird. Während bei einer vollständigen Virtualisierung ein unmodifiziertes Gastbetriebssystem eingesetzt wird und der zwischen der virtuellen und physischen Ebene liegende Hypervisor alle Instruktionen des Gastbetriebssystems auf Instruktionen des Host-Systems abbildet, werden bei Paravirtualisierung modifizierte Gastsysteme eingesetzt, die auf bestimmte Instruktionen verzichten, da sie bei der Übersetzung für das Host-System zu sehr großen Performanzeinbußen führen würden. So können die Instruktionen in vielen Fällen einfach vom Gastsystem an das Host-System weitergeleitet werden, was zu sehr großen Leistungssteigerungen gegenüber der Vollvirtualisierung führt.

Die Prozessorvirtualisierung stellt den Beitrag zur Virtualisierung der Systeme durch die Hardwarehersteller dar, die durch virtualisierungsorientierten Erweiterungen ihrer Instruktionssätze (z.B. privilegierte Instruktionen für den Hypervisor) zusätzlich zu einer Performanzsteigerung virtueller Infrastruktur auf ihren Prozessoren beitragen. Beispiele für diese Entwicklungen sind etwa Intels VT-x [91] oder AMDs AMD-V [1].

Im Zusammenhang mit dem Kommunikations-Overhead stehen sich die effizientere Ausnutzung der CPU-Kapazität durch das Betreiben mehrerer virtueller Maschinen auf einem einzelnen Host-System und die dafür benötigte Synchronisation beim Zugriff auf die geteilten I/O-Geräte (Festplatten, Arbeitsspeicher, Netzwerk) gegenüber. Ähnlich wie der CPU-Overhead durch Para- und Prozessorvirtualisierung gesenkt werden kann, gibt es auch im Bereich der I/O-Virtualisierung Fortschritte [72, 62], die in naher Zukunft auf eine weitgehend leistungsverlustfreie Virtualisierung von Cluster-Systemen hoffen lassen.

2.3 Möglichkeiten zur Erweiterung des Ressourcenraums

Ausgehend vom zuvor dargestellten Basismodell werden in dieser Arbeit Technologien und Algorithmen zur dynamischen Erweiterung des lokalen Ressourcenraums vorgestellt. Die Motivation für einen Betreiber von Rechenressourcen (staatliche Rechenzentren, Universitäten, Unternehmen) kann dabei sehr vielfältig sein.

So zeichnet sich das Scheduling auf dem unterliegenden MPP-System z.B. dadurch aus, dass die eingegebene Arbeitslast zum Teil starken Schwankungen unterworfen ist, die manchmal ganz profane Gründe haben wie den Tag-/Nacht-Wechsel, Urlaubszeiten der Nutzer oder die Bewilligung neuer Forschungsprojekte. Um auftretende Lastspitzen durch genügend Ressourcen abzufangen, müsste der Betreiber weitere physische Ressourcen installieren, was neben den eigentlichen Anschaffungskosten auch zu erheblichen Wartungs- und Betriebskosten führen würde. Gleichzeitig müsste er bei Unterlast Verfahren einsetzen, die nicht benötigte Ressourcen

deaktivieren und bei Bedarf wieder reaktivieren. Doch auch durch temporäre Abschaltung von Ressourcen fallen versteckte Kosten für die Instandhaltung und den belegten Platz an. Da Mechanismen zur automatischen Aktivierung/Deaktivierung oft nicht genutzt werden bzw. eventuelle Reaktionszeiten bei der Reaktivierung vermieden werden sollen, werden auch momentan ungenutzte Ressourcen häufig einfach weiterbetrieben. In jedem Fall ist der Betrieb bzw. die physische Erweiterung von Ressourcen unter ökonomischen Gesichtspunkten kritisch zu betrachten.

Doch auch aus ökologischen Gründen ist eine lokale Erweiterung zu hinterfragen, da damit auch der Bedarf an elektrischer Energie sowohl im Hinblick auf die eigentliche IT als auch auf aktive Kühlung der Komponenten steigt. Die Einsparung elektrischer Energie ist einer der Hauptpunkte der sogenannten GreenIT und daher sind Verfahren zur effizienteren Nutzung vorhandener Ressourcen bzw. zur möglichst energiearmen Erweiterung der Ressourcen neben der Performanzsteigerung bei Lastspitzen auch ökologisch sinnvoll.

Ein weiterer Grund für die Erweiterung um externe Ressourcen kann ebenfalls sein, dass ein Rechenzentrum im Falle einer wechselseitigen Erweiterung auch selbst zum Anbieter von Ressourcen wird, wenn die lokale Arbeitslast dies zulässt.

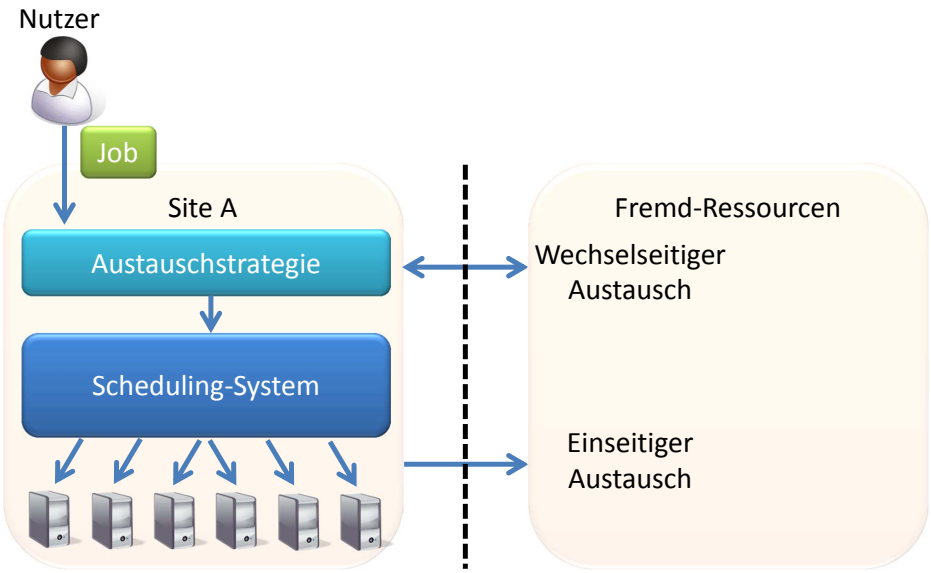


Abbildung 2.3: *Erweitertes Basismodell*

Abbildung 2.3 liefert eine anschauliche Darstellung einer Ressourcenerweiterung. Hierbei wird das Basismodell ergänzt um eine Austauschstrategie, welche den Austausch von Arbeitslast je nach verwendetem Verfahren in die eine und/oder andere Richtung steuert.

Ein bekanntes Beispiel für eine wechselseitige Erweiterung ist das *Grid Computing*. Dieser von Foster & Kesselman [51] geprägte Begriff beschreibt eine Computing-Architektur, bei der verteilte Ressourcen (regional oder auch global verteilt) zu einem Verbund zusammengeschlossen werden. In der Praxis werden solche

Zusammenschlüsse oft über die Durchführung eines gemeinsamen Forschungsvorhabens motiviert und sollen dann einer bestimmten Nutzergemeinschaft (Community) zur koordinierten Nutzung zur Verfügung stehen. Ein solches Grid ist z.B. mit dem für die Klimaforschung entwickelten *C3Grid*[8] [60] oder dem für die Medizin geschaffenen *MediGRID* [70] gegeben. Bei diesen (auch *Computational Grids* genannten) Infrastrukturen nutzen die jeweiligen Nutzergemeinschaften die Ressourcen für die verteilte Speicherung von Forschungs- und Sensordaten und reichen Jobs ein, die z.B. der Extraktion oder Analyse einer bestimmten Datenmenge dienen oder aufwendige Simulationen durchführen (Berechnung von Wettermodellen, Abgleich von Gensequenzen).

Hierbei rührt der Begriff „Grid“ von der Analogie des Modells zu einem Stromnetz (engl. powergrid) her, bei dem der elektrische Strom für den Endverbraucher unabhängig von seinem Einspeisungsort oder den zur Erzeugung verwendeten Rohstoffen aus der Steckdose kommt. In einem Computational Grid sind gewünschte Daten und Rechenleistung für den Nutzer auf ähnlich transparente Weise zugreifbar.

Dieser Community-getriebene Zusammenschluss von Ressourcen zu einem Grid kann auch als Erweiterung der lokalen Ressourcen jedes einzelnen Ressourcenbetreibers gesehen werden, denn in der Regel bedienen Ressourcenbetreiber nicht nur die Anforderungen einer einzelnen wissenschaftlichen Community sondern besitzen darüber hinaus eine lokale Nutzergemeinschaft, die sich aus Mitgliedern diverser Communities zusammensetzt. Die Ressourcen eines Betreibers werden in der Grid-Terminologie auch *Site* (Standort) genannt. In diesem Fall dient das Grid Computing Paradigma dann dem Austausch von Arbeitslast zwischen unterschiedlichen Sites. In Teil II werden technische und algorithmische Ansätze für den Arbeitslastaustausch mittels Grid Computing diskutiert.

Eine zusätzliche Möglichkeit zur Erweiterung des Ressourcenraums, die lediglich einseitigen Arbeitslastaustausch einsetzt, ist die Erweiterung um Cloud-Ressourcen. Dabei werden Ressourcen in Form virtueller Maschinen über einen bestimmten Zeitraum und gegen Bezahlung gemietet. Der Unterschied zum Grid ist also zum einen die einseitige Nutzung der Ressourcen und die damit verbundene monetäre Vergütung und zum anderen die Möglichkeit der potenziell unbeschränkten exklusiven Nutzung der erweiternden Ressourcen.

Aus Sicht der Nutzergemeinschaft ist auch dieser Zugriff transparent. Schließlich ist es einem Nutzer egal, ob die Rechenkraft, von der seine Arbeitslast bewältigt wird, „aus der Steckdose“ kommt oder sich „in einer Wolke“ (engl. Cloud) befindet. Auf letzteres Austauschszenario wird insbesondere in Teil III eingegangen.

[8] Collaborative Climate Community Data and Processing Grid.

Kapitel 3

Modellannahmen für Job- und Maschinenmodell

Sowohl Job- als auch Maschinenmodell dieser Arbeit basieren auf den von Garey & Graham [55] vorgenommenen Vereinfachungen zur Beschreibung von Jobs und Maschinen, die bereits in vielen Arbeiten zu Scheduling in MPP-Systemen genutzt wurden und sich auch mit den Dokumentationen üblicher MPP-Systeme decken [64, 79]. Hierbei werden die unterliegenden Maschinen als homogen und vollvernetzt angesehen.

Die Homogenität begründet sich in unserem Fall durch die Betrachtung des MPP-Systems als Verbund von virtuellen Maschinen einer gleichen Klasse, wie dies z.B. auch bei den virtuellen Infrastrukturen heutiger Compute-Cloud-Anbieter üblich ist. Cloud-Infrastrukturanbieter teilen die bereitgestellte Infrastruktur oft in Klassen ein (tiny, medium, large bzw. Bronze, Silber, Gold). Jede dieser Klassen besitzt dabei unterschiedliche Leistungscharakteristiken (Speicher, Prozessorkerne, Geschwindigkeit) und unter Umständen auch unterschiedliche Garantien zur Servicequalität (z.B. Verfügbarkeit).

Vollvernetzt heißt, dass die virtuellen Maschinen alle direkt (gleich schnell) miteinander kommunizieren können und keine Partitionierung der Ressourcen besteht, wie es z.B. in manchen Supercomputern bei kubischer Vernetzung [36] der Fall ist.

Als Resultat dieser beiden Annahmen können Jobs auf jeder Untermenge von VMs gleich schnell ausgeführt werden. Das ist vor allem wichtig für die spätere Evaluation aller Scheduling-Algorithmen, da vorgegebene Partitionierungen bei heterogenen Ressourcen zum einen eine Einschränkung für die Platzierung paralleler Jobs bedeuten würden. Zum anderen müsste das spätere Arbeitslastmodell (vgl. Abschnitt 4.1) zusätzlich um eine Funktion zur Berechnung der Abarbeitungszeit jedes einzelnen Jobs auf jeder Partition ergänzt werden.

Ähnliche Überlegungen gelten für den „internen Jobaufbau“ wie von Feitelson & Rudolph [37] beschrieben. So existieren bei der Jobmodellierung die Begriffe *moldable* bzw. *malleable*, bei denen Jobs nach Einreichen in das System in unterschiedlichen Parallelitätsgraden betrieben werden können. In der Praxis muss eine

Applikation entsprechend angepasst sein, damit sie mit mehreren Parallelitätsgraden ausgeführt werden kann.

Moldable Jobs sind diejenigen, bei denen der Nutzer eine Menge möglicher Parallelitätsgrade bzw. einen Wertebereich angibt und das Scheduling-System zur Planungszeit entscheiden kann, mit welchem Parallelitätsgrad der Job gestartet werden soll.

Kann sich der Parallelitätsgrad zusätzlich nach Beginn der Ausführung eines Jobs ändern, so ist der Job sogar malleable.

Für die Simulation eines solchen Systems und zur Erprobung und Vergleich unterschiedlicher Scheduling-Algorithmen muss für jeden Job eine Speedup-Funktion existieren, so dass für den Job in Abhängigkeit des Parallelitätsgrades bestimmbar ist, welche Laufzeit er hat. Eine solche Funktion ist jedoch sehr jobabhängig und kann nach Nguyen et al. [92] bereits zwischen unterschiedlichen Implementierungen derselben Applikation stark abweichen. So kann die Funktion nur für eine bestimmte Ressourcenkonfiguration und für eine bestimmte Implementierung durch viele Messungen angenähert werden und ist im Anschluss nicht übertragbar auf andere Hardware.

Da eine allgemeingültige Speedup-Funktion für alle Applikationen unter allen Hardwarekonfigurationen nicht verfügbar ist, werden die Jobs im überwiegenden Teil aller simulativen Scheduling-Untersuchungen — wie auch hier — als *rigid* angenommen also mit einem ab Veröffentlichung bis zum Ende der Ausführung unveränderlichen Grad an Parallelität, der vom Nutzer vorgegeben wird.

Des Weiteren findet die Zuordnung von VMs zu parallelen Jobs nach dem beim HPC üblichen *space-sharing*-Prinzip statt. Hierbei werden die VMs den Jobs für die Dauer ihrer Laufzeit exklusiv zur Verfügung gestellt, um interne Seiteneffekte bei der Ausführung paralleler Programmteile auszuschließen, die auftreten würden, wenn sich unterschiedliche Jobs die den VMs zugeteilten Ressourcen teilen müssten.

Die Unterbrechung (engl. *preemption*) von Jobs nach Beginn der Ausführung wird nicht betrachtet. Somit ist auch keine Migration von Jobs zwischen aktiven VMs zur Laufzeit möglich. Eine Migration von VMs zwischen physischen Hosts ist nicht verboten, doch weder die dadurch eventuell auftretenden Seiteneffekte (Leistungseinbrüche, Fehler bei der Übertragung) noch die tatsächliche technische Umsetzung (Hypervisor über physischem MPP-System) werden in dieser Arbeit berücksichtigt.

Auch wenn es viele theoretische Überlegungen [112] zum Nutzen von preemptiven Scheduling gibt, so findet bei produktiven HPC-Systemen in der Praxis keine Unterbrechung der Jobs statt [68]. Die Unterstützung von Preemption würde bei der Implementierung der Applikationen (insbesondere bei parallelen) größere Anforderungen an den Entwickler stellen wie Checkpointing[1] oder eine aufwändigere Prozesskoordination zwischen parallelen Prozessen, damit der neben dem einzelnen Prozessstatus existierende Status der Netzwerkkommunikation ebenfalls vor einer Pausierung gespeichert wird.

[1] Explizite Implementierung von Rücksetzpunkten einer Applikation für das spätere Fortsetzen.

Die von Kettimuthu et al. [68] benannte Gefahr, dass ein Job potentiell nie abgearbeitet wird (*starvation*), erhöht sich mit der Möglichkeit, ihn zu pausieren und ist ein weiterer Grund, warum sich preemptive Verfahren in der Praxis nicht durchgesetzt haben.

Bei den betrachteten Jobs handelt es sich stets um atomare Jobs ohne Abhängigkeiten. Für *Workflow*-Scheduling sei auf entsprechende Quellen von Yu et al. [130, 131] sowie Senkul & Toroslu [114] verwiesen.

Darüber hinaus wird stets davon ausgegangen, dass alle Jobs erfolgreich abgeschlossen werden. Ein Modell für fehlerhafte Jobs und damit verbundene Mechanismen für die Neuplanung von Jobs, wie von Li et al. [73] und Yu et al. [132] vorgestellt, sind nicht Teil der Arbeit.

3.1 Formale Einführung von Job und Maschinenmodell

Da die Planung der Jobs nach dem space-sharing-Prinzip vorgenommen wird, lässt sich ein Schedule als eine Anordnung von (nicht überlappenden) Jobblöcken innerhalb eines zweidimensionalen Raums aus VMs über die Zeit darstellen. Abbildung 3.1 gibt ein Beispiel für eine solche Anordnung und visualisiert zudem übliche Größen [99] zur Beschreibung von Job und Schedule-Charakteristika. Der Ressour-

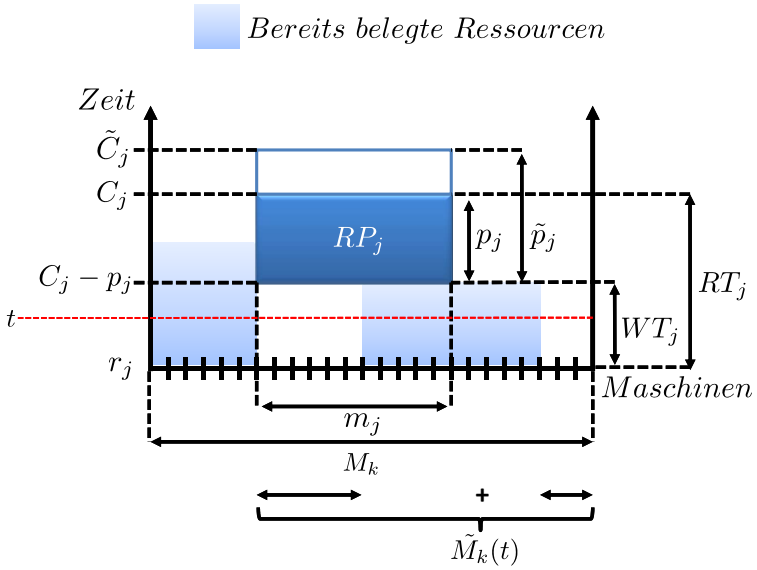


Abbildung 3.1: *Formelle Charakteristika von Job und Schedule*

cenraum eines MPP-Systems k wird beispielsweise über die maximale Anzahl M_k von verfügbaren Maschinen (x-Achse) über die Zeit (y-Achse) definiert[2]. Für einen einzelnen Job j sind ab seinem Einreichungszeitpunkt r_j (*release time*) sowohl seine Anzahl benötigter Maschinen m_j als auch eine geschätzte Laufzeit $\tilde{p}_j$ (*estimated*

[2]In der Literatur sind oft auch Darstellungen mit vertauschten Achsen zu finden.

processing time) bekannt. In Abhängigkeit der bereits belegten Ressourcen kann der Job dann frühestens zu dem Zeitpunkt gestartet werden, an dem genügend parallele Maschinen zu seiner Ausführung bereitstehen. Dies ist genau dann der Fall, wenn die zeitabhängige Größe $\tilde{M}_k(t)$, welche die Anzahl freier Maschinen zu einem Zeitpunkt t beschreibt größer oder gleich m_j ist. Zu diesem Zeitpunkt wird dem Job eine ausreichend große Teilmenge freier Maschinen zugewiesen, welche er in Folge exklusiv für eine unbekannte Dauer p_j (*processing time*) belegt. Die Dauer ist dabei stets kleiner oder gleich der geschätzten Laufzeit ($p_j \leq \tilde{p}_j$). Somit ist der späteste Beendigungszeitpunkt $\tilde{C}_j$ eines Jobs bereits ab Beginn seiner Ausführung bekannt. Aufgrund dieser Tatsache ist die Berechnung der shadow time bei EASY-Backfilling möglich (vgl. Abschnitt 2.1.2). Der tatsächliche Beendigungszeitpunkt C_j ist erst mit dem Ende der Jobbearbeitung bekannt. Der Startzeitpunkt kann dann ebenfalls als Differenz von C_j und p_j dargestellt werden. Die Zeitspanne zwischen der Veröffentlichung des Jobs r_j und seinem Startzeitpunkt $C_j - p_j$ stellt die Wartezeit WT_j (*wait time*) eines Jobs dar, wohingegen die Antwortzeit RT_j (*response time* oder *turnaround time*) die gesamte Zeit umfasst, die der Job im System gewesen ist. Sie berechnet sich daher aus der Differenz von C_j und r_j.

3.2 Bewertungsgrößen zur Feststellung der Scheduling-Qualität

Bei der Bewertung der Scheduling-Qualität können grundsätzlich zwei verschiedene Perspektiven eingenommen werden — die Perspektive einzelner Nutzer oder die Perspektive von Ressourcenbetreibern. Die Nutzer haben dabei ein vorrangiges Interesse an der schnellen Bearbeitung ihrer Jobs, während Ressourcenbetreiber an der effizienten Ausnutzung ihrer Ressourcen interessiert sind. Im Sinne eines serviceorientierten Betriebs können die Betreiber ebenfalls die Interessen der Nutzer vertreten. So ist es auch aus Betreibersicht sinnvoll, die Anforderungen der Nutzer nach schneller Abarbeitung der Jobs zu erfüllen. Spätestens im Wettbewerb zwischen einzelnen Ressourcenbetreibern sind Qualitätsmerkmale wie die Zugänglichkeit, Stabilität, Kosten und auch Abarbeitungsgeschwindigkeiten entscheidend für eine nach Downey & Feitelson [23] fluktuierende Nutzergemeinschaft.

Im Folgenden werden unterschiedliche Bewertungsgrößen für die Performanz von Scheduling-Algorithmen vorgestellt, die im Laufe der Arbeit genutzt werden.

3.2.1 Produktionsspanne und Abarbeitungszeit eines Schedules

Die Produktionsspanne (engl. *makespan*) bezeichnet die Abarbeitungszeit eines Schedules und damit die Zeitspanne, die vom Einreichen des ersten Jobs bis zur Abarbeitung des letzten Jobs vergeht. Dabei wird der Zeitpunkt der Fertigstellung des letzten Jobs auch C_{max} genannt, in Anlehnung an den in Abschnitt 3.1 beschriebenen Beendigungszeitpunkt C_j eines Jobs. So ergibt sich C_{max} aus dem Maximum der Beendigungszeitpunkte aller bearbeiteten Jobs. In der Literatur werden C_{max}

und Produktionsspanne häufig gleichgesetzt [99], was im Online Scheduling-Fall nur gilt, wenn der erste Job auch zum Zeitpunkt $t = 0$ eingereicht wird. Im Rahmen dieser Arbeit werden nur Eingabedaten verwendet, bei denen auch zum Zeitpunkt 0 Jobs vorhanden sind. Daher können C_{max} und Produktionsspanne auch in diesem Fall synonym verwandt werden.

Dabei ist zu beachten, dass die Jobs im Online-Fall nach und nach in das System eingegeben werden (vgl. Abschnitt 2.1 auf S. 10), so dass die Produktionsspanne unabhängig vom eingesetzten Scheduling-Algorithmus stark von den letzten eingereichten Jobs abhängt. Aus diesem Grund wird die Produktionsspanne oft nur zu theoretischen Überlegungen hinzugezogen [55, 90] oder dient als Hilfsmittel zur Berechnung weiterer Bewertungsgrößen (vgl. Abschnitt 3.2.3).

3.2.2 Ressourcenprodukt (RP)

Das Ressourcenprodukt (RP_j) bezeichnet die Ressourcennutzung eines Jobs j als Produkt aus der Anzahl paralleler Maschinen m_j und der Zeit der Nutzung p_j. In Anlehnung an Abbildung 3.1 entspricht es demnach der genutzten Fläche des Ressourcenraums, die von dem Job eingenommen wird.

Des Weiteren kann das Ressourcenprodukt auch zur Berechnung der Gesamtarbeitslast einer Menge von Jobs genutzt werden. Sei beispielsweise mit π_k eine Menge von Jobs gegeben, die auf dem MPP-System k berechnet werden, so entspricht die Gesamtarbeitslast RP_k der kumulierten Arbeitslast aller Jobs und wird durch Gleichung (3.1) berechnet.

$$\mathrm{RP}_k = \sum_{j \in \pi_k} m_j \cdot p_j \tag{3.1}$$

Das Ressourcenprodukt eines gesamten MPP-Systems ist für sich gesehen von geringem Interesse. Erst in Verbindung mit weiteren Kenngrößen (z.B. Auslastung in Abschnitt 3.2.3 bzw. AWRT in Abschnitt 3.2.4) bzw. unter Einsatz von Verfahren zur Erweiterung des Ressourcenraums (vgl. Abschnitt 7.2.1) und den damit verbundenen Änderungen der lokal bearbeiteten Last, findet das Ressourcenprodukt Verwendung.

3.2.3 Auslastung (U)

Die Auslastung (engl. utilization) eines MPP-Systems k beschreibt das Verhältnis aus genutzten zu den insgesamt verfügbaren Ressourcen für einen festgelegten Zeitraum. Auf Basis des zuvor definierten Ressourcenprodukts RP_k lässt sie sich somit durch Gleichung (3.2) beschreiben.

$$U_k = \frac{\mathrm{RP}_k}{C_{max} \cdot M_k} \tag{3.2}$$

Der verfügbare Ressourcenraum wird bei der Berechnung der Auslastung demnach durch das Produkt aus den parallelen Maschinen M_k des MPP-Systems und der

gesamten Produktionsspanne berechnet und steht den genutzten Ressourcen RP_k gegenüber.

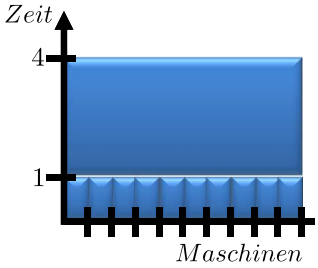
Aufgrund der Produktionsspanne ist auch die Auslastung im Online-Fall stark abhängig vom zuletzt eingereichten Job.

3.2.4 Durchschnittliche gewichtete Antwortzeit (AWRT)

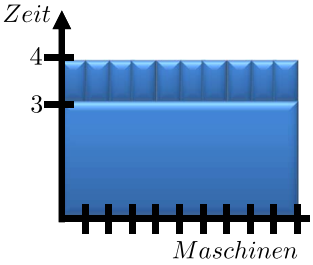
Mit der durchschnittlichen gewichteten Antwortzeit (engl. Average Weighted Response Time) ist nach Schwiegelshohn & Yahyapour [112] eine Bewertungsmetrik gegeben, die vom letzten Einreichungszeitpunkt der bearbeiteten Jobs unabhängig ist, da sie lediglich die Antwortzeit RT_j (vgl. Abschnitt 3.1 auf S. 21) und Ressourcennutzung RP_j von Jobs berücksichtigt.

$$\mathrm{AWRT}_k = \frac{\sum\limits_{j \in \tau_k} \mathrm{RP}_j \cdot \mathrm{RT}_j}{\sum\limits_{j \in \tau_k} \mathrm{RP}_j} \tag{3.3}$$

Gleichzeitig kann durch die in Gleichung (3.3) ersichtliche Gewichtung der Antwortzeit aller Jobs einer Jobmenge τ_k erreicht werden, dass keine Bevorzugung sequentieller vor parallelen Jobs stattfindet. Abbildung 3.2 zeigt das ursprünglich von Schwiegelshohn [110] gegebene Beispiel für eine solche Bevorzugung bei Verzicht auf die faire Gewichtung erzielter Antwortzeiten. Hierbei ergibt sich bei gleicher



(a) $\sum \mathrm{RT}_j = 14$; $\sum \mathrm{RP}_j \cdot \mathrm{RT}_j = 130$



(b) $\sum \mathrm{RT}_j = 43$; $\sum \mathrm{RP}_j \cdot \mathrm{RT}_j = 130$

Abbildung 3.2: *Beispiel für unterschiedliche Schedules bei gleicher Jobmenge mit unterschiedlichen Ergebnissen in Gesamtantwortzeit aber gleicher AWRT.*

Auslastung des Systems je nach Anordnung der Jobs unterschiedliche Ergebnisse hinsichtlich der Gesamtantwortzeit ($\sum \mathrm{RT}_j$) und damit implizit auch durchschnittlichen Antwortzeit ein geringeres Ergebnis, wenn die sequentiellen Jobs vor dem parallelen Job ausgeführt werden. Hinsichtlich der AWRT ist das Ergebnis in beiden Fällen gleich. Dieser Umstand würde auch dann bestehen bleiben, wenn der parallele Job in mehrere sequentielle Jobs aufgeteilt werden würde. Aus diesen Gründen eignet sich die AWRT insbesondere für eine betreiberseitige Bewertung der eingesetzten Scheduling-Verfahren unabhängig von der Aufteilung der Arbeitslast.

Im Gegensatz zu den vorherigen Bewertungsgrößen wird die AWRT in dieser Arbeit nicht auf der Menge der bearbeiteten Jobs (π_k) berechnet sondern auf der Menge (τ_k) von Jobs, die bei einer Site lokal eingereicht wurden. Damit kann diese Größe zur Beurteilung der Community-orientierten Scheduling-Qualität bei Austausch von Arbeitslast zwischen unterschiedlichen Sites eingesetzt werden.

3.2.5 Gesamtwartezeit (TWT)

Die zuvor beschriebene AWRT beruht auf der Antwortzeit RT und ist damit neben der in einem Schedule enthaltenen Wartezeiten auch von den — auf homogenen Ressourcen konstanten — einzelnen Laufzeiten der bearbeiteten Jobs abhängig. Dieser Umstand ist ebenfalls aus den später in Abschnitt 4.3 präsentierten Referenzergebnissen ersichtlich. Außerdem liegt das Interesse der Nutzer eines MPP-Systems unter dem gegebenen Modell in erster Linie in dem zügigen Start ihrer Anwendung, wodurch eine Fokussierung auf die durch das Scheduling entstehende Wartezeit anstatt der Antwortzeit sinnvoll sein kann. Die Gesamtwartezeit (Total Wait Time) einer Jobmenge τ_k lässt sich durch die Gleichung (3.4) berechnen.

$$\mathrm{TWT}_k = \sum_{j \in \tau_k} \mathrm{WT}_j \tag{3.4}$$

Bei Betrachtung dieser nutzerorientierten Metrik muss darauf geachtet werden, dass sie, ähnlich wie bei der Gesamtantwortzeit, zu einer Bevorzugung sequentieller vor paralleler Jobs führen kann und daher nur eingesetzt werden sollte, wenn das unterliegende Modell — wie in unserem Fall — keine Aufteilung von parallelen Jobs zulässt bzw. keine gezielte Umstellung von Jobs hinsichtlich dieses Kriteriums vorsieht. Im Gegensatz zu antwortzeitbasierten Metriken kann die Gesamtwartezeit unabhängig von der bearbeiteten Jobmenge durch Mechanismen zur Erweiterung des Ressourcenraums auf 0 reduziert werden — ein Umstand, der sich insbesondere in Teil III zunutze gemacht wird.

Kapitel 4

Simulation von parallelen Systemen

Für die theoretische Untersuchung von Scheduling-Algorithmen werden häufig einfache Beispiele aus wenigen Jobs mit festgelegten Charakteristika (Parallelität, Laufzeit etc.) und Systeme niedriger Parallelität konstruiert.

Da die in der Realität existierenden Systeme mit einer großen Anzahl paralleler Maschinen ausgestattet sind und zusätzlich über ein großes Arbeitslastaufkommen verfügen, findet die Evaluation von Scheduling-Algorithmen in Bezug auf die zuvor vorgestellten Bewertungsgrößen simulativ statt.

Zusätzlich zu der größeren Komplexität des Problems müssen sich neu entwickelte Strategien auch auf mehreren parallelen Infrastrukturen bewähren, um ihre Robustheit und Scheduling-Performanz nachzuweisen. Eine Nachbildung mit realen Systemen oder der direkte (ungetestete) Einsatz im Produktivbetrieb wäre allerdings zu aufwändig bzw. kostspielig.

Für die Simulation paralleler Systeme sind vor allem zwei Aspekte entscheidend. Zum einen bestimmt schon die Wahl der Eingabedaten darüber, welches Verbesserungspotential ein neuer Scheduling-Algorithmus besitzt bzw. wie aussagekräftig die erzielten Ergebnisse für die spätere praktische Anwendung sind. Zum anderen muss ein geeignetes Simulations-Framework gewählt werden, welches zu den Anforderungen der zu untersuchenden Algorithmen bzw. zu der modellierten parallelen Infrastruktur passt.

4.1 Auswahl von Eingabedaten

Bei der Simulation von Scheduling-Algorithmen sollen die Eingabedaten (engl. Workload) auf möglichst realistische Art und Weise die Jobeinreichcharakteristik einer Nutzergemeinschaft abbilden. Daher existieren für die Auswahl der Eingabedaten grundsätzlich zwei verschiedene Herangehensweisen. Bei der ersten werden reale Arbeitslastaufzeichnungen für die Simulation genutzt. Diese werden z.B. aus Logfiles bei dem Betrieb von Parallelrechnern oder Aufzeichnungen bei der Nut-

zung von Grid-Sites durch Communities erstellt. Besondere Verbreitung haben die von D.G. Feitelson im *Parallel Workloads Archive*[1] gesammelten Aufzeichnungen bei der Simulation von MPP-Systemen gefunden. So erstreckt sich deren Nutzung nicht nur über die Simulation von Parallelrechnern [111, 123] sondern auch auf Computational Grids [28], Mehrkernarchitekturen [134] oder Cloud Computing Infrastrukturen [124].

Der Vorteil von realen Arbeitslastaufzeichnungen liegt insbesondere darin, dass sie versteckte Muster in Bezug auf das Nutzerverhalten umfassen. Dies wird z.B. von Feitelson [32] als das „Verhalten menschlicher Nutzer“ beschrieben. Dabei sind bestimmte Faktoren wie Tag- und Nachtzyklen[2] oder eine Korrelation zwischen einem Nutzer und z.B. der zu erwartenden Laufzeit seiner Jobs noch verhältnismäßig leicht zu identifizieren. Ähnlich sieht es mit der Parallelität von Jobs aus, bei der sich durch eine Analyse mehrerer Workloads gezeigt hat, dass der Parallelitätsgrad eingereichter Jobs auffällig oft eine Zweierpotenz darstellt [23].

Schwieriger wird es hingegen bei der Abschätzung der sogenannten *interarrival times*, die in etwa der Frequenz von Arbeitslasteinreichungen entsprechen. Dieser Umstand liegt vor allem darin begründet, dass das Nutzerverhalten eng gekoppelt ist mit der derzeitigen Servicequalität. Müssen Nutzer z.B. bereits sehr lange auf die Fertigstellung ihrer Jobs warten, so neigen sie dazu, weniger Jobs einzureichen und umgekehrt. Ähnlich verhält es sich mit Fluktuationen innerhalb der Nutzergemeinschaft, die neben der Voraussage über die zu erwartende Arbeitslast für einzelne Nutzer auch noch die Dimension über zu erwartende Nutzer hinzufügt. Auch wenn all diese nutzerpsychologischen Effekte in der Praxis durchaus auftreten, lassen sie sich bei der simulatorischen Erprobung von Algorithmen schwer nachbilden.

Nichtsdestotrotz gibt es viele Veröffentlichungen, die sich mit der statistischen Modellierung synthetischer Workloads befassen. Weit verbreitet ist das sogenannte *sampling* von realen Workloads, bei dem bestimmte Eigenschaften der Jobs wie die besagten interarrival times, Parallelitätsgrade, Laufzeiten der Jobs usw. durch Wahrscheinlichkeitsverteilungen angeglichen werden. Durch Kombination dieser Verteilungsfunktionen über einen weiteren wahrscheinlichkeitsbasierten Prozess werden dann Workloads erzeugt, die der Vorlage möglichst ähnlich sein sollen. So konnte Feitelson [34] beispielsweise durch Analyse von sechs verschiedenen realen Aufzeichnungen ein Modell erstellen, welches einzelne Jobparameter unabhängig voneinander generiert und so Workloads erzeugt, die für jeden einzelnen Parameter gesehen ähnliche Charakteristika aufweisen wie die Originalaufzeichnung. Mit dem Hinzufügen von Expertenwissen — wie eine gewisse Korrelation zwischen dem Parallelitätsgrad und der Laufzeit von Jobs — konnten Lublin & Feitelson [79] ein frei konfigurierbares Workload-Modell entwickeln, welches die Nachbildung mehrerer existierender Aufzeichnungen ermöglicht. Die tatsächlichen Parameterkonfigurationen für die enthaltenen Verteilungen müssten zur Nachbildung bestimmter

[1] http://www.cs.huji.ac.il/labs/parallel/workload/, Zugriff: 16. Januar 2013.

[2] Tagsüber werden oft kurze Testjobs zum Validieren der Software eingereicht, wohingegen die tatsächlichen Experimente eher nachts laufen.

Workloads immer neu evaluiert werden, da sie zwischen den einzelnen Workloads laut Downey [22] stark variieren können. Darüber hinaus läuft man bei der Konfiguration synthetischer Workloads nach Feitelson [35] ebenfalls Gefahr, dass sich bei der Abarbeitung der generierten Workloads keine Konvergenz betrachteter Bewertungsgrößen in Bezug auf den Original-Workload einstellt. Und obwohl es synthetische Workloads erlauben, vorlagenähnliche Aufzeichnungen größerer Länge zu generieren, konnten Lo et al. [76] zeigen, dass es nahezu keine qualitativen Unterschiede in der Performanz mehrerer lokaler Scheduling-Strategien im Hinblick auf unterschiedliche Bewertungsgrößen gibt, unabhängig davon, ob synthetische oder reale Arbeitslastaufzeichnungen verwendet wurden. Des Weiteren wiesen die Autoren darauf hin, dass ähnlich der Parameterkonfiguration synthetischer Workloads auch reale Workloads mit Bedacht eingesetzt werden müssen.

Aus diesen Gründen wurden für die Simulation aller Algorithmen in dieser Arbeit reale Arbeitslastaufzeichnungen aus dem Parallel Workloads Archive verwendet. Diese liegen im von Chapin et al. [12] definierten *Standard Workload Format (SWF)* vor und wurden — wie von Lo et al. [76] empfohlen — nach Kriterien von Feitelson & Tsafrir [39, 38] bereinigt. Die Bereinigungen enthalten beispielsweise Maßnahmen wie die Entfernung von Jobs, die keine oder eine negative Anzahl von Maschinen zur Ausführung aufweisen oder die Beseitigung einmalig auftretender Ereignisse, wie Wartungen und Abstürze von Maschinen aus den Aufzeichnungen. Dazu zählen auch sogenannte *Workload flurries* [38] in denen ein einzelner Nutzer über einen kurzen Zeitraum den gesamten Workload dominiert, sei es aufgrund von Fehlern in der Programmierung oder der Aufzeichnung. Da es sich in diesen Fällen statistisch betrachtet um nicht repräsentative Einzelfälle handelt, sprechen sich Feitelson & Tsafrir [38] mit Bezug auf analoge Aussagen durch Cirne & Berman [14] für eine Entfernung dieser Einzelfälle aus.

Besonders nützlich bei der Konfiguration eines Simulationsaufbaus sind die mitgelieferten Konfigurationen der parallelen Maschinen, von denen die Aufzeichnungen gewonnen wurden, da eine einfache Übertragung von Aufzeichnungen von einer Maschine auf eine andere laut Lo et al. [76] nicht möglich ist.

Neben den zuvor genannten Aufzeichnungen von Parallelmaschinen existieren im *Grid Workloads Archive (GWA)*[3] auch Aufzeichnungen realer Grid-Installationen, die durch Erweiterung des Standard Workload Formats auch noch Informationen über VO-Zugehörigkeit von Nutzern oder Abhängigkeiten von Jobs innerhalb von Workflows beinhalten. Die Verwendung dieser Aufzeichnungen würde gegenüber der Verwendung derjenigen aus dem Parallel Workloads Archive jedoch keinen Vorteil bringen, da im Rahmen des in dieser Arbeit verwendeten Jobmodells (vgl. Kapitel 3) alle Jobs als unabhängig betrachtet und erweiterte Informationen über Nutzer- bzw. Gruppenzugehörigkeit nicht genutzt werden. Darüber hinaus ist bei der Auswahl der Eingabedaten besonders die Auslastung ein kritischer Faktor, da es wenig Sinn macht, Algorithmen über die Ressourcenraumerweiterung anhand von Eingabedaten zu untersuchen, die von einer unterlasteten Maschinenumgebung

[3] http://gwa.ewi.tudelft.nl, Zugriff: 16. Januar 2013.

stammen. Gerade unter diesem Gesichtspunkt erweisen sich die Aufzeichnungen aus dem GWA als unzureichend, da von den elf enthaltenen Aufzeichnungen nur eine über eine Auslastung von 41 bis 60 Prozent verfügt[4] und bei den meisten gar keine Informationen über die Auslastung zur Verfügung stehen.

Bei der Auswahl representativer Workloads aus dem Parallel Workloads Archive wurde neben der Auslastung noch auf zwei weitere Kriterien geachtet. Erstens sollten die Maschinengrößen ein möglichst breites Spektrum an üblichen Konfigurationen abdecken, also von klein (etwa 100 Maschinen) bis groß (fast 2000 Maschinen) und dabei Ausreißer außerhalb dieses Bereichs unbeachtet lassen. Das zweite Kriterium bezieht sich auf die Länge der Aufzeichnungen. Diese bewegen sich innerhalb des Archivs zwischen einem und 42 Monaten, müssen für die Evaluation von Arbeitslastaustauschverfahren allerdings eine vergleichbare Länge besitzen. Als Resultat der Überlegungen wurden fünf verschiedene Aufzeichnungen identifiziert, die im Folgenden für alle Untersuchungen genutzt werden und deren Charakteristika in Tabelle 4.1 zusammengefasst sind.

Bezeichnung	Archivnummer	#Jobs n	Anzahl Maschinen M	RP
KTH	8	28 479	100	2 017 737 644
SDSC00	9	29 810	128	2 780 016 139
CTC	6	77 199	430	8 279 369 703
SDSC03	12	65 584	1 152	23 337 438 904
SDSC05	20	74 903	1 664	29 392 365 928

TABELLE 4.1: *Auf 11 Monate gekürzte Eingabedaten aus dem Parallel Workloads Archive (Sortiert nach Maschinengröße)*

Die Abkürzungen der Aufzeichnungen beziehen sich auf die Institutionen, welche für den Betrieb der zugrunde liegenden Maschinen verantwortlich gewesen sind und falls nötig das Kürzel des Endjahres, an dem die Aufzeichnung erhoben wurde.

So stammt beispielsweise der Workload *KTH* vom Schwedischen Königlichen Institut für Technologie (Stockholm) und wurde vom Oktober 1996 bis August 1997 aufgezeichnet. Dabei wurden auf der 100 Prozessoren-"IBM SP2"-Maschine insgesamt 28479 Jobs eingereicht, die ein Ressourcenprodukt von über 2 Milliarden CPU-Sekunden aufweisen. Innerhalb des Archivs besitzt er die Nummer 8.

Die Abkürzung *CTC* steht für das *Cornell Theory Center* und enthält eine ebenfalls von einer „IBM SP2"-Maschine erhobene Aufzeichnung, die vom Juni 1996 bis Mai 1997 erhoben wurde. Mit 430 Prozessoren fällt diese Maschine jedoch um einiges größer aus als ihr schwedischer Verwandter.

Die übrigen Aufzeichnungen stammen alle vom *San Diego Supercomputer Center* aber aus unterschiedlichen Jahren und von unterschiedlichen Maschinenkonfigurationen. Zwischen den Jahren hat am SDSC stets eine Vergrößerung der Prozessorenanzahl ($128 \rightarrow 1152 \rightarrow 1664$) und wechselseitig eine entsprechende Erhöhung der Arbeitslast pro Aufzeichnungsdauer stattgefunden (vgl. RP). Außerdem wurde

[4] Dies entspricht der Kategorie 3 einer im GWA definierten Einteilung.

zwischen den Aufzeichnungen SDSC03 und SDSC05 ein Architekturwechsel vorgenommen, weg von der zur damaligen Zeit weit verbreiteten „IBM SP2“ hin zu einem „IBM eServer pSeries 655/690“-System. Zwischen den Workloads differieren die Original-Aufzeichnungsdauern zwischen 13 und 32 Monaten. Damit es im weiteren Verlauf der Arbeit möglich ist, Algorithmen für den Arbeitslastaustausch zwischen den Maschinen zu untersuchen, wurden die längeren Aufzeichnungen nach 11 Monaten abgeschnitten, so dass alle Eingabedaten eine vergleichbare Länge besitzen. Daher sind in der Tabelle 4.1 alle Kenndaten bezogen auf die gekürzten Versionen abgetragen.

Abschließend sei erwähnt, dass die Verwendung realer Aufzeichnungsdaten auch dazu beiträgt, die Forschung reproduzierbarer zu machen. Trotzdem muss bei realer Anwendung der in dieser Arbeit vorgestellten Algorithmen auf ein Modell mit unterschiedlichen Rahmenbedingungen (z.B. abweichende Workload-Charakteristiken[5]) stets überprüft werden, ob die erzielten Ergebnisse in dem Fall ihre Gültigkeit behalten.

4.2 Auswahl eines Frameworks zur Simulation paralleler Systeme

Für die Simulation von parallelen Infrastrukturen existiert eine Vielzahl von Frameworks mit unterschiedlichen Ausrichtungen im Hinblick auf die zu evaluierenden Probleme.

So haben Chen & Chen [13] beispielsweise eine Simulationsumgebung für MPP-Systeme vorgestellt. Diese zielte vor allem auf die Optimierung der unterliegenden Hardware, also die Emulation eines Multiprozessorsystem mit geteiltem Speicher ab. Im Fokus lagen hier beispielsweise die Organisation von Caches, Anpassung eines Instruktionssatzes und die Simulation der Lösungen auf einem Cluster. Einen ähnlichen Fokus haben Zheng et al. [136] mit BigSim gesetzt, welches Performanzabschätzungen für sehr große Maschinen (z.B. Blue-Gene/L) auf Basis von Simulationen auf kleineren Systemen noch vor der Implementierung der großen Maschine liefern soll. Für die Erprobung von Scheduling-Algorithmen für Cluster- bzw. verteilte Systeme sind beide Frameworks ungeeignet.

Eine Alternative stellt das von Song et al. [117] vorgestellte *MicroGrid* dar, welches in erster Linie versucht, die Struktur eines Computational Grids innerhalb eines lokalen Clusters nachzubilden. Dabei setzt es in seiner aktuellen Version auf das im Grid-Umfeld häufig genutzte und initial von Foster & Kesselmann [52] vorgestellte Globus Toolkit (in der Version 1.1) auf und konzentriert sich auf die Erprobung von verschiedenen Grid-Strukturen im Hinblick auf Vernetzung oder

[5] Die verwendeten Aufzeichnungen sind teilweise sehr alt ($\geq$ 16 Jahre) und stammen von Parallelrechnern mit sehr spezifischem Aufbau hinsichtlich geteiltem Speicher oder der Vernetzung der einzelnen parallelen Knoten. Trotzdem ist ihre Verwendung heutzutage weitestgehend alternativlos aufgrund der geschilderten Schwachstellen der statistischen Workload-Modelle und übrigen Aufzeichnungsdaten.

heterogene Hardware. Für die Performanzbewertung werden sowohl Mikrobenchmarks eingesetzt, welche einzelne Aspekte der emulierten Hardware (Speicher, I/O-Zugriffe) bewerten als auch parallele Benchmarks, wie NAS [4] (Numerical Aerodynamic Simulation)[6], die eher auf die Evaluation der Kommunikation zwischen parallelen Maschinen spezialisiert sind. Aufgrund seiner Hardwarenähe, der Realisierung über das Globus Toolkit (in einer sehr alten Version) und der Tatsache, dass das Projekt offensichtlich schon seit 2004 nicht mehr weiterentwickelt wird, scheidet auch dieses Framework für die Nutzung aus.

Mit GridSim wurde von Buyya & Murshed [9] ein Simulations-Framework für verteilte Infrastrukturen entwickelt, welches sich im Speziellen für die Evaluation marktökonomischer P2P-Ansätze (*Peer-to-Peer*) im Kontext der Ressourcenallokation eignet. Hierbei wird jedem Nutzer und jeder Ressource ein eigener Broker zugeordnet, der dann entweder die Ziele eines Nutzers (geringer Ausführungspreis, schnelle Abarbeitung) oder die des Ressourcenbetreibers (hoher Gewinn, Maximierung der Auslastung des Systems) vertritt. Weiterhin können technische Gegebenheiten (Netzwerke inklusive Bandbreiten und Latenzen, heterogene Ressourcen mit Geschwindigkeit in MIPS) sehr detailliert abgebildet werden, was im Rahmen der in dieser Arbeit vorgestellten Modelldefinition nicht von Interesse ist. Leider ist das Framework erst seit ca. zwei Jahren in der Lage, reale Workload-Daten wie die aus dem Parallel Workloads Archive zu verarbeiten und wurde zu Beginn der Forschungen daher nicht zur Simulation der Algorithmen genutzt.

Auch mit dem von Casanova [10] bzw. Casanova et al. [11] entwickelten SimGrid wurde ein Framework zur detaillierten Simulation von verteilten Systemen implementiert. Der damalige Fokus auf die möglichst realistische Simulation von Netzwerkstrukturen, sowie auf time-sharing-basiertes Scheduling mit einer einzelnen Scheduling-Instanz sprach in diesem Fall ebenfalls gegen eine Verwendung.

Ausgehend von den bestehenden Anforderungen (homogene Ressourcen, Verwendung von realen Arbeitslastaufzeichungen aus dem Parallel Workloads Archive, dezentrales Scheduling bzw. Erweiterung von Ressourcenräumen, leichte Integration der eigentlichen Simulation in CI-Methoden zur Optimierung von Scheduling-Strategien) wurde daraufhin begleitend zur Forschung das Teikoku Grid-Scheduling-Framework (tGSF) [59] entwickelt. Dieses sehr schlanke Framework konzentriert sich auf die schnelle Simulation des zuvor eingeführten Job- und Maschinenmodells und erlaubt ebenfalls eine unkomplizierte Implementierung von Bewertungsgrößen bzw. verschiedenen Scheduling-Strategien für zentrale und dezentrale Scheduling-Algorithmen unter Verwendung realer Eingabedaten mit anschließender Integration in Lernverfahren zur Optimierung.

[6] http://www.nas.nasa.gov/publications/npb.html, Zugriff: 16. Januar 2013.

4.3 Referenzergebnisse bei der Simulation realer Arbeitslastaufzeichnungen

Um eine spätere Vergleichbarkeit der Ergebnisse im Hinblick auf Änderungen in den zuvor besprochenen Bewertungsgrößen herzustellen, wurden die ausgewählten Eingabedaten sowohl mit FCFS als auch mit EASY als lokalen Scheduling-Algorithmus in tGSF simuliert. Zusätzlich wurden alle Eingabedaten noch einmal in Sequenzen zu 5 und nachfolgenden 6 Monaten unterteilt, da diese bei später eingesetzten Lernverfahren als Lern- bzw. Übertragungsdaten verwendet werden[7].

Die Tabelle 4.2 enthält daher auch die Ergebnisse der Bewertungsgrößen und die grundlegenden Charakteristika beider Teilsequenzen nach der Teilung.

			FCFS				EASY			
Abkürzung	n	RP ($\cdot 10^6$)	C_{max} ($\cdot 10^6$)	U	AWRT	TWT ($\cdot 10^6$)	C_{max} ($\cdot 10^6$)	U	AWRT	TWT ($\cdot 10^6$)
KTH-11	28 479	2 018	29	68,67	418 172	10 426	29	68,72	75 158	193
KTH-5	11 780	893	14	64,84	488 387	5 298	13	68,55	83 325	104
KTH-6	16 699	1 125	16	68,52	99 236	790	16	68,59	68 615	89
SDSC00-11	29 810	2 780	30	72,49	266 619	6 350	29	74,96	73 606	314
SDSC00-5	13 494	1 183	13	71,28	67 717	364	13	71,28	60 435	111
SDSC00-6	16 316	1 597	17	73,38	413 957	5 985	16	77,90	83 362	203
CTC-11	77 199	8 279	29	65,70	58 593	1 319	29	65,70	52 938	244
CTC-5	35 360	3 566	13	63,74	57 898	580	13	63,75	52 324	93
CTC-6	41 839	4 713	16	67,05	59 118	739	16	67,05	53 403	151
SDSC03-11	65 584	23 337	30	68,58	72 418	2 586	29	68,74	50 772	272
SDSC03-5	32 606	10 690	13	71,38	83 719	1 494	13	71,38	56 433	165
SDSC03-6	32 978	12 616	17	66,14	62 826	1 091	17	66,40	45 973	107
SDSC05-11	74 903	29 392	29	60,12	70 579	2 024	29	60,17	54 954	329
SDSC05-5	28 184	9 999	13	45,94	56 925	484	13	45,95	47 904	71
SDSC05-6	46 719	19 393	16	70,97	77 464	1 525	16	71,08	58 513	253

TABELLE 4.2: *Referenzergebnisse für die vereinheitlichten Eingabedaten (*11 *Monate) und ihre jeweiligen Lern- und Übertragungssequenzen (*5 *bzw.* 6 *Monate). Hierbei ist das Ressourcenprodukt (RP) des gesamten Workloads in Prozessorsekunden, und für beide genutzten Scheduling-Algorithmen jeweils C_{max} in Sekunden, die Auslastung (U) in Prozent und die AWRT sowie die Summe der Wartezeiten aller Jobs (TWT) ebenfalls in Sekunden gegeben.*

Die Unterteilung von Eingabedaten in zwei unabhängige Teilsequenzen ist unter zwei Voraussetzungen vertretbar. Zum einen weisen die einzelnen Teilstücke nach der zuvor angesprochenen Bereinigung realer Arbeitslastaufzeichnungen ähnliche Charakteristika in Bezug auf das Nutzungsverhalten der Ressourcen durch die Nutzergemeinschaft auf [38] (z.B. die Einreichung von Jobs im Tag-/Nacht-Zyklus bzw. Wochenarbeitszeiten). Zum anderen werden die Workloads in späteren Simulationskonfigurationen immer nur mit ihren entsprechenden Pendants eingesetzt, damit die von Feitelson & Tsafrir [38] erwähnten Schulungseffekte der Nutzer bei

[7] Dieses Konzept zur Aufteilung und Simulation der Eingabedaten wurde erstmals im Rahmen einer gemeinsamen Veröffentlichung [42] vorgestellt.

der Verwendung der Ressourcen bzw. die Anlaufphasen innerhalb der ersten Wochen einer jeden Aufzeichnung bei allen konfigurierten Maschinen gleichermaßen gegeben sind.

Dennoch ist zu beachten, dass es — mit Ausnahme der Jobzahlen n und des Ressourcenprodukts — nicht möglich ist, die Ergebnisse einer Bewertungsgröße für die Teilsequenzen zu kombinieren, um so das Ergebnis aus der 11-Monatssimulation zu reproduzieren. Das liegt daran, dass bei Simulation einer 6-Monats-Sequenz eine neue Einschwingphase erzeugt wird, die bei durchgängiger Simulation über die gesamte Laufzeit nicht vorhanden wäre und diese führt zu differierenden Ergebnissen bzgl. Bewertungsgrößen wie der Auslastung oder auch Summe der Wartezeiten.

In allen Simulationen zeigen sich die Vorteile, die sich bei der Berücksichtigung von Laufzeitschätzungen für das lokale Scheduling ergeben. So können bei der Anwendung von EASY-Backfilling sowohl TWT als auch AWRT stark gesenkt werden. Besonders auffällig sind die Verbesserungen für den KTH bei 5 und 11 Monaten, sowie den SDSC00-Workload bei 6 und 11 Monaten. Die AWRT sinkt beim KTH in beiden Fällen um über 80% und beim SDSC00 um über 70%. Dies ist zurückzuführen auf die in Abschnitt 2.1.1 beschriebene Blockade-Situation, da bei beiden Aufzeichnungen innerhalb der Sequenzen untypische Jobs mit großer Parallelität enthalten sind, die dann bei Ausführung von FCFS zu einer Verzögerung aller nachfolgenden Jobs führen. Dies kann offensichtlich durch das Vorziehen von nachfolgenden Jobs erheblich verbessert werden. Für diese beiden Maschinenaufzeichnungen könnte also auch eine Erweiterung des Ressourcenraums von Nutzen sein, auch wenn das lokale Scheduling ohne Laufzeitschätzungen auskommen muss.

Darüber hinaus gibt es keine Aufzeichnung und kein Aufzeichnungssegment, welches für FCFS eine bessere Performanz aufweist als für EASY. Weiterhin bestätigt sich, dass C_{max} für die Performanzmessung nicht aussagekräftig ist, da es z.B. für CTC-11 unabhängig vom Scheduling-Algorithmus vollständig identisch ist[8]. Hier zeigt sich, dass trotz Reduktion der Wartezeit aller Jobs um insgesamt über 81% und Reduktion der AWRT um über 9%, C_{max} oftmals nur vom Einreichzeitpunkt der letzten Jobs eines Workloads abhängig ist. Da C_{max} und die Auslastung U eng miteinander gekoppelt sind, gilt diese Aussage auch für die Auslastung.

[8] In der Tabelle wurden die Resultate aus Darstellungsgründen auf Werte $\cdot 10^6$ gerundet. Die tatsächlichen sekundengenauen Resultate sind bei einigen Aufzeichnungen aber auch identisch.

Kapitel 5

Optimierungsverfahren für die Erweiterung lokaler Ressourcen

Neben der technischen Umsetzung von Erweiterungsmechanismen für lokale Ressourcen sollen in dieser Arbeit vor allem auch Strategien zur Verbesserung der Scheduling-Qualität durch Ressourcenerweiterung vorgestellt werden. Im Fokus stehen hierbei sowohl Heuristiken als auch lerngestützte Verfahren zum Austausch von Arbeitslast (vgl. Kapitel 2.3). Dabei kommen unterschiedliche Optimierungsverfahren zum Einsatz, die im Folgenden grundlegend beschrieben werden.

Grundsätzlich lassen sich die verwendeten Optimierungsverfahren in zwei verschiedene Klassen unterteilen. Dies sind zum einen Offline-Optimierungsverfahren, bei denen das zugrundeliegende Problem als geschlossenes System betrachtet und die modellierte Konfiguration einer Strategie hinsichtlich einer oder mehrerer Performanzkriterien optimiert werden.

Dabei ist maßgeblich, dass sich die Konfiguration der Strategie und damit ihr Verhalten während einer einzelnen Evaluation auf Basis verwendeter Eingaben nicht verändert. Eine Veränderung der Konfiguration findet nur zwischen den Evaluationen statt.

Im Gegensatz dazu wird bei Online-Optimierungsverfahren die Strategiekonfiguration an eine sich ändernde Umwelt angepasst mit dem Ziel die Performanz langfristig zu maximieren.

In unserem Beispiel ist das zugrundeliegende Problem ein Online-Scheduling-Problem bei dem entweder Sites zu einem Computational Grid zusammengeschlossen oder lokale Ressourcen um öffentliche Cloud-Ressourcen erweitert werden. Das Modell ist durch das in Kapitel 3 beschriebene Job- und Maschinenmodell sowie durch eine der betrachteten Erweiterungsformen gegeben (Grid/Cloud). Die Eingaben des Modells bestehen dabei aus den konkreten Maschinenkonfigurationen (Setup) und verwendeten Lastaufzeichnungen. Die Strategiekonfigurationen bestimmen das Austauschverhalten von Arbeitslast zwischen den Parteien und sollen derart optimiert werden, dass sich das Gesamtsystem möglichst effizient im Hinblick auf Bewertungsgrößen aus Abschnitt 3.2 verhält. Die Evaluation von Kon-

figurationen und damit unterschiedlichen Strategien erfolgt dabei mit Hilfe von Simulationen.

Bei einer Offline-Optimierung resultieren die optimierten Strategiekonfigurationen aus dem impliziten Expertenwissen, welches durch das wiederholte Auswerten unterschiedlicher Parametersätze unter den immer gleichen Modell- und Eingabebedingungen ergeben. Die Online-Optimierung hingegen versucht einen sinnvollen Parametersatz bereits während eines einzigen Laufs zu finden und funktioniert daher nur in einem Modell, in dem sich die Auswirkungen einzelner Parameter auf das Gesamtsystem schon während dieses Laufs bewerten lässt.

5.1 Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) basieren auf der Idee, Prinzipien der von Darwin beschriebenen natürlichen Evolution für die Optimierung mathematischer oder technischer Systeme zu nutzen. Sie werden meist für Probleme eingesetzt, für die nur wenig Expertenwissen über die Parametereinstellungen oder den unterliegenden Lösungsraum vorhanden ist.

Ähnlich der natürlichen Evolution durchlaufen Spezies hierbei einen Kreislauf aus Selektion sowie beständiger Reproduktion und Variation ihrer Erbinformationen. Während durch Reproduktion und Variation neue Individuen — also Lösungen — produziert werden, stellt die Selektion ein Überleben und damit eine Fortentwicklung guter Lösungen zu einem (lokalen) Optimum hin sicher. Damit folgt die gesamte Spezies iterativ einem Anpassungsprozess bezüglich der Umwelt (das mathematische oder technische System). Innerhalb eines einzelnen Iterationsschritts (*Generation*) wird eine Menge von *Individuen* (*Population*) bezüglich ihrer Lösungsqualität in der Umwelt (*Fitness*) bewertet.

Wichtige Bestandteile eines EAs sind die *Kodierung* der Individuen und die Operatoren für die Reproduktion und Variation. Die Kodierung entspricht hierbei der mathematischen Beschreibung möglicher Lösungen des zu optimierenden Systems, während die Operatoren im Rahmen zufallsbasierter Prozesse dafür eingesetzt werden, neue Lösungen auf Basis vorangehender Lösungen zu erzeugen.

Historisch bedingt haben sich EAs in drei verschiedenen, zunächst weitgehend unabhängigen „Schulen“ entwickelt. So ist die Unterklasse von Algorithmen der Evolutionären Programmierung (EP) beispielsweise durch Fogel et al. [49] mit dem Ziel begründet worden, endliche Zustandsautomaten zu optimieren. Holland [63] und später Goldberg [58] hingegen haben zur Lösung kombinatorischer Optimierungsprobleme die Klasse der Genetischen Algorithmen (GA) begründet, während Rechenberg [101] und Schwefel [108] zur Lösung ingenieurstechnischer Optimierungsprobleme die Evolutionsstrategien (ES) entwickelt haben.

Die Unterschiede zwischen Algorithmen der verschiedenen Klassen sind nach heutigem Stand größtenteils verschwunden. Dies gilt insbesondere für Evolutionsstrategien und Genetische Algorithmen.

Da im Rahmen dieser Arbeit vor allem Evolutionsstrategien für die *Einkriterielle Optimierung* eingesetzt werden, wird für die Darstellung der wichtigen Aspekte evolutionärer Algorithmen auf die Prinzipien und Notationen der Evolutionsstrategien nach Schwefel [109] zurückgegriffen.

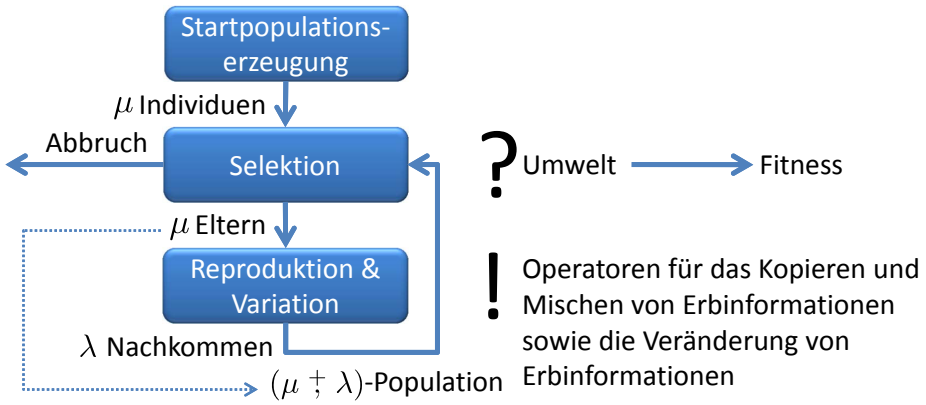


Abbildung 5.1: *Optimierungskreislauf einer Evolutionsstrategie nach Beyer [6].*

Abbildung 5.1 verdeutlicht den evolutionären Optimierungskreislauf. Zu Beginn der Optimierung werden zunächst μ zufällige Individuen der Spezies erzeugt.

Jedes Individuum k besteht dabei zum einen aus einem u-dimensionalen Vektor der Objektparameter $\vec{o}_k = \{o_1, o_2, \cdots, o_u\}$. Zum anderen besitzt ein Individuum einer Evolutionsstrategie einen v-dimensionalen Vektor $\vec{s}_k$ von sogenannten EA-Parametern.

Während die Objektparameter die eigentlichen Lösungen des betrachteten Problems darstellen, beeinflussen die EA-Parameter die Ausprägung neuer Lösungen. Die Idee, auch die EA-Parameter in den Lösungen zu optimieren, beruht auf der Annahme, dass sich so indirekt auch die Effizienz des Evolutionsprozesses an sich steigern lässt. Zusammen bilden alle Parameter das Genom des Individuums.

Mit wachsender Zahl der Objektparameter und ihrer Wertebereiche kann der betrachtete Lösungsraum (als Menge aller möglicher Lösungen) schnell so groß werden, dass die Evaluation aller Lösungen nicht effizient möglich ist. Die EA-Parameter beeinflussen daher die Auswahl von zu evaluierenden Parametersätzen und damit die sogenannte *Abdeckung* des Lösungsraums.

Die Kodierung der Individuen in Form ihrer Objektparameter hängt sehr stark von dem zu optimierenden Problem ab. In erster Linie unterscheiden sich mögliche Kodierungen in dem verwendeten Alphabet (z.B. binär, ganzzahlig oder reellwertige) an Werten, die ein Parameter des Objektvektors annehmen kann.

Während der Selektion werden alle Individuen einer Population bzgl. der gemeinsamen Umwelt evaluiert, eine Interaktion zwischen den Individuen findet dabei in der Regel nicht statt. Am Ende der Evaluation besitzt jedes Individuum einen eigenen Fitnesswert $F_k = f(\vec{o}_k, \vec{s}_k)$, welcher dessen Lösungsgüte im Bezug auf das Optimierungsproblem ausdrückt.

Bezugnehmend auf die in dieser Arbeit diskutierten Scheduling-Problem werden in den Objektparametern der Individuen beispielsweise bestimmte Strategien bzw. das Verhalten eines Schedulers kodiert und die Fitness dann mit Hilfe von Bewertungsgrößen zur Feststellung der Scheduling-Qualität (vgl. Abschnitt 3.2) ermittelt. Ist das im Individuum kodierte Verhalten des Schedulers bzgl. seiner Umwelt (Eingabedaten, Konfiguration der parallelen Maschinen) also besonders gut (führt z.B. zu einer hohen Auslastung der Maschinen) so besitzt das Individuum auch einen besseren[1] Fitnesswert.

Auf Basis der Fitnesswerte findet dann die Auswahl der Eltern, der nächsten Generation, statt (z.B. durch Auswahl der μ besten Individuen nach Sortierung). Diese Individuen bilden nun die Grundlage für die λ Nachkommen, die im Anschluss durch Reproduktion und Variation der Erbinformationen erzeugt werden.

Die Bildung der gesamten Folgepopulation ist von dem Selektionsoperator „PLUS“ oder „KOMMA“ abhängig. Eine „PLUS“-Strategie führt dabei zu einer Vereinigung von Eltern und Kindern zur Folgegeneration, während die Eltern bei einer „KOMMA“-Strategie nicht Teil der Folgegeneration sind. Die Entscheidung für einen der beiden Operatoren hat Auswirkungen auf das Lernverhalten insbesondere, wenn der Suchraum viele lokale Optima besitzt. Eine „PLUS“-Strategie neigt beispielsweise dazu, schneller gegen ein lokales Optimum zu konvergieren, dieses jedoch nur schwer wieder verlassen zu können. Landet der Algorithmus in der Nähe eines lokalen Optimums wird dieses rasch angenähert, da mit dem verantwortlichen Elternteil auch in den Folgegenerationen ein Ausgangspunkt für die lokale Suche erhalten bleibt. Eine „KOMMA“-Strategie wiederum weist ein größeres *Explorationsverhalten* auf.

Unabhängig von der Wahl des Selektionsoperators findet in dem Schritt Reproduktion & Variation die Erzeugung neuer Lösungen statt. Die Idee ist, dass bei der Fortpflanzung von Individuen, die sich in der Umwelt bewährt haben, positive Eigenschaften an die Nachfolgegeneration weitergegeben werden. Hierbei lässt es eine Evolutionsstrategie offen, ob dies auf Basis eines, zweier oder mehrerer Individuen geschieht. In unserem Fall gehen wir trotzdem nur von einer paarweisen *Rekombination* aus.

Wichtig ist hierbei die Auswahl eines Paares, da die zwei Individuen für jeden erzeugten Nachkommen aus der Menge der μ Eltern gewählt wird. An dieser Stelle kann bereits zusätzliches Expertenwissen einfließen oder — wie häufig üblich — einfach gleichverteilt gewählt werden.

Grundsätzlich existieren für die *Rekombination* zweier Individuen unabhängig von der Kodierung zwei verschiedene Ansätze:

1. Bei der diskreten Rekombination wird für jedes der Gene genau eine Aus-

[1] An dieser Stelle wurde bewusst von einem *besseren* Fitnesswert gesprochen, da es sich bei dem zu optimierenden System häufig um ein Minimierungsproblem handelt und somit die bessere Lösung einen kleineren Fitnesswert besitzt.

prägung der beiden Eltern gewählt. Auf diese Art und Weise besteht das neue Individuum aus disjunkten Teilmengen der Gene seiner Eltern und ist unabhängig von der Art der Kodierung einsetzbar.

2. Die intermediäre Rekombination hingegen eignet sich nicht für binär kodierte Gene, da die neuen Gene eines Individuums hierbei durch die Schwerpunktbildung der Gene der Eltern entstehen —für reellwertige oder ganzzahlige Kodierung beispielsweise durch Bildung des arithmetischen Mittels beider Ausprägungen jedes einzelnen Gens.

Für die Rekombination der Objektparameter empfiehlt Schwefel [109] eine diskrete Rekombination, wohingegen die EA-Parameter intermediär rekombiniert werden.

Ein weiterer Aspekt der Reproduktion und Variation ist die Mutation der Erbinformationen. Ausgangslage der Mutation sind die zuvor rekombinierten λ Nachkommen, die jeweils durch paarweise Rekombination der Vektoren $\vec{o}_k$ und $\vec{s}_k$ entstanden sind.

Wie schon zuvor beim Rekombinationsoperator so ist auch die Wahl eines geeigneten Mutationsoperators abhängig von der Kodierung und dem unterliegenden Problem. Für binäre Objektparameter sieht Beyer [6] eine Mutation mit einer Mutationsrate ω vor, die der Wahrscheinlichkeit zum *Kippen* eines jeden Bits der binären Kodierung entspricht. Bei reeller Kodierung wird der Objektvektor nach Beyer [6] mit Hilfe einer Gaußschen Normalverteilung additiv mutiert.

Formal entsteht der neue Objektvektor $\vec{o}'_k$ also durch $\vec{o}'_k = \vec{o}_k + \vec{z}_k$, wobei die Elemente $\{z_1, z_2, \cdots, z_u\}$ jeweils zufällig über die Normalverteilung mit Erwartungswert 0 und Standardabweichung σ entstehen. In diesem Fall ist σ Teil der EA-Parameter und wird als *Mutationsschrittweite* bezeichnet.

Da im Rahmen einer Evolutionsstrategie auch die EA-Parameter Teil des Evolutionsprozesses sind, werden auch diese nach der Rekombination mutiert und zwar vor der Mutation der Objektparameter. Dies stellt sicher, dass die mutierten Nachkommen innerhalb der Fitness-relevanten Objektparameter implizit auch das Ergebnis der EA-Parameter-Mutation in sich tragen. Diese *Schrittweitenanpassung* erfolgt multiplikativ mit $\tilde{\sigma} = \sigma \cdot \xi$.

Hierbei entspricht ξ einem Mutationsoperator, für dessen Bildung wiederum unterschiedliche Varianten existieren. Eine detaillierte Analyse der Einflussnahme unterschiedlicher Schrittweitenanpassungsverfahren war nicht Teil der Arbeit und so wurde aufgrund seiner einfachen Umsetzung und zum Problem passenden Konfigurationsempfehlung in der Literatur, lediglich ein *Zweipunktoperator* nach Rechenberg [102] angewendet.

Dieser setzt zur Bildung von ξ einen vom Nutzer vorgegebenen (exogenen) Parameter β ein, der —wie in unserem Fall— für kleine Objektvektoren ($u < 100$) nach Kost [69] maximal ($\hat{=}$0,6) gewählt werden soll. Im Anschluss wird ξ vor der

Mutation der Mutationsschrittweite durch Gleichung (5.1) berechnet, wobei r einer gleichverteilten reellen Zufallsvariable zwischen 0 und 1 entspricht.

$$\xi = \begin{cases} 1+\beta, & \text{wenn } r \leq 0\,5 \\ \frac{1}{1+\beta}, & \text{wenn } r > 0\,5 \end{cases} \tag{5.1}$$

Insgesamt umfasst die exogene Konfiguration der Evolutionsstrategie also die Parameter μ, λ und β, wobei nach Schwefel [109] für μ und λ ein Verhältnis von $1:7$ empfohlen wird. Alle anderen Einstellungen (Startwerte für Standardabweichungen, Grenzen für Objektparameter) sind stark vom Optimierungsproblem abhängig und werden daher erst bei der Anwendung in Abschnitt 7.2.4 vorgestellt.

Der Kreislauf aus Selektion mit anschließender Reproduktion und Variation endet erst mit dem Eintreten einer vorgegebenen Abbruchbedingung. Im einfachsten Fall ist dies eine zuvor festgelegte Anzahl von Generationen. Aber auch die Konvergenz gegen eine bestimmte Lösungsqualität kann eine geeignete Abbruchbedingung darstellen.

5.2 Evolutionäre Algorithmen für die mehrkriterielle Optimierung

Generell geht es bei der mehrkriteriellen Optimierung darum, Kompromisslösungen für Probleme zu finden, die aus mindestens zwei oftmals aber auch mehreren gegensätzlichen Kriterien zur Bewertung der Lösungsqualität beruhen.

In diesem Zusammenhang wird der Begriff der *Pareto-Optimalität* von Lösungen genutzt. Dieser ursprünglich auf den Ökonom und Soziologen Vilfredo Pareto zurückgehende Begriff beschreibt genau die Art von Kompromisslösungen, die nicht bezüglich eines der Kriterien verbessert werden können, ohne dass dies gleichzeitig zu einer Verschlechterung mindestens eines anderen Kriteriums führt. Abbildung 5.2 verdeutlicht dieses Prinzip für den Fall eines zweikriteriellen Minimierungsproblems. Auf den Achsen sind hierbei die jeweiligen Kriterien K_1 und K_2 abgetragen. Alle eingezeichneten Lösungen stellen Kompromisse im Bezug auf die beiden Kriterien dar.

Zusätzlich sind drei Lösungen L_1 bis L_3 hervorgehoben, deren Fitnesswerte bezüglich jedes einzelnen Kriteriums durch $K_1(L_x)$ für das erste Kriterium und $K_2(L_x)$ für das zweite Kriterium berechnet werden. Da es sich um ein Minimierungsproblem handelt, ist ein Fitnesswert „besser“, wenn er niedriger ist als ein Vergleichswert. So besitzt L_2 einen besseren Wert für das zweite Kriterium (K_2) als L_3 ($K_2(L_2) < K_2(L_3)$). Für das erste Kriterium gilt allerdings der umgekehrte Fall ($K_1(L_3) < K_1(L_2)$). Da beide Lösungen in jeweils einem Fitnesswert besser sind, kann keine Aussage darüber getroffen werden, welche insgesamt eine bessere Lösung für das Problem darstellt.

Ganz anders hingegen sieht es für die paarweisen Relationen zu L_1 aus. In diesem Fall ist L_1 in allen Kriterien schlechter als die beiden anderen Lösungen L_2 und

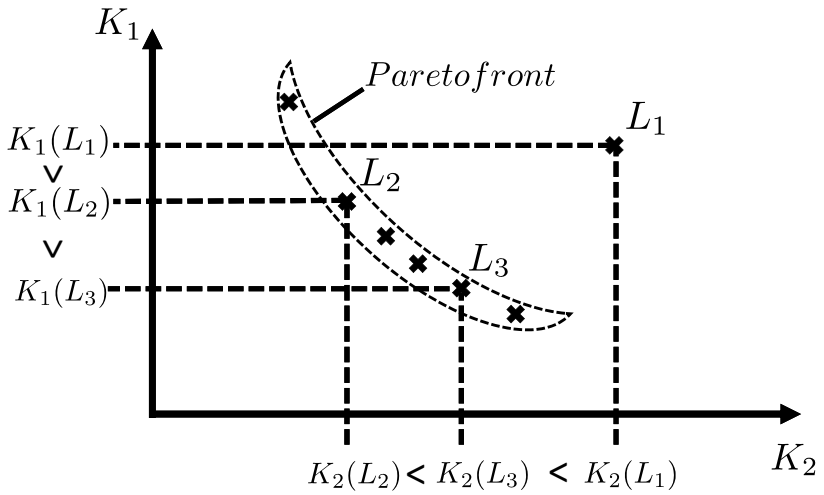


ABBILDUNG 5.2: *Darstellung von Pareto-optimalen Lösungen eines zweikriteriellen Optimierungsproblems.*

L_3. In einer solchen Situation wird auch von der *Dominanz* der Lösungen L_2 und L_3 gegenüber der Lösung L_1 gesprochen[2]. Die Menge aller Lösungen, die nicht dominiert werden (in der Abbildung alle Lösungen außer L_1) wird auch *Paretomenge* genannt und ihre Darstellung im Kriterienraum (Bildraum) *Paretofront*.

Ziel der mehrkriteriellen Optimierung ist es, diese (zuvor unbekannte) Front möglichst vollständig anzunähern[3].

Die *Diversität* bezeichnet in dem Sinne die breite Streuung der gefundenen Lösungen im Lösungsraum, so dass die Lösungsmenge neben Extremlösungen auch Lösungen aus jedem anderen Bereich der Front enthält. Auf die Art und Weise soll ein Algorithmus möglichst alle Pareto-optimalen Kompromisse ermitteln.

Die Generierung der Pareto-optimalen Kompromisse kann dabei auch einkriteriell erfolgen (z.B. mit Hilfe einer Evolutionsstrategie), wenn alle Kriterien zu einem einzelnen Kriterium zusammengefasst werden. Nach Miettinen [86] entspricht eine Aggregation vor der Generierung von Lösungen den sogenannten *A priori-Methoden*. Dem gegenüber stehen *A posteriori-Methoden*, die der Auswahl einer bestimmten Lösung aus der bereits generierten Paretomenge eigentlich „unvergleichbarer“ Lösungen dienen.

Ein sehr einfacher Ansatz, der sich für beide Herangehensweisen eignet ist die Gewichtungsmethode. Diese nutzt für die Aggregation der Kriterien K_1 bis K_n reellwertige Gewichte $w_i \in [0,1]$ von w_1 bis w_n mit der Zusatzbedingung $\sum_{i=1}^{n} w_i = 1$.

Mit Hilfe der Gewichte kann so ein Einzelkriterium K_{Gesamt} berechnet werden durch $K_{Gesamt} = \sum_{i=1}^{n} w_i \cdot K_i$.

[2] Alternativ: L_2 und L_3 *dominieren* L_1.

[3] Man spricht auch von der *Konvergenz* zur Front.

Durch Veränderung der Gewichte der Einzelkriterien w_i lassen sich durch wiederholte Ausführung des gleichen Algorithmus zur Fitnessberechnung sowohl die Extremlösungen als auch die sich dazwischen befindlichen Kompromisslösungen berechnen. Wird die Gewichtung der Kriterien bereits während der Evaluation einer Lösung angewendet, so handelt es sich um eine A priori-Aggregation. Werden die Lösungen zunächst ungewichtet erzeugt und erst im Anschluss anhand variierender Gewichtungen bewertet, so handelt es sich um eine A posteriori-Aggregation.

Da Individuen innerhalb einer Paretomenge per Definition „unvergleichbar“ sind, ist eine direkte Bewertung der enthaltenen Lösungen im Sinne der Selektion wie bei einkriteriellen EAs nicht möglich. Dieses Problem muss von einem mehrkriteriellen EA behandelt werden.

Der zweite wichtige Aspekt ist die Erhaltung der Diversität bei der Generation neuer Populationen, also zu verhindern, dass sich die Populationen in den Extremlösungen der beteiligten Kriterien sammeln.

Das Standardwerk von Coello Coello et al. [16] liefert in diesem Rahmen eine gute Übersicht über Varianten evolutionärer Verfahren zur mehrkriteriellen Optimierung, die unterschiedliche Ansätze zur Lösung der zuvor beschriebenen Anforderungen einsetzen.

An dieser Stelle soll allerdings auf eine umfassende Diskussion der verschiedenen Algorithmen (SPEA, SMS-EMOA und weitere) verzichtet werden.

Stattdessen werden die Aspekte anhand des weit verbreiteten *Non Dominated Sorting Genetic Algorithm* (NSGA-II) [19] erläutert, der im Rahmen der mehrkriteriellen Optimierung in dieser Arbeit eingesetzt wurde. Ein expliziter Vergleich mit anderen Algorithmen war dabei nicht Teil der Arbeit, da die erzielten Lösungen lediglich der Orientierung und zur Einschätzung der Lösungsqualität von Online-Optimierungsverfahren und erprobten Heuristiken dienen sollten.

Bei NSGA-II handelt es sich um ein A posteriori-Verfahren. Gleichzeitig stellt es den Nachfolger eines mehrkriteriellen Optimierungsverfahrens dar, welches eine gezielte Selektion auf Basis der Pareto-Dominanz von Lösungen durchführt. Die Verbesserungen gegenüber der Vorgängerversion beziehen sich im wesentlichen auf drei Punkte:

1. Optimierung der Laufzeit für große Populationsgrößen.
2. Verstärkung des Elitismus, also des Erhalts von vorteilhaften Lösungen (ähnlich der „PLUS“-Strategie bei ES).
3. Vereinfachung in der Nutzung durch automatische Anpassung eines vormals exogenen Strategieparameters.

Der Kern des Algorithmus ist der spezielle Ablauf der Selektion (Auswahl von Individuen zur Reproduktion). Dieser findet bei NSGA-II gleichzeitig über das Pareto-Dominanz-Kriterium als auch über eine spezielle Distanzfunktion statt, welche die

Individuen bevorzugen soll, die „relativ allein“ innerhalb eines Suchraumbereichs liegen. Die Selektion wird dabei stufenweise durchgeführt.

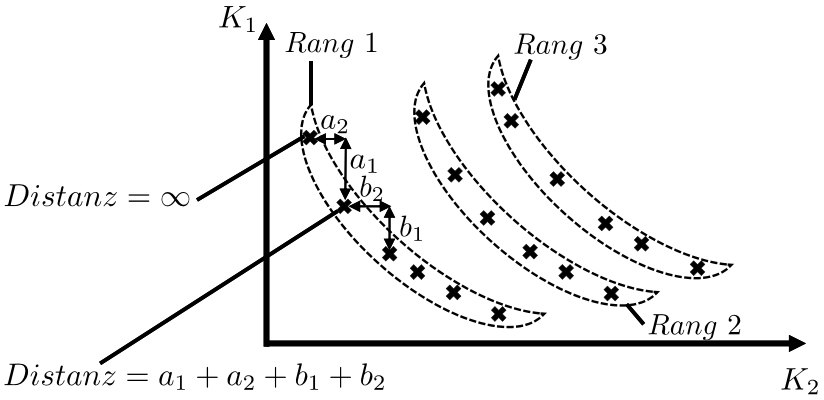


ABBILDUNG 5.3: *Bildung von Rängen und Berechnung der Distanzfunktion anhand eines zweikriteriellen Optimierungsproblems.*

Wie in Abbildung 5.3 beispielhaft dargestellt, werden alle Lösungen zunächst in sogenannte Ränge unterteilt, wobei der niedrigste Rang aus den dominierenden Lösungen gebildet wird. Durch ein in NSGA-II gegenüber „Version I“ verbessertem Verfahren werden nun stufenweise weitere Ränge gebildet, wobei die Lösungen jedes berechneten Ranges aus der verbleibenden Menge von Lösungen entfernt wird. Zusätzlich wird für jedes Individuum noch eine *Crowding Distance* berechnet, welche der Summe normalisierter paarweiser Abstandsberechnungen zwischen benachbarten Individuen über alle Einzelkriterien entspricht. Die Berechnungspaare werden dabei lediglich innerhalb der Lösungen desselben Ranges gebildet. Da Extremlösungen nur einen Nachbarn innerhalb des Ranges haben können[4], ist das Distanzmaß in dem Fall per Definition unendlicher Ausprägung (vgl. Abbildung 5.3).

Auf Basis dieser beiden Selektionsoperatoren wird auf allen Lösungen einer evaluierten Population inklusive der Eltern eine partielle Ordnung definiert, in der zunächst nach Rang und dann nach Distanzmaß sortiert wird, wobei eine größere Distanz zwischen benachbarten Elementen vorteilhafter ist.

Nach dieser Ordnung wird zunächst eine Teilmenge von Individuen bestimmt, die überhaupt erst an der Rekombination beteiligt werden sollen. Innerhalb dieser Teilmenge werden zur Bestimmung der fortpflanzenden Elternpaare jeweils zwei Individuen gezogen, wobei dann immer jenes genommen wird, welches in der Ordnung weiter vorne steht. Auf die Art und Weise besitzen Individuen mit niedrigerem Rang eine höhere Wahrscheinlichkeit, sich zu reproduzieren, als welche mit höherem Rang. Während die Rangbildung die Konvergenz zur Pareto-Front fördert, führt das zweite Sortierkriterium (Crowding Distance) innerhalb der Ränge zu einer Erhöhung der Diversität. Damit sind beide Anforderungen an einen mehrkriteriellen EA erfüllt.

[4]Dies gilt mindestens bzgl. eines Kriteriums.

Die genutzten Operatoren für die eigentliche Rekombination und Mutation nach Selektion sind problemabhängig und werden daher erst in Abschnitt 11.2.1 näher beschrieben.

5.3 Reinforcement Learning

Im Gegensatz zu den beiden vorherigen Offline-Optimierungsverfahren handelt es sich bei dem *Reinforcement Learning*[5] (RL) um ein Online-Lernverfahren. Nach Sutton & Barto [120] versucht ein *Agent* hierbei durch Interaktion mit seiner Umwelt — und in der Regel ohne Vorwissen, durch Ausprobieren — zu ermitteln, unter welchen Zuständen (engl. *States*) seiner Umwelt welche Aktionen (engl. *Actions*) am vorteilhaftesten für ihn sind. Dabei versucht er auf lange Sicht eine Strategie (engl. *Policy*) anzunähern, die ihm einen maximalen Gewinn (engl. *Gain*) beschert.

Da er — für ein Onlineproblem typisch — die Belohnung (engl. *Reward*) von Aktionen unter einer bestimmten Situation lediglich schätzen kann, spricht man auch von zeitlich verzögerter Belohnung.

Diese beiden Aspekte (Ausprobieren von Aktionen und verzögerte Belohnung) sind nach Sutton & Barto [120] charakteristisch für das Reinforcement Learning. Als Unterart des *Maschinellen Lernens* unterscheidet es sich von Verfahren wie dem *Überwachten Lernen* (engl. *Supervised Learning*) dadurch, dass es nicht um die Klassifikation von bereits bekannten Lösungen geht, sondern vielmehr um die Förderung positiven Verhaltens in einer unbekannten Umwelt.

Formal gesehen wiederholt sich beim Reinforcement Learning der in Abbildung 5.4 dargestellte Kreislauf, in dem ein Agent ausgehend von einem Zustandssignal s_t und einem Belohnungssignal r_t, welches er aus seiner Umwelt empfängt, sein Verhalten anpassen und über eine Aktion a_t Einfluss auf die Umwelt nehmen kann. Dies führt — unter Umständen in Verbund mit anderen Einflüssen — zu einer Überführung der Umwelt in den Zustand s_{t+1} sowie einem Belohnungssignal r_{t+1}, welches mit der vorherigen Wahl von a_t gekoppelt ist.

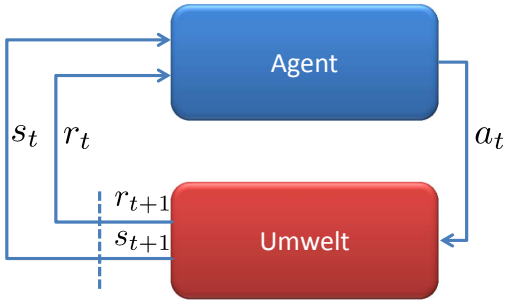


Abbildung 5.4: *Kreislauf aus Interaktion und Perzeption eines RL-Agenten mit seiner Umwelt nach Sutton & Barto [120].*

[5] In der deutschen Literatur auch manchmal mit *Bestärkendem Lernen* übersetzt.

Ein einzelner Durchlauf wird im Falle des RL nicht Generation sondern *Schritt* (engl. *Step*) genannt. Eine Kette von Schritten $t = 0 \rightarrow t = 1 \rightarrow \cdots \rightarrow t = T$ bezeichnet man als Episode. Eine Episode kann dabei unendlich lang ($T = \infty$) sein oder aus einer endlichen Kette von Schritten bestehen. Konsequenterweise werden Probleme, die in Episoden aufteilbar sind (mehrere Runden ein und desselben Spiels) auch als *episodisch* bezeichnet, während Probleme mit unendlichen Episoden als *kontinuierlich* bezeichnet werden.

Innerhalb des Reinforcement Learning existieren unterschiedliche Klassen von Algorithmen. Das *Dynamic Programming* beispielsweise setzt ein bereits vorhandenes Modell der Umgebung voraus — inklusive Zustandsübergängen in Abhängigkeit von gewählten Aktionen und Belohnungen für die Übergänge.

Ausgehend von diesem vordefinierten Modell finden Algorithmen der Dynamischen Programmierung dann durch iterative Annäherung einer sogenannten *Optimal Value Function* für jeden Zustand, in dem sich das Problem befinden kann, heraus, welche Aktion als nächstes gewählt werden sollte. Algorithmen dieser Art eignen sich aufgrund der hohen Anforderungen an das Modell der Umwelt nicht für unser paralleles Online-Scheduling-Problem, in dem Zustandsübergänge beispielsweise von dem über die Zeit eingehenden Workload abhängig sind und sich das Modell demnach erst über die Zeit aufbaut.

Ähnlich verhält es sich bei den von Sutton & Barto [120] beschriebenen *Monte Carlo Methoden*. Diese setzen zwar kein umfassend definiertes Modell voraus, arbeiten jedoch auf episodischen Problemen, da einzelne Zustands-/Aktionspaare stets als Teil einer Sequenz von Schritten bzw. ganzen Episoden bewertet werden. Ein kontinuierliches Problem, in dem bereits während des Lernens Entscheidungen getroffen werden, lässt sich auf diese Weise nicht lösen.

Das sogenannte *Temporal Difference Learning* ist für kontinuierliche Probleme hingegen eher geeignet. Hierbei führt ein Agent stets eine einzelne Aktion aus und passt nach Empfang des Zustands- und Belohnungssignals sein Verhalten an. Die Wahl der Aktion hängt dabei von dem gesammelten Wissen der vergangenen Schritte ab und ist daher ohne Vorwissen zu Beginn noch explorativ (erkundend). Mit einer zunehmenden Anzahl von Schritten vergrößert sich das Wissen des Agenten und die Entscheidungen passen sich (auch dynamischen Änderungen) der Umwelt an.

Zwei bekannte Vertreter des Temporal Difference Learning sind *SARSA* [120] und das ursprünglich von Watkins [126] eingeführte *Q-Learning*, die im Ablauf sehr ähnlich sind. Sie unterscheiden sich allerdings in dem Verhältnis aus Exploration — dem Austesten von unter Umständen wenig erfolgversprechenden Aktionen — und der Ausnutzung von erlangtem Wissen, wobei Q-Learning stärker auf letzteres fokussiert ist.

In dieser Arbeit wird an späterer Stelle (vgl. Abschnitt 11.3) ein reaktives System konzipiert, welches auf dynamische Änderungen der Umwelt reagieren soll und daher eher auf die Ausnutzung erarbeiteten Wissens fokussiert ist. Daher wird

im weiteren Verlauf der Arbeit lediglich das besser passende Q-Learning als Reinforcement Learning Algorithmus genutzt[6].

Beim Q-Learning ordnet die namensgebende *action-value*-Funktion $Q(s, a)$ jedem Zustands-/Aktionspaar (s, a) mit $s \in \mathbb{S}$ und $a \in \mathbb{A}$ einen — nach bisherigem Kenntnisstand — erwarteten Nutzen zu. Zentraler Bestandteil des Q-Learning ist die in 5.2 gegebene *Update*-Regel, welche für die Anpassung der Q-Funktion und somit der Anpassung des vorhandenen Wissens sorgt.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \underbrace{\gamma \max_{a \in \mathbb{A}} Q(s_{t+1}, a)}_{\text{Maximal erwarteter Nutzen}} - Q(s_t, a_t)\Big] \tag{5.2}$$

Ausgehend von dem in Abbildung 5.4 gezeigten Kreislauf der Interaktion eines Agenten mit seiner Umwelt, findet ein solcher Aktualisierungsschritt nach Ausführung einer Aktion a_t und Aufnahme der Signale für den neuen Zustand s_{t+1} und die unmittelbare Belohnung r_{t+1} statt. Hierbei wird der Q-Wert der zum vorherigen Zeitpunkt t gewählten Aktion a_t unter dem Zustand s_t additiv angepasst.

Der *Schrittweite*-Parameter $\alpha \in \Re^+$ bestimmt die Lerngeschwindigkeit des Algorithmus. Je größer der Faktor ist, desto stärker sind die Veränderungen eines Q-Werts, aber desto reaktiver ist das Verhalten des Algorithmus, da momentane Belohnungen stärker gewichtet werden als die im alten Q-Wert enthaltenen vergangenen Belohnungen. Sutton & Barto [120] empfehlen eine Konfiguration von $\alpha = 0{,}1$ für die meisten Probleme.

Bei r_{t+1} handelt es sich um das zuvor angesprochene Belohnungssignal, welches aus der Aktion a_t auf Basis des Zustands s_t hervorgeht. Die Modellierung dieses reellwertigen Belohnungssignals ist eine der Hauptaufgaben bei der Anpassung des Lernalgorithmus an das zugrundeliegende Optimierungsproblem.

Gleichzeitig berücksichtigt das Q-Learning im Aktualisierungsschritt auch den maximal zu erwartenden Nutzen, indem dieser über den *Discount-rate*-Parameter $0 \leq \gamma \leq 1$ gewichtete maximale Q-Wert aller Aktionen $a \in \mathbb{A}$ eingeht, die auf Basis des neuen Zustandes s_{t+1} vorhanden sind. Diese Berücksichtigung fußt auf der Annahme, dass die Wahl der Aktion a_t direkten Einfluss auf den Zustandswechsel $s_t \rightarrow s_{t+1}$ gehabt hat und der Q-Wert der Aktion daher auch von dem Nutzen dieser Transition abhängen sollte. Wird γ klein gewählt, so steigt der Einfluss des unmittelbaren Belohnungssignals. Mit einem großen γ hingegen steigt der Einfluss der Zustandswechsel.

Da die reellwertigen Q-Werte lediglich den zu erwartenden Nutzen ausdrücken, wird beim Q-Learning zusätzlich noch eine Strategie zur Aktionswahl benötigt, die ausgehend von den Q-Werten aller Aktionen für einen bestimmten Zustand s_t genau eine Aktion bestimmt, die als nächstes vom Agenten ausgeführt wird. Diese Verfahren dienen üblicherweise dazu, das Verhältnis aus Exploration und Ausnutzung des Aktionsraums zu steuern. Die einfachste Form ist dabei das sogenannte

[6] Da es auch hier wieder viele unterschiedliche Formen des Q-Learning gibt, sei der Vollständigkeit halber darauf hingewiesen, dass alle Ausführung sich auf das *1-Step Q-Learning* beziehen.

ϵ-*Greedy*-Verfahren. ϵ steht in diesem Fall für die gleichverteilte Wahrscheinlichkeit, dass nicht die Aktion mit dem größten Q-Wert zur Ausführung gewählt wird, sondern ebenfalls gleichverteilt eine zufällige Aktion aus dem gesamten Aktionsraum $\mathbb{A}$ bestimmt wird. Mit $\epsilon = 0$ besitzt das Q-Learning demnach ein deterministisches Verhalten, während ein größeres ϵ zu einem nichtdeterministischen Verhalten mit explorativem Lernen führt. Abbildung 5.5 zeigt den kompletten Ablauf eines 1-

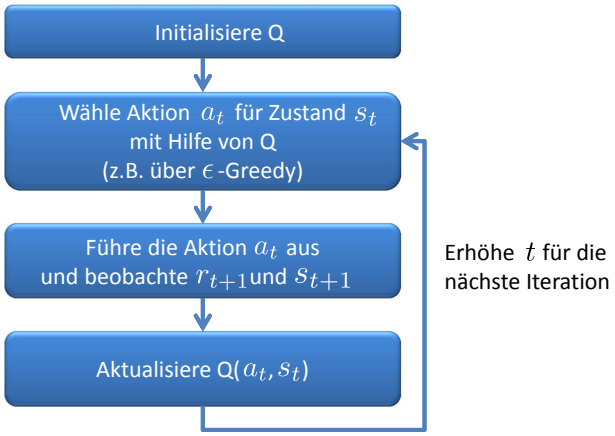


Abbildung 5.5: *1-Step Q-Learning für kontinuierliche Probleme nach Sutton & Barto [120].*

Step Q-Learnings für kontinuierliche Probleme. Die detaillierte Konfiguration des Zustandsraum $\mathbb{S}$, des Aktionsraums $\mathbb{A}$, sowie der Lernparameter ϵ, α und γ als auch die Berechnung des Belohnungssignals sind Teil der Anpassung an das spezifische Lernproblem und werden daher an entsprechender Stelle in Abschnitt 11.3 diskutiert.

Teil II

Ressourcenerweiterung mit Computational Grids

Kapitel 6

Grundlagen von Computational Grids

Wie bereits in Kapitel 2.3 erwähnt, steht bei einem Computational Grid der wechselseitige Austausch von Rechenkapazitäten im Vordergrund. Dieser kann grundsätzlich auf zwei verschiedene Arten vonstatten gehen. Bei Ersterer handelt es sich um die Delegation von Aktivitäten (AD). Dies entspricht im einfachsten Fall dem klassischen *load balancing* nach Foster & Kesselman [51], bei dem zu berechnende Arbeit — in einer nicht näher spezifizierten Art und Weise — immer der am wenigsten ausgelasteten Grid-Site des Verbundes zugewiesen wird.

Jedoch ist genau die Art und Weise der Verteilung von Arbeitslast ein Forschungsfeld, welches sehr vielfältig ist und für das bereits auf mehrere Jahrzehnte intensiver Forschungen verwiesen werden kann. Dies betrifft sowohl die technische Umsetzung des Zusammenschlusses von Grid-Sites als auch algorithmische Forschungen für die Steuerung sowie Optimierung des Lastaustauschs zwischen den Sites. Beide Aspekte werden bzgl. des aktuellen Forschungsstands und den eigenen Ansätzen im Rahmen dieser Delegationsart in Kapitel 7 näher diskutiert.

Bei der zweiten Art des wechselseitigen Austausches von Rechenkapazitäten handelt es sich um die Delegation von Ressourcen (RD). Zwar werden hierbei am Ende ebenfalls Rechenaufgaben auf die Gesamtinfrastruktur des Grids verteilt, jedoch geht dies anders vonstatten und hat damit auch Auswirkungen auf die dafür zu entwickelnden technologischen sowie algorithmischen Umsetzungen. Auch wenn dieser Ansatz — da recht innovativ — nicht auf eine langjährige Grundlagenforschung zurückblicken kann, schlägt er dennoch eine Brücke zwischen dem etablierten Grid Computing und dem aktuellen *Scientific Cloud Computing* (vgl. Teil III). Daher werden in Kapitel 8 die eigenen Entwicklungen im Rahmen dieser Delegationsart ebenfalls vorgestellt.

Grundsätzlich existieren für die Grid-Scheduling-Architektur zwei verschiedene Ansätze und eine Menge von Mischformen, die aus Kombination der beiden Ansätze entstehen können. Im folgenden Abschnitt werden diese zwei grundlegenden

Ausprägungen diskutiert und die im Weiteren angenommene Architektur vorgestellt.

6.1 Betrachtete Grid-Scheduling-Architektur

Die am weitesten verbreitete Architektur eines Computational Grids ist die hierarchische bzw. zentrale Grid-Scheduling-Architektur. Sie zeichnet sich — wie in Abbildung 6.1 dargestellt — dadurch aus, dass für die Nutzer innerhalb einer Grid-Community unabhängig ihres physischen Standorts und ihrer Zugehörigkeit zu einem der Ressourcenbetreiber eine einzelne Einreichschnittstelle zur Verfügung steht. Dieser *Meta-Scheduler* vereint alle technischen und algorithmischen Aspekte

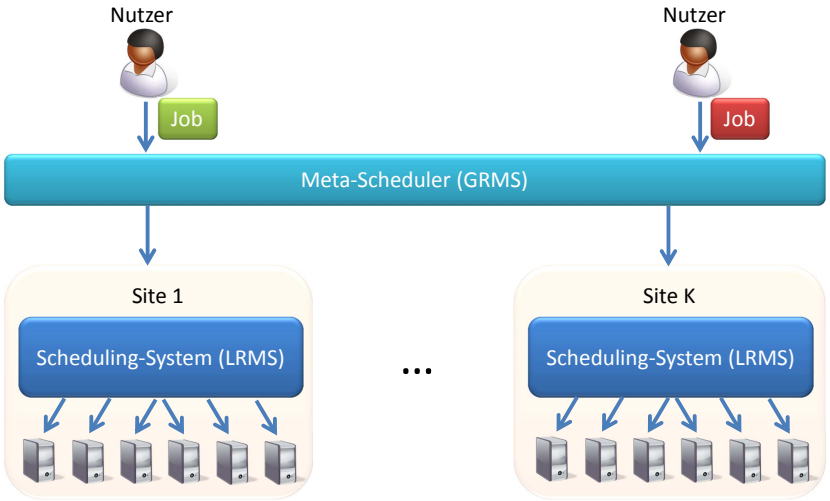


Abbildung 6.1: *Hierarchisches Grid-Scheduling mit zentralem Meta-Scheduler.*

auf Grid-Ebene wie z.B.

- ... das Auffinden und die Überwachung von Ressourcen. Bei diesem auch *resource discovery and monitoring* genannten Prozess erlangt der Meta-Scheduler eine Art globale Sicht über die unterliegenden Ressourcen, so dass er stets weiß, welche Ressourcen innerhalb des Grids zur Verfügung stehen (z.B. durch An- und Abmelden von einzelnen Ressourcen bzw. Eintritt/Austritt gesamter Betreiber in das oder aus dem Grid).

  Dazu wird üblicherweise auch direkt auf die unterliegenden Informationssysteme der einzelnen LRMS/Batch-Systeme zugegriffen. Das daraus konsolidierte Modell dient dem Meta-Scheduler daraufhin als Grundlage für seine getroffenen Scheduling-Entscheidungen. Zusätzlich werden auch alle sich im Grid befindlichen Jobs überwacht, um dem Nutzer entsprechend Rückmeldung über den Status seiner Anwendungen geben zu können.

- ... die Authentifizierung und Autorisierung. Als zentrale Einreichschnittstelle verifiziert der Meta-Scheduler sowohl die Identität der Nutzer als auch die der Ressourcenbetreiber. Dies kann beispielsweise über Zertifikate geschehen, die allen Parteien ausgestellt werden. Ausgehend von der sichergestellten Identität der Parteien kann darüber hinaus auch ein gewisses Rechte-Management realisiert werden. Dies umfasst beispielsweise das Filtern der unterliegenden Ressourcen nach dem Kriterium, ob der entsprechende Nutzer die Berechtigung zur Ausführung seiner Anwendung bzw. seines Rechenjobs auf einer Ressource besitzt.

- ... das Site-übergreifende Datenmanagement. Zu jeder im Grid gestarteten Anwendung gehören unter Umständen auch Ein- und Ausgabedaten, die verarbeitet bzw. berechnet werden müssen. Da systembedingt zum Zeitpunkt des Einreichens noch nicht bekannt ist, auf welcher Ressource eine Anwendung tatsächlich ausgeführt wird, muss ein Meta-Scheduler ebenfalls über ein Datenmanagementsystem verfügen bzw. ein solches angebunden sein, damit ausgehend von den Scheduling-Entscheidungen entsprechend Datentransfers vorgenommen werden können.

  Diese unterteilen sich in das *stage-in* als die Übertragung der Eingabedaten zum Ausführungsort der Anwendung und das *stage-out* als analogen Rücktransfer der Ausgabedaten. Heutige Datenmanagementsysteme unterstützen darüber hinaus auch noch Mechanismen zur Ausfallsicherheit wie z.B. die Replikation von Daten.

- ... Heuristiken bzw. Scheduling-Strategien für die dynamische Verteilung der Rechenjobs auf die Ressourcen gemäß des — zuvor beschriebenen — globalen Modells.

Nahezu jede heutige Grid-Installation beruht auf diesem Konzept, und nutzt für die Umsetzung ihrer Grid-Infrastruktur eine der sogenannten *Grid-Middlewares*. Die zwei bekanntesten — und derzeit noch in der Weiterentwicklung befindlichen — Vertreter sind das Globus Toolkit (GT) [50] und UNICORE [30, 118]. Beide bilden die oben genannten Aspekte auf unterschiedliche technische Standards ab und sind daher — obwohl sie nahezu die gleiche Funktionalität bieten — nicht miteinander kompatibel.

Neben der globalen Sicht auf das System geht der zentrale Ansatz ebenfalls von einer alleinigen Kontrolle des Meta-Schedulers über die unterliegenden Sites aus. Im Falle einer Zuweisung von Arbeitslast zu einer der Sites ist diese demnach gezwungen die Arbeit zu übernehmen. Der gewollte Austausch von Arbeitslast zwischen den einzelnen Sites findet vorgelagert innerhalb des Meta-Schedulers und ohne Einflussmöglichkeit der Ressourcenbetreiber statt. Eine Kommunikation zwischen den einzelnen Sites ist dabei nicht vorgesehen.

Auch wenn ein solches zentral verwaltetes Grid unter Umständen aus mehreren Sites mit einzelnen LRMS besteht, so unterscheidet es sich konzeptionell aufgrund

der Informationspolitik und der Kontrolle über die angebundenen Ressourcen wenig von einem einzelnen MPP-System (vgl. Abschnitt 2.1 auf S. 10). Innerhalb eines MPP-Systems setzt der Ressourcenbetreiber ein LRMS ein, welches die Zuweisung von Jobs zu einzelnen Maschinen vornimmt. Innerhalb eines Grids, welches von einer virtuellen Organisation (aus Ressourcenbetreibern und Nutzern) verwaltet wird, werden die Jobs von einem GRMS (Grid Ressourcen Management System) auf die einzelnen Sites verteilt. Somit lassen sich die algorithmischen Ansätze aus dem lokalen Scheduling bei Verzicht auf die „Site-Autonomie“ ebenfalls auf ein zentral verwaltetes Grid übertragen.

Eine größere Herausforderung hinsichtlich der eingesetzten Technologie und dem zu lösenden Scheduling-Problem ergibt sich jedoch, wenn die Kontrolle über unterschiedliche Scheduling-Domänen hinweg verteilt wird. Dies mündet in einer dezentralen Grid-Scheduling-Architektur, wie in Abbildung 6.2 dargestellt.

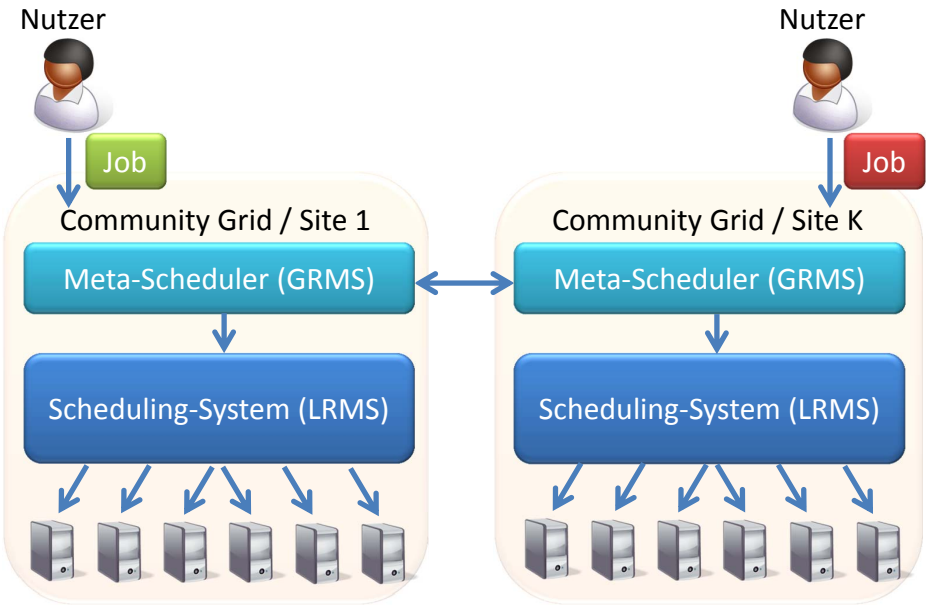


Abbildung 6.2: *Dezentrales Grid-Scheduling mit autonomen Meta-Schedulern/Sites.*

Hierbei verfügt nun jede einzelne Partei über eine eigene Einreichschnittstelle für Jobs ihrer Nutzer. Aufgrund der zuvor angesprochenen Parallelen wird dabei bewusst offengelassen, ob es sich bei den gleichberechtigten Partnern des dezentralen Grids um einzelne Sites unter Kontrolle eines einzelnen Ressourcenbetreibers oder um Meta-Scheduler unter Kontrolle einer virtuellen Organisation handelt.

Im dezentralen Grid kommen neue Aspekte hinzu, die in einer einzelnen Scheduling-Domäne noch zweitrangig gewesen sind. Da es sich bei allen Parteien um weitgehend unabhängige Akteure handelt, ist anzunehmen, dass sich diese auch unterschiedlich entwickelt haben. Dies reicht von der Wahl und Entwicklung einer

Grid-Middleware oder der Beschreibungssprache von Jobs[1] bis zur Installation unterschiedlicher Anwendungen und Protokolle. Um die bestehende Autonomie der Sites auch bei der Teilnahme an einem Grid zu erhalten, wird der Lastaustausch zwischen den Parteien üblicherweise in Form von Verhandlungen durchgeführt.

Neben der technischen Umsetzung zeichnet sich auch das Scheduling-Problem durch eine höhere Komplexität aus, da die einzelnen Parteien des Grids nun kooperativ Arbeitslast mit dem Ziel austauschen wollen, um die Scheduling-Qualität zu verbessern und trotzdem ihre Autonomie über die lokalen Ressourcen behalten.

Diese Arbeit wird zeigen, dass sich eine solche Kooperation im dezentralen Grid als vorteilhaft für alle Parteien erweist, auch wenn sie unter sehr restriktiven Modellannahmen hinsichtlich des Informationsaustauschs und unter der vollständigen Wahrung der Site-Autonomie in Form einer dezentralen Kooperation stattfindet. Im Zuge dessen werden einfache Heuristiken diskutiert und zusätzlich Methoden vorgestellt, Strategien für eine erfolgreiche Durchführung des Arbeitslastaustauschs mit Hilfe von maschinellen Lernverfahren zu optimieren.

6.2 Technische Umsetzung eines dezentralen Computational Grids am Beispiel von DGSI

Neben der Implementierung und Evaluation neuartiger Lastaustauschverfahren war der Autor dieser Arbeit ebenfalls an der Standardisierung und Umsetzung technischer Verfahren im Rahmen eines Drittmittelprojekts beteiligt.

Das vom Bundesministerium für Bildung und Forschung (BMBF) geförderte DGSI[2]-Forschungsprojekt [8] hat sich zum Ziel gesetzt, eine verhandlungsorientierte Interoperabilitätsschicht für unterschiedliche Meta-Scheduler der deutschen Grid-Initiative (D-Grid[3]) zu erarbeiten und prototypisch umzusetzen. Mit einer vereinheitlichten Architektur konnten die technischen Problemstellungen der Aktivitäten- und Ressourcendelegation im Sinne eines dezentralen Computational Grids mit vollständiger Site-Autonomie gelöst werden.

Dabei wird die Interoperabilität der Meta-Scheduler auf oberster Ebene durch die Verhandlung von elektronischen Verträgen zur Dienstnutzung (engl. *Service Level Agreement* (SLA) [7]) erreicht. Die technische Umsetzung der Verhandlung und Überwachung der Verträge wird mit Hilfe des WSAG4J[4]-Frameworks vorgenommen.

Die darunterliegende Ausführungsebene realisiert den Transfer erfolgreich verhandelter Aktivitäten zwischen den Vertragspartnern inklusive dem Staging von

[1] Die Beschreibung von Rechenjobs enthält beispielsweise Informationen über die Ausführungsdatei, Eingabedaten, die benötigte Ausführungsumgebung und vieles mehr.

[2] D-Grid Scheduler Interoperability.

[3] http://www.d-grid.de/, Zugriff: 16. Januar 2013.

[4] http://wsag4j.sourceforge.net/site/, Zugriff: 16. Januar 2013.

Ein- und Ausgabedaten sowie der Überwachung der eigentlichen Ausführung auf Basis etablierter Standards.

Für die temporäre Bereitstellung von Ressourcen werden auf Ausführungsebene sogenannte virtuelle Frontends (VFE) bereitgestellt. Dabei handelt es sich im Kern um voll funktionsfähige virtualisierte Grid-Middleware-Endpunkte (GM) wie beispielsweise Globus-GRAM[5] oder UNICORE-Gateway (vgl. Kapitel 6.1). Dadurch kann ein verleihender Meta-Scheduler jedem Verhandlungspartner eine kompatible Einreichschnittstelle für den Zugriff auf die verliehenen Ressourcen zur Verfügung stellen.

Neben der Auswahl geeigneter Technologien und Frameworks für eine standardisierbare Umsetzung der Architektur, wurden ebenfalls die Spezifizierung von Delegationsprotokollen sowie die Anbindung von fünf verschiedenen Meta-Schedulern vorgenommen.

In den anschließenden paarweisen „Kreuztests“ konnte die Funktionalität der Interoperabilitätsschicht nachgewiesen werden. Daraufhin wurde sie auf der abschließenden D-Grid Ergebniskonferenz im April 2012 der breiten Öffentlichkeit vorgeführt. Die detaillierte Dokumentation der Architektur sowie aller Protokolle, Services und verwendeter Standards zur Realisierung der Interoperabilitätsschicht wurde in Form von Lieferobjekten (engl.: *Deliverables*)[6] vorgenommen.

[5] *Grid Resource and Allocation Manager.*

[6] http://forge.it.irf.tu-dortmund.de/projects/dgsi/publications/deliverables.html, Zugriff: 22. Januar 2013.

Kapitel 7

Aktivitätendelegation in dezentralen Computational Grids

Der Begriff einer Aktivität kapselt jedwede Form von einfachen und komplexen Rechenaufgaben. So kann eine Aktivität entweder einem sequentiellen Rechenjob entsprechen oder einen parallelen Job bzw. einen gesamten Workflow beschreiben. Auf Ebene der Interoperabilität und den damit verbundenen Teilproblemen (resource discovery, staging vgl. Kapitel 6.1) existiert kein Unterschied, ob ein einzelner Job oder ein ganzer Workflow delegiert wird, der wiederum aus voneinander abhängigen Jobs besteht. Daher werden Aktivitäten im Folgenden stets als einzelne parallele Jobs betrachtet.

7.1 Existierende algorithmische Ansätze für die Aktivitätendelegation

Aufgrund der Fokussierung der eigenen Arbeit auf das dezentrale Grid-Scheduling-Modell liegt auch bei der Aufarbeitung des algorithmischen Hintergrundes der Schwerpunkt auf Verfahren, die für dezentrales Grid-Scheduling konzipiert sind. Auch wenn diese wissenschaftliche Nische innerhalb der Aktivitätendelegation sich mehr oder weniger erst in den letzten 15 Jahren etabliert hat, kann sie bereits auf viele unterschiedliche Ansätze verweisen, wobei nicht immer eindeutig ersichtlich ist, ob es sich wirklich um einen dezentralen Ansatz handelt.

So haben beispielsweise Suri & Singh [119] einen Ansatz entwickelt, bei dem unabhängige *Coordinator Nodes (CN)*, welche vergleichbar mit den Meta-Schedulern in unserem Modell sind, die jeweils lokal eingereichte Gesamtlast unter allen Sites ausbalancieren.

Dies geschieht durch die Kommunikation der momentan gemessenen Speicher-

und CPU-Auslastung innerhalb der Site[1], die jede CN in regelmäßigen Abständen an eine zentrale Komponente sendet.

Wenn nun bei Eingabe eines neuen Jobs bei der jeweiligen CN ein zuvor festgelegter Grenzwert hinsichtlich der Lastgrößen überschritten wird, so fragt die CN bei der zentralen Komponente eine Liste von bekannten CNs mit niedrigem Lastlevel an und wählt nach einem in der Veröffentlichung beschriebenen Algorithmus eine davon für die Abgabe des Jobs aus.

Bei diesem Ansatz sind vor allem zwei Aspekte kritisch. Zum einen werden die Entscheidungen zwar dezentral in jeder CN getroffen, dennoch führt die Datenhaltung innerhalb der zentralen Komponente dazu, dass das ganze System im Grunde genommen kein dezentrales Scheduling-System mehr ist.

Zum anderen muss jede der CNs all ihre Lastinformationen veröffentlichen, damit das System funktioniert und darf die Annahme eines fremden Jobs bei Unterlast zudem nicht verweigern.

Einen ähnlichen Weg — allerdings ohne zentrale Komponente — gehen England & Weissman [26]. In ihrem Ansatz entspricht der Lastindex einer Site entweder der Queue-Länge (Anzahl der Jobs in der Queue) oder der sogenannten *gewichteten Queue-Länge*, welche die Summe der Parallelität aller wartenden Jobs in Verhältnis zur lokalen Maschinengröße setzt und somit ein genaueres Abbild der lokal wartenden Last liefert[2].

In ihrem Modell sehen die Autoren vor, dass die Lastindizes aller Delegationspartner für jeden lokal eingereichten Job gesammelt werden, sofern die lokale Queue nicht leer ist. Im Anschluss wird der Job einfach an die Site delegiert, welche den geringsten Lastindex besitzt. Auf Basis synthetischer Workloads, die durch das Abtasten realer Aufzeichnungsdaten entstanden sind (vgl. Abschnitt 4.1 auf S. 27), weisen die Autoren die Vorteilhaftigkeit von Lastaustausch zwischen den Sites im Hinblick auf den durchschnittlichen *Showdown*[3] aller Jobs im Gesamtsystem nach.

Trotz Verzicht auf die zentrale Informationshaltung kann das jobgebundene *Polling* von Lastinformationen einerseits zu einer großen Anzahl expliziter Nachrichten führen. Andererseits müssen die Sites auch in diesem Model alle Fremdjobs annehmen.

Zumindest für den ersten Kritikpunkt liefern Lu et al. [77] Abhilfe. Sie nutzen einen Peer-basierten Ansatz von Sites, welche bei wachsender Site-Anzahl in dynamische Nachbarschaftsgruppen unterteilt werden. Jede der Sites berechnet zur Laufzeit einen lokalen Lastindex, welcher auf der kumulativen erwarteten Lauf-

[1] Die genaue Berechnung dieser Größen erfolgt zum Teil über eine einfache (geometrische oder arithmetische) Mittlung aus Erhebungen der Einzelgrößen angebundener Rechner der Domäne.

[2] Diese Herangehensweise wird später auch im Rahmen der eigenen Lösung (vgl. Abschnitt 7.2.1) verwendet.

[3] Der Slowdown (SD_j) eines Jobs j bezeichnet das Verhältnis aus der Antwortzeit des Jobs und seiner Laufzeit ($SD_j = \frac{\mathrm{RT}_j}{p_j}$). Je kleiner dieser gegen 1 strebende Wert wird, desto effizienter werden die Jobs im System abgearbeitet.

zeit aller momentan ausgeführten und wartenden Jobs basiert, dabei jedoch nicht berücksichtigt, dass einige Sites eventuell größere Rechenkapazitäten besitzen als andere.

Diese Informationen werden nun weder — wie im vorherigen Ansatz — in einer zentralen Komponente verwaltet noch werden sie explizit über das Netz ausgetauscht. Stattdessen werden sie gemeinsam mit den ausgetauschten Jobinformationen übertragen[4], um den expliziten Nachrichtenaustausch zwischen den Sites zu minimieren.

Des Weiteren verwenden die Autoren die ursprünglich von Eager et al. [25] eingeführten Begriffe *Transferstrategie* und *Lokationsstrategie*, welche für die sogenannten *Sender-orientierten*[5] Lastverteilungsverfahren definiert sind.

Die Transferstrategie legt für lokal eingehende Jobs fest, ob und welche davon delegiert werden sollen, wohingegen die Lokationsstrategie bei mehreren Delegationszielen bestimmt, an welche Site delegiert werden soll.

Die Umsetzung der Strategien erfolgt bei Lu et al. [77] dadurch, dass für jeden Job zunächst ermittelt wird, ob seine Delegation gegenüber der lokalen Ausführung einen Vorteil bringt. Dies hängt von einer Kostenfunktion ab, welche für jeden potentiellen Delegationspartner separat ermittelt wird und sich aus dem zuvor erwähnten Lastindex, der erwarteten Ausführungsdauer des betrachteten Jobs und den modellierten Kosten für den Datentransfer zusammensetzt.

Sind die Kosten für mindestens einen der Partner niedriger als bei der lokalen Ausführung, so wird der Job direkt an denjenigen mit der niedrigsten Last delegiert. Zusätzlich wird bei Änderung der lokalen Informationen über einen Partner (bei Eingang eines Fremdjobs) noch ein Lastverteilungsalgorithmus eingesetzt, der das gleiche Verfahren über alle lokal bereits eingereihten Jobs durchführt.

Trotz der dezentralen Informationshaltung wird auch in diesem Ansatz davon ausgegangen, dass die einzelnen Sites ihre vollständige Autonomie zugunsten eines gemeinsamen Lastverteilungsverfahrens aufgeben. Darüber hinaus findet die Evaluation des Ansatzes über einem sehr vereinfachten Workload-Modell statt, welches lediglich sequentielle Jobs vorsieht und wahrscheinlichkeitsverteilt die Einreich- und Laufzeiten der Jobs modelliert.

Die zuvor angesprochenen Konzepte (expliziter bzw. impliziter Lastinformationsaustausch, Sender-orientierte Verteilung, Evaluation mit einfachen synthetisch generierten Workloads) sind charakteristisch für viele aktuelle Ansätze im dezentralen Grid-Scheduling wie z.B. von Nandagopal et al. [89], Al-Azzoni & Down [2] oder Saravanakumar & Prathima [105], die versuchen auf Basis verteilter Informationshaltung einen Lastausgleich zu etablieren. Allen gemein bleibt der Verlust der vollen Site-Autonomie, da sie zwar die Kontrolle über ihre lokale Scheduling-Domäne besitzen, jedoch nicht über die zu akzeptierende fremde

[4] Die Autoren verwenden den Fachausdruck *mutual information exchange*.

[5] Hierbei gehen Delegationsanfragen stets von der Site aus, welche einen lokal eingereichten Job abgeben möchte. Die gegensätzliche Alternative wird entsprechend *Empfänger-orientiert* genannt.

Arbeitslast.

In dieser Hinsicht sticht der von Ernemann et al. [28] vorgestellte marktökonomische Ansatz heraus, der die lokale Site-Autonomie bezüglich beider Aspekte (Kontrolle über die Domäne und angenommene Last) wahrt und sogar innerhalb einer Domäne in Form eines hierarchischen Site-Modells umsetzt. So besitzen bereits innerhalb einer Site alle angebundenen Cluster/Rechenzentren einen eigenen *Ressourcenmanager*.

Wenn der Nutzer nun einen neuen Job bei seiner lokalen Site einreicht, gibt er zusätzlich noch eine Zielfunktion an, bezüglich der die Allokation seines Jobs optimiert werden soll. Dies kann z.B. eine geringe Wartezeit bis zur Ausführung des Jobs oder niedrige (monetäre) Kosten sein.

Auf Basis der Jobcharakteristika und der Zielfunktion ist der *Metamanager* der Site nun in der Lage, Angebote von entfernten Metamanagern und lokalen Ressourcenmanagern einzuholen, wobei auf jeder passierten Hierarchiestufe zur Minimierung des Kommunikationsaufwands immer nur das beste Angebot weitergeleitet wird.

Dieser auktionsorientierte Ansatz dient mehr als generelles Konzept, da er sehr viele konfigurierbare Freiheitsgrade besitzt. Das bezieht sich sowohl auf die seitens des Nutzers für jeden Job frei wählbare Zielfunktion als auch auf die bei jedem Ressourcenmanager individuell einstellbare Funktion zur Angebotserstellung. Aufgrund der vielen Freiheitsgrade evaluieren die Autoren auch nur einige wenige feste Konfigurationen und weisen so die generelle Funktionalität ihres Konzepts nach. Eine wirkliche (maschinelle) Optimierung der Zielfunktionen hinsichtlich ausgewählter Performanzkriterien bleiben die Autoren schuldig.

In genau diesem Bereich — der Optimierung von Job-Scheduling in Computational Grids mit maschinellen Lernverfahren— existieren ebenfalls einige Veröffentlichungen. Diese wurden von Fölling & Lepping [48] zusammengefasst und anhand bestimmter Kriterien (Grid-Modell, Repräsentation von Wissen, angewandte Lernverfahren und genutzte Zielfunktionen) kategorisiert.

Die Zusammenfassung macht zusätzlich deutlich, dass ausnahmslos alle Fremdarbeiten[6] entweder ein deterministisches Scheduling-Problem oder einen zentralen Grid-Scheduling-Ansatz ohne Site-Autonomie annehmen. Die Erforschung von Lernverfahren in einem dezentralen Grid-Scheduling-System unter Online-Scheduling-Bedingungen stellt damit ein Novum dar.

Unabhängig von der Modellierung des Optimierungsproblems existiert im Hinblick auf die Wissensrepräsentation eine gewisse Häufung von sogenannten *Fuzzy*-basierten Regelbasen. Diese Form der Entscheidungsfindung kann dazu genutzt werden, um in komprimierter Form Scheduling-Regeln zu definieren, die auch für unbekannte Eingaben (lokale Auslastung einer Site, Parallelität eines Jobs), zu robusten Scheduling-Entscheidungen führen. Für die detailliertere Beschreibung der

[6] Arbeiten, an denen der Autor dieser Dissertation nicht beteiligt war.

Eingabekodierung, des Regelaufbaus und der regelbasierten Entscheidungsfindung sei auf den späteren Abschnitt 7.2.4 verwiesen.

7.2 Eigene Ansätze für die Aktivitätendelegation

Die im Folgenden vorgestellten eigenen Ansätze zur Lösung des dezentralen Grid-Scheduling-Problems orientieren sich an dem marktökonomischen Ansatz von Ernemann et al. [28] und verzichten dabei vollständig auf den Austausch von dynamischen Lastinformationen zwischen den beteiligten Sites. Zusätzlich setzen sie evolutionäre Optimierungsverfahren ein, um einfache Heuristiken hinsichtlich ausgewählter Performanzmaße zu optimieren. Es wird gezeigt, dass die zugunsten der maschinellen Optimierungsverfahren — gegenüber der Vorlage — reduzierten Freiheitsgrade trotzdem vorteilhafte Scheduling-Lösungen zulassen.

Das verwendete dezentrale Grid-Modell geht dabei von einer vollständigen Site-Autonomie aus, in der jede Site nur mit Hilfe lokal erhebbarer Informationen (restriktive Informationspolitik) Delegationsentscheidungen trifft. Dabei wird auf übliche Metriken zur Lastbeschreibung und die Terminologie für Austauschstrategien zurückgegriffen. Die folgenden Inhalte (Modell, Heuristiken und lernorientierte Ansätze zur Delegation) wurden bereits im Rahmen eigener[7] Forschungsarbeiten [42, 43] veröffentlicht.

7.2.1 Erweiterung von Modell und Bewertungsgrößen

Das betrachtete Modell von Grid-Umgebungen baut grundsätzlich auf die in Kapitel 3 definierten Modellannahmen für MPP-Systeme auf. Durch die Verknüpfung von MPP-Systemen zu einem Grid kommen weitere Aspekte hinzu, die im Folgenden näher erläutert werden.

Viele der in Abschnitt 7.1 angesprochenen Veröffentlichungen versuchen bei der Verteilung der Arbeitslast innerhalb des Grids sowohl eine „Inter-Site-Heterogenität“ der Ressourcen als auch die bei der Übertragung entstehende Kommunikationszeit abzubilden.

Mit Inter-Site-Heterogenität ist zum einen gemeint, dass nicht alle Jobs auf allen Sites ausgeführt werden können. Zum anderen wird hierbei davon ausgegangen, dass die Maschinen (wenn auch innerhalb einer Site homogen) auf unterschiedlichen Sites auch unterschiedliche Rechengeschwindigkeiten besitzen. Für die Simulation solcher Systeme bedeutet dies — wie auch schon bei der Heterogenität innerhalb einer Site —, dass für jedes Tupel aus Job und Grid-weiten Rechenknoten eine Übertragungsfunktion nötig wäre, welche die Rechenzeit eines Jobs auf eben dieser Maschine berechnet. Aus diesem Grund werden die Sites innerhalb eines Grids in dieser Arbeit nicht nur intern sondern auch über Site-Grenzen hinweg als homogen angenommen.

Auch bei der Modellierung von Kommunikationszeiten existieren Schwierigkeiten. Denn zum einen braucht es ein Datenmodell, welches entweder ähnlich den in der Arbeit verwendeten Eingabe-Workloads (vgl. Abschnitt 4.1 auf S. 27) aus

[7] Beide Arbeiten sind in Zusammenarbeit mit Kollegen am Institut für Roboterforschung erstellt worden.

realen Aufzeichnungsdaten besteht oder auf einem statistischen Modell beruht und mit den realen Arbeitslastaufzeichnungen verknüpft werden muss. Zum anderen müssten auch spezifische Netzkonfigurationen mit entsprechend gegebenen Bandbreiten, redundanten Verbindungen und Netzwerkknotenpunkten modelliert werden.

All diese Probleme, die eher Teil der Forschungen des Daten-Schedulings im Sinne der sogenannten *Coallokation von Jobs und Daten* [106] sind, würden in unserem Fall zu einer unpraktikablen Vielzahl an Kombinationsmöglichkeiten führen und sind daher nicht Teil des verwendeten Modells. Darüber hinaus gibt es auch in der Praxis Ansätze zur Vermeidung von Kommunikationszeiten durch *Pre-/Postfetching*[8] [27] von Jobdaten, die eine gesonderte Berücksichtigung von Kommunikationszeiten zwischen den Sites weitgehend unnötig machen.

Auch wenn die Kommunikationszeiten zwischen den Sites nicht expliziter Teil des Systems sind, sei trotzdem darauf hingewiesen, dass die im weiteren Verlauf präsentierten Strategien in der Praxis auftretende Kommunikationszeiten indirekt z.B. als Teil verlängerter Antwortzeiten berücksichtigen und sich daran anpassen würden.

Des Weiteren ist in dem verwendeten Modell keine Site-übergreifende (*Multi-Site-*) Ausführung, wie z.B. von Ernemann et al. [28] eingesetzt, vorgesehen. Somit können parallele Jobs in der Gesamtheit nur auf Ressourcen genau einer Site ausgeführt werden, da genau wie bei der Inter-Site-Kommunikation auch für die Interprozesskommunikation zwischen parallelen Prozessen eines einzelnen Jobs keine explizite Modellierung vorgenommen wird. Diese Einschränkung deckt sich ebenfalls mit den meisten praktisch eingesetzten Grid-Umgebungen (vgl. Kapitel 6.1).

Mit dem Austausch von Arbeitslast zwischen den Sites werden auch weitere Bewertungsgrößen zur Untersuchung der Effekte notwendig, die über die Bewertung des lokalen Schedulings hinausgehen (vgl. Abschnitt 3.2 auf S. 22).

Eine Betrachtungsmöglichkeit besteht darin, die Veränderung an Arbeitslast im Vergleich zur lokal eingereichten Arbeitslast zu betrachten. Diese Metrik (VRP) kann für jede Site k in Anlehnung an das Ressourcenprodukt RP_k aus Abschnitt 3.2.2 durch Gleichung (7.1) berechnet werden.

$$\mathrm{VRP} = \frac{\sum\limits_{j \in \tau_k} m_j \cdot p_j - \mathrm{RP}_k}{\sum\limits_{j \in \tau_k} m_j \cdot p_j} \tag{7.1}$$

Während sich das ursprüngliche RP_k auf die Menge von lokal berechneten Jobs (π_k) bezieht, wird für die Bildung des eingereichten Ressourcenprodukts auf die

[8] Prefetching bezeichnet hierbei die gleichzeitige Übertragung von Eingabedaten und Postfetching die von Ausgabedaten während sich der eigentliche Jobs bereits bzw. noch in der Ausführung befindet.

Menge lokal eingereichter Jobs τ_k der Site k zurückgegriffen.

Eine Möglichkeit zur Untersuchung des Performanzgewinns (im Hinblick auf die Ausführungszeit) durch das Computational Grid liegt in der Änderung der AWRT bei Verwendung eines Austauschverfahrens gegenüber der AWRT bei einem Vergleichsexperiment. Im einfachsten Fall werden die Vergleichswerte einer Site durch rein lokales Scheduling ohne Teilnahme am Grid ermittelt. Die AWRT-Verbesserung (AWRT_I) berechnet sich durch Gleichung (7.2)

$$\mathrm{AWRT}_I = \frac{\mathrm{AWRT}_{Referenz} - \mathrm{AWRT}_{Austausch}}{\mathrm{AWRT}_{Referenz}} \tag{7.2}$$

Sie stellt somit ein Maß zur Bewertung der Vorteilhaftigkeit des Jobaustausch gegenüber einem Referenzverfahren dar.

Beide Metriken werden in der Regel und so auch in dieser Arbeit prozentual dargestellt. Zur besseren Lesbarkeit wurde bei beiden Metriken auf eine Site-Indizierung (z.B. VRP_k) verzichtet.

7.2.2 Aufbau von Strategien zur Aktivitätendelegation

Bei der eigenen Modellierung der Sites im dezentralen Grid wurde insbesondere auf eine starke Modularisierung der beteiligten Komponenten Wert gelegt. Nur auf diese Weise kann eine höchstmögliche Flexibilität in der Konfiguration erreicht werden. Bei diesem Modell reichen die Nutzer ihre Jobs, wie in Abbildung 7.1 dargestellt, bei ihrer lokalen Site direkt in das GRMS ein.

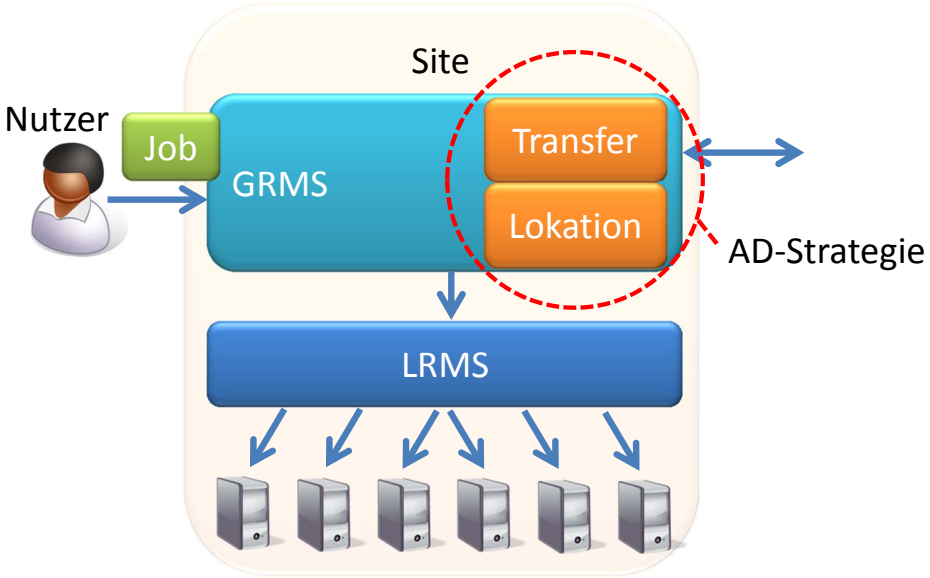


Abbildung 7.1: *Komponenten einer einzelnen Grid-Site.*

Aus Sicht der Nutzer ist die Teilnahme der Site an einem Grid-Verbund transparent. Es wird demnach eine Nutzersicht angenommen, in welcher der Ausführungsort der Jobs für die Nutzer nicht von Interesse ist sondern vielmehr die schnelle Abarbeitung derselben[9].

Eine solche Transparenz würde technisch z.B. darüber realisiert werden, dass die Einreichschnittstelle des GRMS mit der des LRMS identisch ist.

Das GRMS entscheidet nun mit Hilfe einer Transferstrategie[10], ob ein eingereichter Job abgegeben oder lokal ausgeführt werden soll. Zusätzlich befindet diese Strategie auch darüber, ob von extern angebotene Jobs abgelehnt oder übernommen werden sollen. Die Lokationsstrategie kann entweder nach oder vor dieser Entscheidung eingesetzt werden und bestimmt, mit welchen Delegationspartnern hinsichtlich der Übernahme von Arbeitslast verhandelt werden soll und in welcher Reihenfolge dies geschieht. Bei Entscheidung zur lokalen Ausführung eines Jobs leitet das GRMS diesen an das LRMS weiter. Zusammengenommen bilden beide Teilstrategien die Aktivitätendelegationsstrategie (AD-Strategie).

Die Zugriffstransparenz ist dabei nicht nur zwischen Nutzer und Site gegeben, sondern besteht auch zwischen GRMS und LRMS. Während das LRMS kein Wissen über die Existenz des GRMS besitzt und alle weitergeleiteten Jobs wie lokal eingereichte behandelt, benötigt das GRMS kein Wissen über die Konfiguration des LRMS (lokaler Scheduling Algorithmus, Anzahl und Konfiguration von Queues, usw.). Im Zuge dessen findet auch keine Anpassung zwischen den beiden Komponenten statt. Das GRMS hat lediglich Zugriff auf lokale Bewertungsgrößen wie z.B. momentane Auslastung oder wartende Jobs in der Queue, die aber auch von einem separaten Monitoring-System stammen können.

Der Austausch von Jobs zwischen GRMS und LRMS erfolgt zudem ausnahmslos in einer Richtung. Dadurch wird sichergestellt, dass sich die Jobs spätestens nach einer Verhandlung in dem LRMS der ausführenden Site befinden, um somit einer *Job-Starvation* (dem potentiell endlosen Verbleib eines Jobs im System) vorzubeugen.

Abbildung 7.2 zeigt detailliert, wie eine AD-Strategie — als Kombination aus Lokations- und Transferstrategie — für jeden lokal eingegebenen oder extern angebotenen Job zu einer Grid-Scheduling-Entscheidung gelangt.

Nach der lokalen Einreichung eines Jobs wird zuerst die Lokationsstrategie aufgerufen (ⓘ). Diese gibt eine geordnete Teilmenge aller bekannten Delegationspartner zurück. Auf diese Art und Weise kann noch vor der eigentlichen Verhandlung mit potentiellen Delegationspartnern beeinflusst werden, mit wem sich eine Verhandlung am meisten lohnt, was gleichzeitig auch zur Minimierung des Nach-

[9] Im realen Betrieb ist oftmals sehr wohl von Interesse, wo ein Job ausgeführt wird, sei es aus datenschutzrechtlichen oder manchmal auch politischen Gründen.

[10] Die Bezeichnungen der Strategien sind von Eager et al. [25] übernommen worden.

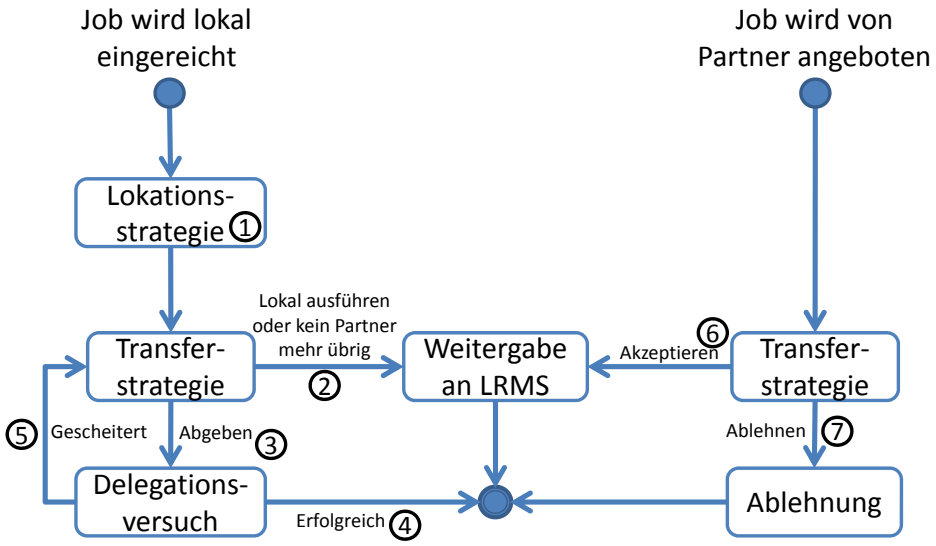


ABBILDUNG 7.2: *Ablauf der Entscheidungsfindung innerhalb einer Strategie zur Aktivitätendelegation (AD-Strategie).*

richtenaustauschs zwischen den Sites beiträgt. Auch die Implementierung nachbarschaftsorientierter Ansätze wie der von Lu et al. [77] sind auf diese Weise möglich.

Die geordnete Liste dient im weiteren Verlauf als Eingabe für die Transferstrategie, die nun für jeden Delegationspartner separat in der Lage ist, zu bestimmen, ob an diesen delegiert werden soll oder nicht. Auch wenn die gleichzeitige Anfrage mehrerer Delegationspartner prinzipiell möglich ist, wurde zunächst nur auf eine sequentielle Abarbeitung der Liste gesetzt, was wiederum zu der Minimierung der Kommunikation zwischen den Sites beiträgt.

Entscheidet die Transferstrategie während der sequentiellen Abarbeitung bei einem der Partner dafür, den Job lokal auszuführen, so wird die Schleife direkt abgebrochen und der Job an das LRMS weitergegeben ((2)).

Die Entscheidung für die Delegation hingegen führt zu einer Verhandlung mit dem entsprechenden Partner ((3)). Dieser Delegationsversuch kann entweder erfolgreich sein ((4)), womit die Aktivität delegiert und die Entscheidungsfindung abgebrochen wird, oder sie scheitert, was zur Abarbeitung des nächsten Partners in der Reihenfolge führt ((3)).

Sind alle Partner ohne erfolgreicher Delegation abgearbeitet worden, so bleibt der Site nichts anderes übrig, als die Aktivität selbst durchzuführen, weshalb sie ebenfalls an das LRMS weitergeben wird ((2)).

Für die Annahme eines Jobs bei einer externen Anfrage hingegen wird lediglich eine Transferstrategie genutzt. Diese kann sich entweder von der vorherigen Transferstrategie unterscheiden oder — wie im Rahmen unserer Untersuchungen

— deckungsgleich sein, ohne zwischen internen und externen Jobs zu differenzieren[11].

Genau wie auch die vorherige Transferstrategie, kann ein extern angebotener Job dabei akzeptiert und an das LRMS weitergegeben werden (⑥). Bei einer negativen Entscheidungen reicht in diesem Fall eine einfache Ablehnung des Jobs ⑦.

Das genutzte Grid-Modell sieht hierbei auch nicht vor, dass Ablehnungen in irgendeiner Art begründet werden. Dies unterstreicht noch einmal die sehr restriktive Informationspolitik, die Informationsaustausch zwischen den Sites bezüglich lokaler Gegebenheiten (Last, Konfiguration des LRMS, Implementierung der eigenen Strategie) vor den jeweils anderen Partnern verbirgt.

Rückschlüsse über das Verhalten bzw. die Qualität eines Delegationspartners soll lediglich auf Basis lokal bestimmbarer Größen geschehen.

7.2.3 Eine einfache Heuristik zur Realisierung einer Transferstrategie

Mit *Accept When Fit (AWF)* wurde als erster Schritt eine sehr einfache Heuristik evaluiert, die innerhalb des restriktiven dezentralen Grid-Modells eingesetzt werden kann.

Diese Heuristik beruht auf dem Prinzip, dass ein Job immer genau dann für die lokale Ausführung akzeptiert wird, wenn zum Entscheidungszeitpunkt t mindestens so viele lokale Ressourcen frei sind, wie der Parallelitätsgrad des entsprechenden Jobs beträgt, also die Bedingung $m_j \leq \tilde{M}_k(t)$ gilt (vgl. Abschnitt 3.1 auf S. 21).

Im Falle eines lokalen Jobs soll dieser also nur dann abgegeben werden, wenn er mehr parallele Maschinen benötigt als momentan verfügbar sind. Für einen extern angebotenen Job resultiert dieser Umstand hingegen in einer einfachen Ablehnung des Jobs ohne weitere Begründung.

Aufgrund der Trennung zwischen LRMS und GRMS wird bei der Entscheidungsfindung auf GRMS-Ebene keine Rücksicht auf detaillierte Eigenschaften des LRMS genommen (mögliches Backfilling des aktuellen Jobs, Beschaffenheit der Jobs in der Queue usw.).

Da für diese Entscheidungen keine Informationen über den entfernten Delegationspartner nötig sind und der entfernte Partner auch keine Kenntnis darüber besitzt, dass AWF eingesetzt wird, genügt die Heuristik damit auch den Anforderungen an eine restriktive Informationspolitik zwischen den Sites.

Die Transferstrategie, die im weiteren Verlauf als Referenzstrategie genutzt werden soll, wurde in paarweisen Kreuztests evaluiert. Hierbei wurden jeweils beide Sites des Setups mit AWF als Transferstrategie betrieben und die Veränderungen der Scheduling-Qualität hinsichtlich der AWRT und der bearbeiteten

[11] Streng genommen existieren auch in der deckungsgleichen Form Unterschiede in der Behandlung von internen und externen Jobs z.B. wenn ein externer Job einen höheren Parallelitätsgrad besitzt, als die Empfänger-Site Ressourcen hat. Ein solcher Job wird immer abgelehnt.

Last untersucht. Tabelle 7.1 zeigt die prozentualen Verbesserungen der AWRT ($AWRT_I$) und die Arbeitslaständerungen (VRP) bei lokaler Anwendung von FCFS für alle Paarungen nach Simulation für die Teilsequenzen zu fünf und sechs Monaten sowie bei Anwendung auf die gesamte Länge von elf Monaten.

		KTH		SDSC00		CTC		SDSC03		SDSC05	
	Mon	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
KTH	5			1	4	79	-12	78	-17	85	-25
	6			-35	1	26	-9	34	-18	30	-11
	11			13	1	79	-10	80	-18	83	-18
SDSC00	5	-124	-3			19	-11	5	-15	30	-23
	6	23	0			71	-3	84	-7	83	-5
	11	1	-1			69	-10	74	-18	76	-17
CTC	5	-9	3	1	4			0	2	19	-13
	6	-2	2	-13	3			8	-5	6	-4
	11	-4	2	-5	3			6	-2	9	-8
SDSC03	5	-5	1	-9	2	8	-1			39	-13
	6	-6	2	-6	3	2	2			-3	1
	11	-5	2	-7	2	4	1			20	-5
SDSC05	5	-3	2	0	3	-2	5	-7	14		
	6	2	1	3	1	2	1	8	-1		
	11	0	1	0	2	2	2	4	4		

TABELLE 7.1: *Ergebnisse der paarweisen Simulation von AWF für alle fünf Workloads in voller Länge und für Segmente zu* 5 *bzw.* 6 *Monaten unter Anwendung von FCFS als lokalen Scheduling-Algorithmus. Sowohl die Verbesserungen in der AWRT gegenüber der lokalen Ausführung ($AWRT_I$) als auch die Änderungen in der Last (VRP) wurden prozentual dargestellt. Bei den rot markierten negativen Werte handelt es sich um Verschlechterungen der AWRT.*

Bereits bei dieser einfachen Transferstrategie zeichnet sich ein eindeutiger Trend ab. In einem Setup, in dem sich die Sites hinsichtlich ihrer Größe und damit auch in ihren Workload-Charakteristika stark unterscheiden, profitieren vor allem die kleineren Sites von dem wechselseitigen Austausch von Arbeitslast. So reduziert sich bei der KTH-Site die AWRT um bis zu 85% beim Grid-Verbund mit der SDSC05-Site über die ersten fünf Monate.

Diese starke Performanzsteigerung bleibt der Site mit einer Gesamtverbesserung von 83% ebenfalls über die gesamte Laufzeit (elf Monate) erhalten. Die Steigerung ist in erster Linie auf den starken einseitigen Lastaustausch von 25% bzw. 18% der ursprünglichen KTH-Last hin zum größeren SDSC05 zurückzuführen. Im umgekehrten Fall muss der SDSC05 in diesem Setup bis zu 3% Performanzeinbußen in Kauf nehmen.

Nähern sich die Sites in ihrer Größe an, so weicht dieser eindeutige Trend einer starken Workload-Abhängigkeit bei der Simulation. Dies zeigt sich insbesondere in der paarweisen Simulation von KTH und SDSC00, die mit 100 zu 128 lokalen Maschinen eine ähnliche Größe besitzen. Hier kann der KTH in den ersten fünf Monaten mit 1% AWRT-Verbesserung moderat von dem Austausch profitieren. Dies geschieht in extremem Maße auf Kosten des SDSC00, bei dem sich mit einer $AWRT_I$ von -124% mehr als eine Verdopplung der AWRT bei lokaler Ausführung einstellt und das, obwohl die Migration von Arbeitslast zwischen den Parteien mit 3% bis 4% verhältnismäßig gering ist.

Der gegenteilige Fall (23% Verbesserung beim SDSC00 auf Kosten von 35%

Leistung beim KTH) stellt sich bei den nachfolgenden sechs Monaten ein. Wie in Abschnitt 4.3 bereits beschrieben kommt es bei den beiden Workloads häufig zur Blockade-Situation, in der hochparallele Jobs kleinere Jobs an der Abarbeitung hindern (vgl. Abschnitt 3.2.4 auf S. 24).

Allerdings treten diese Einzelfälle beim KTH-Workload gehäuft in den ersten fünf Monaten auf und beim SDSC00 hingegen in den darauf folgenden sechs Monaten. Die Konsequenz ist die je nach Beobachtungszeitraum wechselnde Performanz der beiden Parteien, da eine blockierte Site stärker von der Möglichkeit zum Austausch profitiert als der Verhandlungspartner, dessen abgegebenen Jobs ebenfalls in die Blockade geraten.

Obwohl die Transferstrategie nach einem sehr einfachen Muster verläuft und weder auf Laufzeitschätzungen noch auf Zustandsmaße wie Queue-Längen oder ähnliches zurückgreift, kann sie in manchen Konstellation bereits durch die Möglichkeit zum Arbeitslastaustausch zu beidseitigen Verbesserungen der Scheduling-Qualität führen.

Dies lässt sich beispielsweise anhand der Simulationen des Paares CTC und SDSC03 beobachten. Je nach Betrachtungszeitraum können sich hierbei entweder beide Sites gleichzeitig in der AWRT_I verbessern (trifft für die Simulation von sechs oder elf Monaten zu) oder die Verbesserung der einen Site führt zumindest nicht zur Verschlechterung der anderen (Simulation des fünfmonatigen Traces).

Dieses wechselseitige Verbesserungspotential wird noch stärker, wenn AWF — obwohl konzeptionell unabhängig vom LRMS — mit EASY als lokales Scheduling-Verfahren eingesetzt wird.

		KTH		SDSC00		CTC		SDSC03		SDSC05	
	Mon	AWRT_I	VRP	AWRT_I	VRP	AWRT_I	VRP	AWRT_I	VRP	AWRT_I	VRP
KTH	5			11	1	19	-9	24	-12	34	-23
	6			7	3	15	-7	19	-13	15	-5
	11			10	1	17	-8	22	-13	24	-12
SDSC00	5	7	-1			15	-7	20	-10	29	-20
	6	15	-1			28	-3	40	-5	39	-4
	11	12	-1			24	-8	34	-14	35	-14
CTC	5	2	2	2	2			7	3	16	-11
	6	2	2	1	3			14	-2	15	-1
	11	3	2	1	3			11	0	16	-4
SDSC03	5	0	1	-2	1	4	-1			24	-12
	6	0	1	1	2	3	1			5	1
	11	0	1	-1	2	3	0			15	-5
SDSC05	5	0	2	1	2	-1	4	2	13		
	6	-1	0	1	1	1	0	9	0		
	11	0	1	1	1	1	1	7	4		

TABELLE 7.2: *Ergebnisse der paarweisen Simulation von AWF für alle fünf Workloads in voller Länge und für Segmente zu* 5 *bzw.* 6 *Monaten unter Anwendung von EASY als lokalen Scheduling-Algorithmus.*

Tabelle 7.2 liefert eine Übersicht der Ergebnisse bei einem analogen Kreuztest mit EASY als lokalen Scheduling-Algorithmus. Da die AWRT-Performanz in allen Fällen bereits durch die lokale Anwendung von EASY gegenüber FCFS gesteigert wird, sinkt zwar das Verbesserungspotential der Delegationspartner durch

die Möglichkeit des Jobaustauschs, doch geschieht dies — mit Ausnahme von vier Einzelfällen — nicht auf Kosten des jeweiligen Delegationspartners.

Der Grund für die bessere Zusammenarbeit der AWF-Transferstrategie und EASY liegt darin, dass AWF seine Entscheidung zur lokalen Ausführung eines Jobs (egal ob lokal oder von extern) lediglich davon abhängig macht, ob der Job momentan ausführbar wäre. Ist dies der Fall, wird der Job an das LRMS übergeben, welches diesen — aufgrund der Trennung zwischen den beiden Schichten — am Ende der Queue einfügt.

Somit müssen ausgetauschte Jobs mit ihrer Abarbeitung auf der Fremdmaschine erst auf die Bearbeitung voranstehender Jobs der Queue warten, was wiederum den Vorteil des Jobaustauschs dämpft. Erst mit dem Einsatz von EASY werden ausgetauschte Jobs in vielen Fällen durch Vorziehen direkt gestartet. Dies hat zwei positive Effekte.

Zum einen trägt die frühere Ausführung des Fremdjobs zu einer Verringerung der AWRT des Delegationspartners bei. Zum anderen wird die Anzahl der freien Maschinen umgehend reduziert, was wiederum Auswirkungen auf folgende AWF-Entscheidungen hat und damit unter Umständen eine „Überschwemmung“ der Site mit zusätzlichem Fremd-Workload verhindert.

Zusammenfassend kann gesagt werden, dass besonders für den Fall, in dem keine Laufzeitschätzungen vorhanden sind, ein großes Optimierungspotential gegeben ist. Daher, und um das GRMS nicht von Laufzeitschätzungen abhängig zu machen, wird im weiteren Verlauf der Strategieentwicklung auf die Verwendung von Laufzeitschätzungen auf GRMS-Ebene verzichtet.

Gleichzeitig soll untersucht werden, in wie weit insbesondere Lernverfahren zur Effizienzsteigerung des Grid-Schedulings unter restriktiven Informationsbedingungen und bei Anwendung beider lokaler Scheduling-Algorithmen beitragen können.

7.2.4 Lernbasierte Realisierung einer Transferstrategie

Um eine jobbezogene und dezentrale Entscheidung zur Delegation zu treffen, sind stets mindestens zwei Kriterien notwendig. Das erste Kriterium beschreibt dabei den Job (z.B. die durch ihn entstehende Arbeitslast). Das zweite Kriterium beschreibt die Lastsituation auf der lokalen Site.

Die zuvor beschriebene AWF-Transferstrategie besaß bereits durch ihre Konzeption einen fundamentalen Nachteil, denn die bei der Entscheidungsfindung genutzte freie Parallelität einer Site als Last-Kriterium berücksichtigt die lokale Last der Site nur unzulänglich. Ihre eventuelle „Überlastung“ ist unter Umständen auch davon abhängig, wie viel Arbeit bereits in der Queue auf Abarbeitung wartet.

Die Länge der Queue gibt in diesem Sinne bereits mehr Aufschluss über die zu erwartende Arbeitslast, jedoch bleiben auch hier die Charakteristika der einzelnen Jobs in der Queue unberücksichtigt. Da Laufzeiten der Jobs im Rahmen des Online-Scheduling-Problems nicht gegeben sind und auf die Verwendung von geschätzten

Laufzeiten im GRMS verzichtet werden soll, bleibt lediglich die Parallelität der Jobs in der Queue als Bemessungsgrundlage übrig.

Die Ausprägung dieser Größe ist stark vom simulierten Workload bzw. der Site-Konfiguration abhängig. Eine größere Site besitzt oftmals auch Jobs höherer Parallelität und verwaltet zu Hochlastzeiten mehr Jobs innerhalb der Queue. Im Gegenzug ist sie allerdings in der Lage, mehr Arbeitslast (RP) pro Zeit zu verarbeiten.

Um diesem Umstand bei der Modellierung unserer Lastgröße Rechnung zu tragen, wird die „wartende Parallelität" einer Site noch mit der Zahl der lokalen Maschinen normiert. Somit ergibt sich die *Normalisierte Wartende Parallelität (NWP$_k$)* einer Site k mit ihrer Queue q_k durch Gleichung (7.3).

$$\mathrm{NWP}_k = \frac{1}{M_k} \cdot \sum_{j \in q_k} m_j \tag{7.3}$$

Die NWP ist ausgehend von 0 prinzipiell unbeschränkt. In der Praxis nimmt sie jedoch selten einen Wert an, der größer ist als 10. Um den späteren Lernraum zu begrenzen, wird die NWP daher auf das Intervall [0,10] beschränkt.

Analog dazu ist es für das jobbezogene Kriterium ebenfalls sinnvoll, die Parallelität des zu entscheidenden Jobs zu normalisieren, um eine Site-bezogene Größe zur Bemessung der durch den Job zusätzlich entstehenden Arbeit zu erhalten. Die *Normalisierte Job-Parallelität (NJP$_{(j,k)}$)* eines Jobs j auf einer Site k berechnet sich somit durch Gleichung (7.4).

$$\mathrm{NJP}_{(j,k)} = \frac{m_j}{M_k} \tag{7.4}$$

Die NJP ist auf dem Intervall]0,1] definiert, wird oft allerdings prozentual auf dem Intervall]0,100] dargestellt.

Kodierung von Entscheidungsregeln

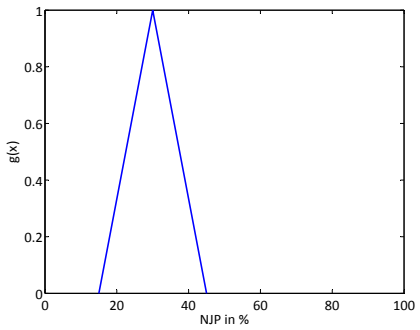
Wie bei den zuvor beschriebenen marktökonomischen Ansätzen zum Grid-Scheduling nach Ernemann et al. [28] sowie Buyya & Murshed [9] existiert nach der Modellierung der Problemgrößen nun die Schwierigkeit, dass nicht von vornherein klar ist, wie die Ausprägungen der Größen zu bewerten sind — mit anderen Worten: Bei welcher NWP befindet sich viel Arbeitslast in der Queue und bei welcher wenig? Dieser Umstand erschwert allerdings die Implementierung einfacher Heuristiken zur Entscheidungsfindung.

Die Lösung dieses Problems liegt zum einen in der Verwendung einer regelbasierten Transferstrategie, welche ihre Entscheidung auf Basis einfacher „Wenn-Dann-Regeln" trifft und zum anderen in der Art der Kodierung dieser Regeln. Jede der verwendeten Regeln besitzt dabei eine Bedingung, nach der die Regel greift und eine entsprechende Konsequenz. In unserem Fall basiert die

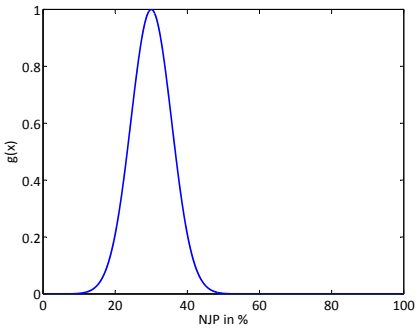
Bedingung einer Regel auf den zuvor definierten Eingangsgrößen zur Last- und Jobbeschreibung und in der Konsequenz spricht sich eine Regel dann für oder gegen das lokale Scheduling aus.

Für die Modellierung „unscharfer“ Übergänge (z.B. sehr niedrig⟷niedrig⟷mittel⟷hoch⟷sehr hoch) zwischen den Bewertungsgrößen wird auf Fuzzy-Mengen zurückgegriffen.

Der ursprünglich auf Zadeh [133] zurückgehende Begriff von *Fuzzy-Mengen* beschreibt eine Mengen-Repräsentation, bei der Elemente nicht nur binär in der Menge enthalten oder nicht enthalten sein können, sondern mit einer gewissen Wahrscheinlichkeit (ausgedrückt durch einen reellwertigen Zugehörigkeitswert $g(x)$ des Intervalls [0,1]) zur Menge gehören. Für die Bildung dieser Zugehörigkeitsfunktion existieren unterschiedliche Ansätze.



(a) Symmetrische Dreieckszugehörigkeitsfunktion mit Definition über Mittelwert und Abweichung.



(b) Gauß´sche Zugehörigkeitsfunktion (GZF).

Abbildung 7.3: *Beispiele für unterschiedliche Zugehörigkeitsfunktionen bei der Modellierung für „niedrige NJP“.*

Abbildung 7.3 zeigt beispielhaft die Modellierung des *linguistischen* Ausdrucks „niedrige NJP“ anhand einer symmetrischen Dreiecksfunktion[12] und einer Gauß´sche Zugehörigkeitsfunktion (GZF) im Vergleich.

Während die symmetrische Dreiecksfunktion in der Abbildung über den Mittelwert 30 und die Abweichung 15 spezifiziert wird, ist die GZF über einen Mittelwert $\kappa = 30$ und eine Standardabweichung $\sigma = 8$ nach Gleichung (7.5) definiert[13].

$$g(x) = \exp\left\{\frac{-(x-\kappa)^2}{\sigma^2}\right\} \tag{7.5}$$

[12]Neben einer symmetrischen Dreiecksfunktion sind ebenfalls die asymmetrische Form sowie eine Trapezfunktion möglich, welche allerdings beide in zusätzlichen Parametern für die Beschreibung resultieren und damit den Suchraum vergrößern.

[13]Entgegen der sonst üblichen Namensgebung γ für den Mittelwert wird in dieser Arbeit κ verwendet, um den Namenskonflikt mit der *Discount-Rate* γ des Q-Learnings in Abschnitt 5.3 aufzulösen.

Der Vorteil der GZF gegenüber der Dreiecksfunktion liegt in der Modellierung „weicherer“ Übergänge zwischen den Zugehörigkeiten, was die spätere Anpassung an den Lösungsraum (im Scheduling-Problem) begünstigt. Daher wird die GZF im weiteren Verlauf für die Kodierung des Bedingungsteils von Regeln eingesetzt.

Für die Kodierung der Regeln ist das verwendete *Inferenz-System* ausschlaggebend, welches für einen Vektor $\vec{x}$ von Eingangsgrößen (in unserem Fall die aktuellen Ausprägungen von NWP und NJP) eine Ausgangsgröße/Konsequenz y (in unserem Fall eine binäre Entscheidung) berechnet.

Aufgrund der vereinfachten Ausgangsgrößenberechnung[14] wurde das *TSK-Modell* nach Takagi & Sugeno [122] zur Regelkodierung verwendet[15].

Hierbei wird eine Regelbasis aus N_r Regeln R_i gebildet, die auf N_f Eingangsgrößen beruhen und der Form aus Gleichung (7.6) entsprechen.

$$\begin{aligned} R_i \quad := \quad & \text{WENN } x_1 \text{ ist } g_i^{(1)} \text{ und } \ldots \text{ und } x_{N_f} \text{ ist } g_i^{(N_f)} \\ & \text{DANN } y_i = b_{i0} + b_{i1}x_1 + \ldots + b_{iN_f}x_{N_f} \end{aligned} \tag{7.6}$$

Bei der Auswertung des Bedingungsteils einer Regel bei gegebener Eingabe $\vec{x} = (x_1, x_2) = (\text{NWP,NJP})$[16] wird dann zunächst der *Zugehörigkeitsgrad* $\phi_i(x)$ jeder Regel (R_i) bestimmt durch Gleichung (7.7).

$$\phi_i(\vec{x}) = g_i^{(1)}(x_1) \wedge g_i^{(2)}(x_2) = g_i^{(1)}(x_1) \cdot g_i^{(2)}(x_2) \tag{7.7}$$

Analog zu den durch die $g_i^{(j)}$ beschriebenen Zugehörigkeitsfunktionen (GZF) von Einzelvariablen ist der Regler so ebenfalls in der Lage den „Grad der Erfüllung“ einer einzelnen Regelbedingung mit $\phi_i(\vec{x}) \in [0,1]$ zu berechnen. Die Konsequenz y_i einer Regel R_i wird in unserem Fall binär kodiert mit $(+1)$ für die lokale Ausführung eines Jobs und (-1) für die Abgabe bzw. Ablehnung eines Jobs[17].

Um von den Zugehörigkeitsgraden $\phi_i(\vec{x})$ und den Ausgaben y_i aller Regeln R_i einer Regelbasis zur Reglerausgabe $y_D(\vec{x})$ zu gelangen, wird lediglich das gewichtete Mittel über die Zugehörigkeitsgrade der Einzelkonsequenzen nach Gleichung (7.8) gebildet.

$$y_D(\vec{x}) = \frac{\sum_{i=1}^{N_r} \phi_i(x) \cdot y_i}{\sum_{i=1}^{N_r} \phi_i(x)} \tag{7.8}$$

[14] Gegenüber anderen Modellen wie z.B. dem Mamdani-Modell [80], welches neben den Eingaben auch mehrere Fuzzy-Mengen als Ausgabe besitzt, werden beim TSK-Modell „scharfe“ Werte als Ausgabe modelliert.

[15] Für eine umfassende Beschreibung verschiedener Modelle zur Inferenzbildung und den damit verbundenen Operatoren sei auf den Sammelband von Kandel & Langholz [67] zum Thema *Fuzzy Control Systems* verwiesen.

[16] In unserem Fall existieren nur zwei Eingangsgrößen $\rightarrow N_f = 2$.

[17] Das entspricht dem Ansatz, dass bei der linearen Kombination $y_i = b_{i0} + b_{i1}x_1 + \ldots$ alle Gewichte b_i gleich Null gesetzt werden und nur das Gewicht b_{i0} direkt die Entscheidung bestimmt. Es handelt sich somit um eine vereinfachte Form des TSK-Modells

Analog zu den Einzelregeln wird auch die Reglerausgabe y_D binär interpretiert. Eine Gesamtausgabe $y_D \geq 0$ entspricht der lokalen Ausführung bzw. Akzeptanz eines Jobs, wohingegen eine Ausgabe $y_D < 0$ unter den Eingangsgrößen $\vec{x}$ zu einer versuchten Abgabe bzw. Ablehnung des Jobs führt. Abbildung 7.4 visualisiert

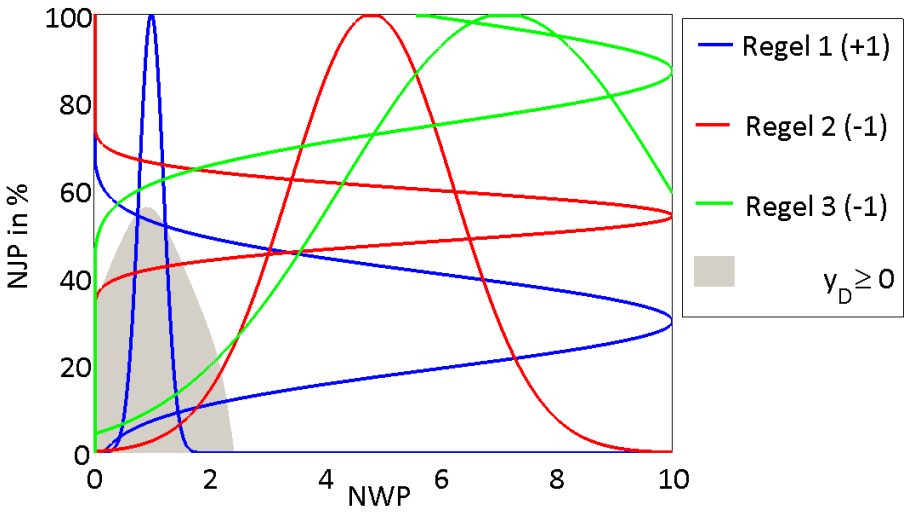


Abbildung 7.4: *Beispiel für die Überlagerung von drei Regeln (*15 *Parameter) und der dazugehörigen Reglerausgabe. Regel* 1 *besitzt in ihrer Konsequenz ein positives Ergebnis, wohingegen die übrigen Regeln ein negatives Ergebnis besitzen.*

beispielhaft eine Regelbasis aus drei Regeln und den durch die Überlagerung der Regeln berechneten positiven Entscheidungsbereich ($y_D \geq 0$)[18]. Die gezeigte Regelbasis würde demnach keine Jobs mit mehr als 60% der Site-Größe akzeptieren und diese auch nur in Situationen, in denen die wartende Parallelität in der Queue nicht viel größer als das zweifache der konfigurierten Site-Größe beträgt[19].

Zusammenfassend kann eine solche TSK-Regelbasis für die Transferstrategie, die aus N_r Regeln besteht umfassend durch $l = N_r \cdot 5$ Variablen beschrieben werden, da für jede Regel N_i zwei GZF für NJP und NWP mit jeweils einem Mittelwert und einer Standardabweichung sowie eine binäre Ausgabe (+1 oder -1) nötig sind.

[18] Für die bessere Darstellbarkeit wurden die einzelnen GZF auf den Maximalwert des jeweils anderen Kriteriums gestreckt.

[19] Diese Angaben sollen nur dem allgemeinen Verständnis dienen und sind demnach etwas ungenau, z.B. in dem Bereich mit NJP $\in$ [40,60] bei sehr geringer NWP.

Training von Regelbasen

Das Training einer Regelbasis findet über Evolutionsstrategien statt, welche grundlegend bereits in Abschnitt 5.1 eingeführt wurden. In der Literatur wird ein solches System, welches Fuzzy-kodierte Regler evolutionär optimiert, auch *Evolutionäres Fuzzy System* genannt. Einen guten Überblick zu diesem Thema mit Ausführungen, die über den Detailgrad dieser Arbeit hinausgehen, liefert das Werk von Cordón et al. [17].

Im Rahmen dieses Abschnitts wird erläutert, wie ein einzelnes Trainingssetup aufgebaut ist und wie die Kodierung des Lernproblems erfolgt, bevor eine Diskussion der Ergebnisse erfolgt.

Ziel eines Trainings ist es, das Verhalten einer *Trainings-Site* dahingehend zu optimieren, dass sie bei Interaktion mit einem *Trainingspartner* eine möglichst hohe Scheduling-Performanz für die lokale Nutzergemeinschaft erreicht.

Damit sich die Umwelt, mit der die lernende Site interagiert, während des Trainingsvorgangs zunächst nicht ändert, wird — wie in Abbildung 7.5 dargestellt — der Trainingspartner mit der zuvor evaluierten Referenzstrategie AWF konfiguriert.

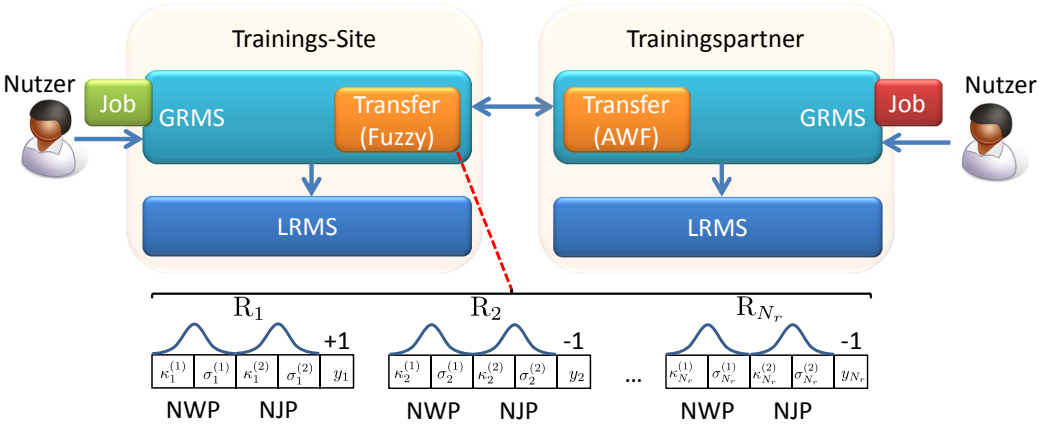


Abbildung 7.5: *Trainingssetup für das Erlernen von vorteilhaften Regelbasen für den Jobaustausch.*

Innerhalb der Evolutionsstrategie stellt jede Lösung des Problems (in Form einer Regelbasis) ein eigenständiges Individuum dar. Damit folgt der Evolutionskreislauf dem sogenannten *Pittsburgh-Ansatz* [116], in dem sich die Reproduktion und Variation der Individuen über vollständigen Regelbasen vollzieht.

Um gleichzeitig eine hohe Granularität der Reglerausgaben zu erhalten und den Parameterraum des Optimierungsproblems überschaubar zu halten, wurde die Anzahl von Regeln pro Individuum auf 10 beschränkt. Damit ergibt sich für jedes Individuum k ein Objektparametervektor $\vec{o}_k$ der Dimension $u = 50$, mit 40 reellwertigen (GZF) und 10 binären (Konsequenzen) Parametern (vgl. Abschnitt 5.1 auf S. 36).

Die EA-Parameter $\vec{s}_k$ eines Individuums k hingegen führen die Mutationsschrittweite für die beiden Größen NWP und NJP[20], sowie die Mutationsrate ω für die „Kippwahrscheinlichkeit“ der binären Regelkonsequenzen.

Da die beiden Bedingungsgrößen sich im Maximalwert um den Faktor 10 unterscheiden, wurde dieses Verhältnis auch bei der Initialisierung der Mutationsschrittweiten berücksichtigt. Die initiale Schrittweite für die NWP wurde daher mit 0,01 und die der NJP mit 0,001 festgelegt. Die Mutationsrate ω ist zu Beginn 0,1.

Der Parameter β zur Schrittweitenanpassung über den Zweipunktoperator wurde mit $\beta = 0{,}6$, wie von Kost [69] empfohlen, maximal gewählt.

Die Populationsgröße der Evolutionsstrategie wurde mit $\mu = 13$ und $\lambda = 91$ im $(1 : 7)$-Verhältnis bei ausgewählter PLUS-Strategie zu insgesamt 104 Individuen gewählt.

Die Objektparameter $\vec{o}_k$ der μ Startindividuen wurden zu Beginn gleichverteilt zufällig auf ihren jeweiligen Wertebereichen bestimmt. Die Optimierung einer Regelbasis wurde im Anschluss über 150 Generationen durchgeführt.

Als Zielfunktion für die Optimierung — und damit als Performanzmaß für das Scheduling der lokalen Nutzergemeinschaft der Trainings-Site — wurde die AWRT gewählt.

Demnach ist die Fitness eines Individuums A besser als die eines Individuums B wenn das — in der Regelbasis von A kodierte — Austauschverhalten zu einer geringeren AWRT führt als das Verhalten der gleichen Site in Individuum B.

Zentrale Forschungsfragen des Trainings waren unter anderem, ob sich die trainierte Site in jedem Trainingsvorgang gegenüber ausschließlich lokaler Ausführung verbessert und wenn ja, ob ihr dies nur auf Kosten des jeweiligen Trainingspartners gelingt. Des Weiteren galt es zu untersuchen, ob das Verbesserungspotential einer Site proportional zur Größe des Trainingspartners steigt, da ein größerer Trainingspartner potentiell über mehr freie Rechenkapazitäten verfügt. Die nachfolgenden Experimente haben gezeigt, dass der Lastaustausch zwischen den Sites bereits in der Trainingssituation komplex ist, da es sowohl Konstellationen gibt, in denen sich die Trainings-Site im Gegensatz zum Partner verschlechtert als auch Konstellationen in denen beide Sites von dem alleinigen Training einer Partei profitieren. Gleichzeitig konnte eine Proportionalität zwischen der Größe es Trainingspartners und den Verbesserungen für die trainierte Site widerlegt werden. Neben einem Performanzvergleich mit der AWF-Strategie war es ebenfalls wichtig, zu untersuchen, inwieweit das erlernte Verhalten in Abhängigkeit der Trainingskonstellation bzw. in Hinblick auf den Austausch des lokalen Scheduling-Algorithmus während der Optimierung variiert. Hier hat sich gezeigt, dass ein erlerntes Verhalten immer zu besseren Resultaten führt, als die Grid-weite Verwendung von AWF. Allerdings geht dies auch mit einer zunehmenden Spezialisierung einher, die sich häufig in einer Verschiebung der

[20] Sowohl für den Mittelwert κ als auch die Standardabweichung σ einer einzelnen Größe wird die gleiche Schrittweite genutzt.

Arbeitslast von der trainierten Site hin zum Trainingspartner äußerte. Erst mit dem Austausch des lokalen Scheduling-Algorithmus FCFS durch EASY während des Trainings entsprach das Lernverhalten den Erwartungen und führte in erster Linie zu Verbesserungen kleinerer Trainings-Sites auf Kosten der größeren Partnern sowie beidseitigem Performanzgewinn, wenn die größere Site einer Konstellation trainiert wurde.

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
Partner			-16	10	-2	8	-9	2	-7	3
KTH			83	-14	86	-32	84	-25	88	-37
Partner	84	15			-3	8	-4	2	-2	3
SDSC00	-6	-11			25	-25	15	-17	32	-28
Partner	87	4	21	16			14	-1	-3	6
CTC	6	-1	8	-5			8	4	21	-15
Partner	72	-6	-42	4	-34	17			-4	17
SDSC03	7	0	7	0	26	-6			47	-16
Partner	87	-30	-4	-12	4	0	48	-3		
SDSC05	2	3	5	1	13	0	17	3		

TABELLE 7.3: *Ergebnisse der paarweisen Trainingsdurchläufe für alle fünf Workloads auf dem Trainingszeitraum von fünf Monaten und unter Einsatz von FCFS. Die $AWRT_I$ ist erneut gegenüber rein lokalem Scheduling berechnet.*

Die Ergebnisse an prozentualer $AWRT_I$ und VRP für die paarweisen Trainingsläufe mit FCFS sind in Tabelle 7.3 gegeben. Die Berechnung der rund 20 separaten Evolutionsstrategien benötigte auf dem zur Verfügung stehenden Rechencluster ca. eine Woche, weshalb eine vollständige Analyse aller Setups mit mehreren Läufen (z.B. 10 Wiederholungen) nicht möglich war. Die Auswirkungen unterschiedlicher Startpopulationen und Populationsentwicklungen wurden stichpunktartig anhand von drei unterschiedlichen Setups (KTH-SDSC00, CTC-SDSC05 und SDSC03-KTH) untersucht. Die Abweichungen in den resultierenden Regelbasen und damit auch den Ergebnisgrößen waren allerdings so gering, dass diese nicht ausschlaggebend für die in der Evaluation gemachten Aussagen sind. Darüber hinaus wird eine Untersuchung mit mehreren Läufen später in kleinerem Umfang und bei der Evaluation der Coevolutionären Optimierung in Abschnitt 7.2.6 durchgeführt.

Neben der Performanz der jeweiligen Trainings-Site (untere Zeile eines Einzelergebnis) ist auch die dabei erreichte Performanz des Trainingspartners (obere Zeile) gegeben. Die beiden Metriken sind — analog zu den vorherigen Ergebnissen bei der Untersuchung von AWF — in Bezug zur lokalen Ausführung angegeben.

Grundsätzlich ist bei dem durchgeführten einseitigen Training eines einzelnen Workloads gegen einen unveränderlichen Partner davon auszugehen, dass die Trainings-Site zu einer Regelbasis kommt, bei der sie sich auf Kosten des Trainingspartners hinsichtlich der AWRT verbessern kann, da dieser nicht optimiert wird.

Am einfachsten ist dies durch die verstärkte Abgabe eigener Arbeitslast an den Trainingspartner möglich.

Deutlich zu erkennen ist dieser Umstand z.B. bei allen Trainingsläufen der KTH-Site, welche je nach Trainingspartner Verbesserungen zwischen 83% und 88% erreicht und dafür zwischen 14% und 37% weniger Arbeitslast bearbeitet, als lokal bei ihr eingereicht wird.

Die Annahme, dass die Verbesserung mit zunehmender Größe des Delegationspartners steigt, kann nicht bestätigt werden. Während beispielsweise die SDSC00-Site gegen CTC noch eine Verbesserung von 25% erreicht, schrumpft die Verbesserung gegen die mehr als doppelt so große SDSC03-Site auf 15%.

Auch wenn sich fast alle Sites in allen Trainingsdurchläufen verbessern können, so gilt dies für das Training des SDSC00 gegen KTH nicht. Hierbei handelt es sich um genau jenes Setup, welches bereits bei der Simulation von beidseitigem AWF zu erheblichen Problemen geführt hat (vgl. Abschnitt 7.2.3 auf S. 68).

Gegenüber dem AWF-Ergebnis mit einer mehr als doppelt so hohen AWRT wie bei lokaler Ausführung ist eine Verschlechterung von 6% allerdings marginal.

Viel interessanter ist die Tatsache, dass sich die SDSC00-Site bei dem Training hinsichtlich der AWRT verschlechtert hat, obwohl sie 11% ihrer lokalen Arbeitslast an den Trainingspartner abgibt, der wiederum stärker von dem Austausch profitiert als wenn die Trainingsrollen vertauscht sind (84% AWRT_I anstatt 83%)[21]. Dies ist ein deutlicher Hinweis darauf, dass der Jobaustausch zwischen den Parteien sehr komplex ist und — obwohl gewisse Tendenzen bestehen — nicht immer nur von der Größe der Parteien oder der Rollenverteilung (wer trainiert wird und wer nicht) abhängig ist.

Die beobachtete Vorteilhaftigkeit für den Trainingspartner nimmt sogar zu, je mehr Einfluss die trainierte Site auf das Grid besitzt. So kann die SDSC05-Site ihre Performanz je nach Partner um 2% bis 17% steigern, was mit Ausnahme des SDSC00 auch bei den jeweiligen Trainingspartnern zu merklichen Verbesserungen zwischen 4% und 87% führt. Diese Leistung gelingt allein durch den geschickten (und erlernten) Austausch von Jobs zwischen den Parteien, bei dem sich verhältnismäßig wenig Workload vom Trainingspartner zum SDSC05 hin verschiebt ($0 \leq \text{VRP} \leq 3$).

Die optimierten Regelbasen lassen sich hinsichtlich der betrachteten Kriterien auch im Vergleich zu den AWF-Ergebnissen für den gleichen Zeitraum (fünf Monate) darstellen. Dafür wurden in Tabelle 7.4 alle Größen nicht mehr in Relation zur lokalen Ausführung sondern zur Anwendung von AWF auf beiden Sites berechnet.

Dadurch werden die Vorteile eines lernbasierten Ansatzes gegenüber einer heu-

[21] Auch wenn der Unterschied zwischen den Resultaten quantitativ nicht besonders groß ist, so ist er auffällig, zumal die AWRT_I der KTH-Site deutlich niedriger sein müsste, wenn die nur Trainingspartner ist.

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
Partner			48	14	6	5	-4	1	-4	1
KTH			83	-17	36	-22	30	-10	16	-15
Partner	84	10			-3	5	4	0	-2	1
SDSC00	53	-8			7	-16	11	-2	2	-7
Partner	40	19	2	30			6	-1	-1	1
CTC	13	-4	8	-8			8	2	2	-2
Partner	-28	13	-49	22	-34	15			3	3
SDSC03	12	-1	15	-2	20	-5			14	-3
Partner	11	-7	-49	15	-19	15	14	11		
SDSC05	4	0	5	-1	14	-4	23	-9		

TABELLE 7.4: *Ergebnisse der paarweisen Trainingsdurchläufe für alle fünf Workloads auf dem Trainingszeitraum von fünf Monaten und im Vergleich zur AWF-Performanz im gleichen Zeitraum und ebenfalls mit FCFS. Hierbei wurde als Referenz für die $AWRT_I$-Berechnungen auf die entsprechenden AWRT-Ergebnisse unter Anwendung von AWF zurückgegriffen. Um auch die VRP relativ darstellen zu können, wurden die Differenzen der VRP-Ergebnisse vom Lernverfahren und der Anwendung von AWF gebildet.*

ristischen Lösung deutlich. Jede Trainings-Site ist in der Lage ihre AWRT zu senken, wenn sie auf ein vorheriges Training setzt, anstatt AWF als Transferstrategie einzusetzen.

Mit Ausnahme eines einzelnen Setups (CTC gegen SDSC03) führt das gezielte Training einer Partei zu einer stärkeren Abgabe der eigenen Arbeitslast als noch bei der Anwendung von AWF. Dies gilt auf der einen Seite für eine KTH-Site, die bereits gegenüber der lokalen Ausführung 14% bis 37% mehr Arbeitslast delegiert als akzeptiert (vgl. Tabelle 7.3 zuvor) und somit im Vergleich zu AWF als Transferstrategie zwischen 10% (gegen SDSC03) und 22% (gegen CTC) mehr abgibt.

Auf der anderen Seite gilt dies auch für eine SDSC05-Site, die stärker von der Rechenkapazitäten des Trainingspartners profitiert, als dies noch bei AWF der Fall war (gleichbleibend gegen KTH bis hin zu 9% mehr Abgabe gegen den nächstgrößeren Partner SDSC03).

Mit dem Austausch des lokalen Scheduling-Algorithmus FCFS gegen EASY verschwinden die auf Blockade-Situationen zurückzuführenden Schwierigkeiten sowohl für den Einsatz von AWF auf beiden Seiten sowie beim paarweisen Lernen. So führt wie in Tabelle 7.5 dargelegt nun jeder Trainingsdurchlauf zu einer Verbesserung der Trainings-Site.

Zwar existieren nun keine Verbesserungen jenseits der 40% mehr, da durch die Verwendung eines effizienteren lokalen Scheduling-Algorithmus auch das Potential durch den Jobaustausch sinkt. Dafür entsprechen die Ergebnisse im höheren Maße den Erwartungen.

So erlangen ausnahmslos alle trainierten Sites eine höhere oder zumindest gleichgroße Performanzsteigerung mit Vergrößerung des Trainingspartners. Das Training gegen größere Trainingspartner (alle Setups über den graugefärbten Bereichen) führt stets zu einer Verbesserung auf Kosten des Partners, wohingegen das

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
Partner KTH			-13 20	16 -21	-10 30	15 -61	-9 31	5 -59	-2 36	7 -83
Partner SDSC00	6 9	-35 27			-9 21	16 -49	-13 23	7 -63	-2 32	5 -43
Partner CTC	19 4	-35 9	0 5	2 -1			-7 14	8 -23	-4 18	17 -47
Partner SDSC03	5 4	5 0	3 4	4 0	1 13	23 -8			-5 28	31 -29
Partner SDSC05	31 2	-20 2	27 2	-19 2	1 5	9 -3	24 7	-3 3		

TABELLE 7.5: *Ergebnisse der paarweisen Trainingsdurchläufe für alle fünf Workloads auf dem Trainingszeitraum von fünf Monaten und unter Einsatz von EASY.*

Training gegen kleinere Partner (alle Setups unter den graugefärbten Bereichen) die Performanz des Trainingspartners eher ebenfalls steigert als senkt.

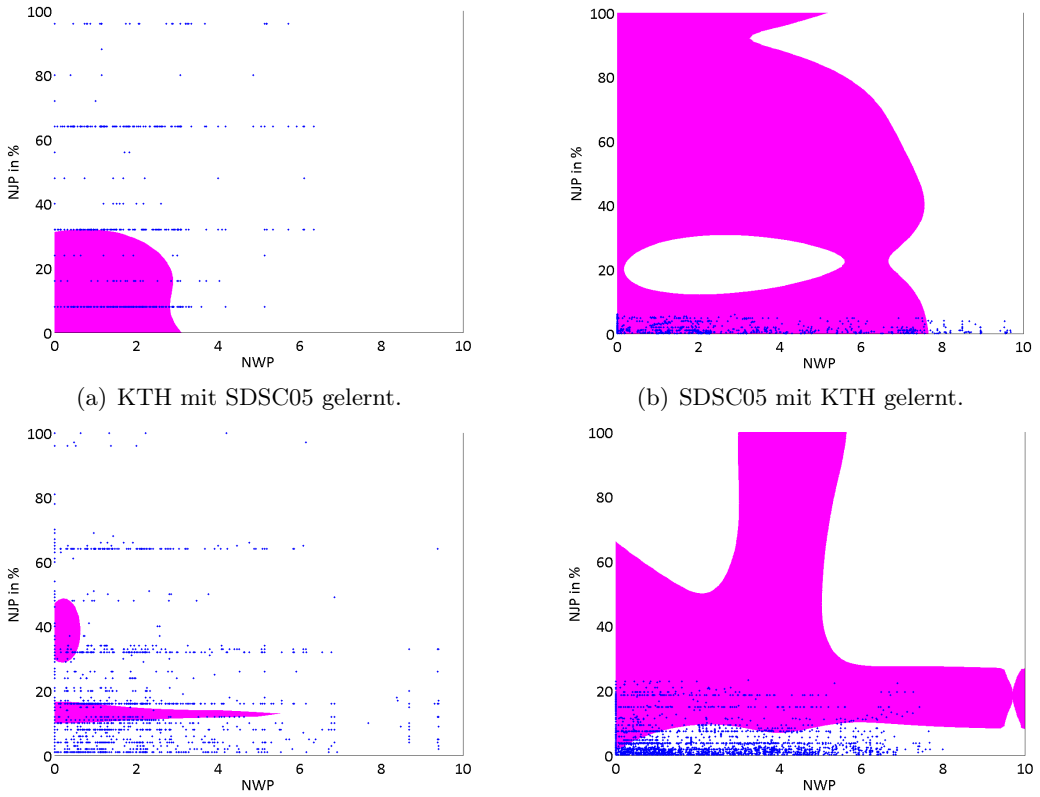
Somit werden die kleineren Trainings-Site eines Setups stets dahingehend optimiert, dass sie die größere Site ausnutzen, während die größeren Sites im umgekehrten Fall ihre Verbesserungen nur durch den gezielten Austausch des lokal bearbeiteten Workloads erreichen können. Dieses Verhalten ist für die größeren Sites nur möglich, wenn sie auch die Performanz des Partners steigern, damit dessen Ressourcen im Gegenzug öfter für die Abarbeitung der eigenen Arbeitslast zur Verfügung stehen. Obwohl die Regelbasen aufgrund der einseitigen Fitnessberechnung sehr egoistisch gelernt werden, so müssen gerade die größeren Partner ihre egoistische Position aufgeben, um sich zu verbessern.

Abbildung 7.6 zeigt beispielhaft vier Kennlinienfelder für Regelbasen (vergleichbar mit der Regelüberlagerung in Abbildung 7.4), die jeweils aus einem paarweisen Trainingslauf resultieren. In allen vier Beispielen war KTH an der Bildung der Regelbasis beteiligt — zweimal als trainierte Site und zweimal als Trainingspartner. Da sowohl NJP als auch NWP einer Anfrage immer bezüglich der trainierten Site berechnet werden, finden sich in den Abbildungen (a) und (c) mit KTH als Trainings-Site auch Anfragen in höheren Bereichen der NJP als dies bei (b) und (d) mit den größeren CTC und SDSC05 als trainierte Sites der Fall ist.

Vergleicht man die beiden Regelbasen miteinander, in denen KTH trainiert wurde, so wird deutlich, dass der Bereich der Akzeptanz von Arbeitslast (positive Konsequenz) insgesamt kleiner wird, wenn sich die Zahl der Anfragen[22] erhöht. So findet beim Training mit dem CTC (Abbildung 7.6(c)) eine starke Spezialisierung auf zwei Bereiche statt, die nur einen geringen Anteil der Anfragen positiv beantworten.

Nur die Jobs, die in einen solchen Bereich fallen, werden direkt lokal ausgeführt. Beim Training mit dem SDSC05 hingegen (Abbildung 7.6(a)) ist es grundsätzlich sinnvoller, kleine Jobs (NJP unter 30%) bei geringer wartender Parallelität (NWP unter 3) lokal auszuführen, anstatt sie abzugeben bzw. abzulehnen. Insgesamt

[22] Bei den dargestellten Anfragen wird nicht zwischen Anfragen aus externen oder lokal eingereichten Jobs unterschieden.



(a) KTH mit SDSC05 gelernt.

(b) SDSC05 mit KTH gelernt.

(c) KTH mit CTC gelernt.

(d) CTC mit KTH gelernt.

Abbildung 7.6: *Beispiele für Regelbasen bei der paarweisen Optimierung über ein evolutionäres Fuzzy-System. Alle gezeigten Regelbasen stammen aus fünfmonatigen Trainingsdurchläufen mit FCFS als lokalen Scheduling-Algorithmus. Während die farbigen Bereiche die mit einer positiven Regelkonsequenz darstellen, steht jeder der blauen Punkte für eine Anfrage an die Regelbasis.*

existieren hier allerdings auch viel weniger Delegationsanfragen seitens des Trainingspartners, da dieser gemäß der verwendeten AWF-Heuristik seine Jobs nur abzugeben versucht, wenn die eigenen Ressourcen momentan nicht ausreichen.

Gleichzeitig unterscheiden sich die Regelbasen sehr stark in Abhängigkeit von der eigenen Größe. Während die kleinere KTH-Site bei ihrem Training gegen die SDSC05-Site (Abbildung 7.6(a)) die 88% verbesserte AWRT (vgl. Tabelle 7.3) durch die vermehrte Abgabe von Arbeitslast (Anfragen an die Regelbasis außerhalb des Akzeptanzbereichs) erreicht, ist die Strategie der größeren SDSC05-Site eine ganz andere. Sie setzt viel mehr darauf, alle Jobs, die in einer niedrigen Lastsituation eingehen, zu akzeptieren und durch Abgabe einzelner Jobs in Zeiten hoher Last einen Performanzvorteil zu erlangen (Abbildung 7.6(b)). Letzteres Verhalten bestätigt auch die zuvor getroffene Aussage, dass größere Sites selbst nur durch die verstärkte Übernahme von Fremdjobs vom Austausch profitieren können.

In diesem Extremfall (Größenunterschied mit Faktor 16) schadet die bedingungslose Übernahme von Jobs der eigenen Performanz der SDSC05-Site offensichtlich nicht. Werden die Größenunterschiede geringer wie beim Training von CTC mit KTH (Abbildung 7.6(d)), muss sich die trainierte CTC-Site vor einer Überflutung mit Fremdjobs seitens des KTH schützen, da die lokalen Kapazitäten eben nicht ausreichen, um weite Teile der KTH-Jobs zu übernehmen.

Daher werden kleine Jobs mit Ausnahme einer sehr geringen Lastsituation abgelehnt. Größere Jobs (NJP größer 10%) hingegen werden unabhängig von der Lastsituation lokal ausgeführt. Das ist zum einen sinnvoll, um den kleineren Partner nicht mit Jobs hoher Parallelität[23] zu verstopfen und zum anderen, um ihn durch die Abarbeitung seiner größeren Jobs zu unterstützen.

Die Ergebnisse zeigen darüber hinaus, dass die trainierten Regelbasen sehr stark abhängig von den jeweiligen Partnern sind, gegen die sie trainiert wurden. Um dies zu untersuchen, wurde ebenfalls überprüft, ob sich Regelbasen, die auf einer Site gelernt wurden, sowohl für den Einsatz in anderen Sites als auch für die Anwendung gegen eine andere Site eignen. Beides ist nur der Fall, wenn — bei Regelübertragung — die anwendende Site eine ähnliche Größe besitzt und auch der Partner — bei Einsatz gegen einen fremden Partner — eine ähnliche Größe besitzt, wie die Site, gegen die zuvor trainiert wurde.
Man kann demnach nicht einfach eine Regelbasis nehmen, die auf der KTH-Site gegen den SDSC05 gelernt wurde und anschließend im CTC gegen SDSC00 einsetzen. Auf eine detailliertere Darstellung dieser Untersuchungen soll an dieser Stelle verzichtet und auf die entsprechende Veröffentlichung [42] verwiesen werden.

Trotzdem gilt es im Folgenden, die Robustheit der erlernten Regelbasen im Hinblick auf die Übertragbarkeit auf neue Workloads für die gleiche Site-Konstellation zu untersuchen.

Robustheitsanalyse der trainierten Regelbasen

Das vorangehende Training von Regelbasen aus vergangenen Eingabedaten (Aufzeichnungen über fünf Monate) macht nur Sinn, wenn diese bei späteren Eingaben (die nächsten sechs Monate) ebenfalls zu einer Verbesserung der Scheduling-Qualität führen[24]. Um dies zu überprüfen, wurden alle paarweise erlernten Regelbasen auf Basis der neuen Eingaben und im Zusammenspiel mit den jeweils anderen Regelbasen des Trainingspartners getestet.

Bei dem paarweisen Kreuztest wurden insgesamt zehn Übertragungsexperimente durchgeführt, in denen jeder der beiden Partner seine Regelbasis einsetzt, die er im Rahmen der unabhängigen Trainingsläufe jeweils gegen den Anderen optimiert hat.

[23] 10% NJP beim CTC entsprechen 43% NJP beim KTH.
[24] Dies entspricht der Robustheit gegenüber neuen Eingabedaten.

Im Zuge der Übertragungsexperimente wird sich zeigen, dass erlerntes Verhalten bei Anwendung von FCFS auch in der Übertragung auf neue Eingabedaten vorteilhaft ist und alle Sites von der Teilnahme am Grid profitieren. Im Vergleich mit AWF kommt es in der Übertragung zudem zu einer gerechteren Umverteilung des Gewinns.

Die Übertragung der unter EASY gelernten Regelbasen wird jedoch eine Schwachstelle des Anpassungsprozesses der Trainings-Site offenlegen, so dass bei lokaler Anwendung von EASY in der Übertragung auf neue Daten stets der Einsatz von AWF zu empfehlen ist.

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
KTH			9	18	39	-10	36	-5	33	-17
SDSC00	60	-4			87	2	79	-4	80	-3
CTC	3	2	2	-2			5	15	15	0
SDSC03	2	0	5	2	21	-6			24	12
SDSC05	1	1	5	1	9	0	21	-8		

TABELLE 7.6: *Ergebnisse der paarweisen Anwendungsdurchläufe für alle fünf Workloads auf sechs Monaten Übertragungsdaten und Verwendung von FCFS als lokalen Scheduling-Algorithmus.*

Tabelle 7.6 zeigt die daraus resultierenden prozentualen Verbesserungen in der AWRT und Änderungen in der Arbeitslast (VRP) gegenüber den Resultaten bei ausschließlich lokalem Scheduling und Verzicht auf Jobaustausch. Unabhängig vom Setup profitiert jede Site auch in den folgenden Monaten von einer trainierten Regelbasis.

Besonders wichtig ist dabei, dass auch die größeren Sites durchgängig von der Teilnahme am Grid profitieren. So liegen die Verbesserungen der SDSC05-Site zwischen moderaten 1% AWRT-Verbesserung und (für eine so große Site sehr großen) 21% Verbesserung der AWRT. Bei letzterem Setup (SDSC03 mit SDSC05) kann sich auch die SDSC03-Site um 24% verbessern, obwohl sie 12% mehr Arbeitslast bearbeitet als lokal eingereicht wird. Dies ist ein deutlicher Hinweis darauf, dass ein gezielter Lastausgleich zwischen den Partnern bei Lastspitzen auf lange Sicht für beide von Vorteil ist.

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
KTH			32	17	18	-1	2	16	5	-6
SDSC00	48	-4			53	6	-33	3	-14	2
CTC	5	0	14	-5			-4	21	10	5
SDSC03	7	-1	11	-1	20	-7			26	10
SDSC05	-1	0	2	0	7	-1	15	-7		

TABELLE 7.7: *Ergebnisse der paarweisen Anwendungsdurchläufe für alle fünf Workloads bei Anwendung von FCFS und im Vergleich zur AWF-Performanz im gleichen Zeitraum und ebenfalls mit FCFS. Hierbei wurden demnach für die $AWRT_I$-Berechnungen die entsprechenden AWRT-Ergebnisse unter Anwendung von AWF verwendet. Um auch die VRP relativ darstellen zu können, wurden die Differenzen der VRP-Ergebnisse vom Lernverfahren und der Anwendung von AWF gebildet.*

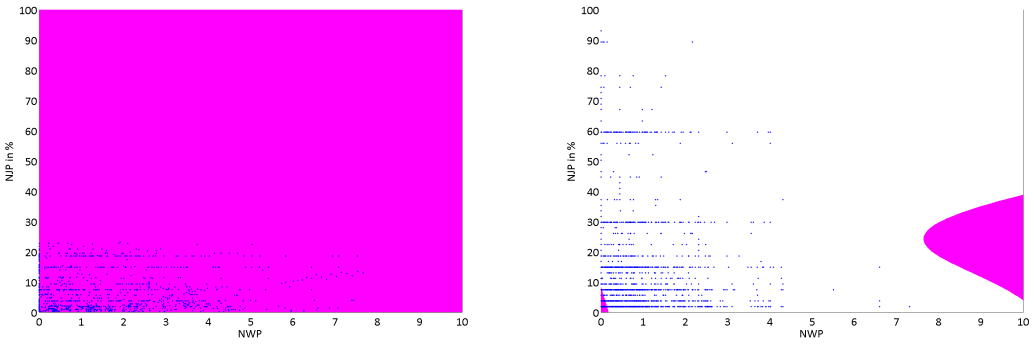
Tabelle 7.7 zeigt analog zu Tabelle 7.4 die Performanz des lernorientierten Ansatzes gegenüber der Anwendung von AWF. In 16 von 20 Paarungen profitieren beide Sites davon, wenn erlernte Regelbasen eingesetzt wurden. Obwohl zwar immer noch alle Sites besser sind als ohne die Aktivitätendelegation, kommt es in den übrigen vier Paarungen zu einer Umverteilung des Gewinns.

So ist die AWRT der SDSC00-Site bei Anwendung der Fuzzy-Regelbasis zwar 33% höher als bei Anwendung von AWF, doch führt dies zu einer Verbesserung der AWRT des Partners SDSC03 um 11%. Erst durch diese Verschiebung der Performanz haben am Ende beide vom Grid profitieren können (zur Erinnerung: In dem gleichen Setup von SDSC00 und SDSC03 bei sechs Monaten Simulationszeit mit AWF hat sich lediglich der SDSC00 verbessert).

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
KTH			5	-8	-3	-100	16	-39	17	-69
SDSC00	13	2			29	-10	37	-13	44	-12
CTC	-29	24	-3	10			14	-3	13	-10
SDSC03	-1	3	-3	5	-1	1			6	4
SDSC05	-6	4	-2	3	0	2	11	-3		

TABELLE 7.8: *Ergebnisse der paarweisen Anwendungsdurchläufe für alle fünf Workloads bei Anwendung von EASY als lokalen Scheduling-Algorithmus.*

Die in Tabelle 7.8 dargestellten Ergebnisse der Übertragungsexperimente der bei lokalem EASY-Backfilling trainierten Regelbasen zeigen eine Schwachstelle des Lernverfahrens auf. So ergeben sich in Experimenten, in denen größere Sites entweder gegen KTH oder gegen SDSC00 gelernt haben, auch Verschlechterung gegenüber dem lokalen Scheduling. Der Grund dafür fällt erst bei näherer Betrachtung der beim Training entstehenden Regelbasen auf.



(a) CTC mit KTH gelernt. (b) CTC mit SDSC03 gelernt.

ABBILDUNG 7.7: *Beispiele für Regelbasen bei der paarweisen Optimierung über ein evolutionäres Fuzzy-System. Alle gezeigten Regelbasen stammen von fünfmonatigen Trainingsdurchläufen mit EASY als lokalen Scheduling-Algorithmus. Während die farbigen Bereiche diejenigen mit einer positiven Regelkonsequenz darstellen, steht jeder der blauen Punkte für eine Anfrage an die Regelbasis.*

Abbildung 7.7 verdeutlicht das beim Training unter EASY-Backfilling auftretende Problem anhand von zwei optimierten Regelbasen. Beide sind beim Training der CTC-Site entstanden jedoch einmal mit KTH als Partner (Abbildung 7.7(a)) und in dem anderen Beispiel mit SDSC03 als Partner (Abbildung 7.7(b)).

Im ersten Fall (Abbildung 7.7(a)) erreicht die Site während ihres Trainings offenbar genau dann die beste AWRT, wenn sie alle eigenen und auch die vom Partner angebotenen Jobs selbst berechnet.

Im zweiten Fall (Abbildung 7.7(b)) liegt ein genau gegenteiliges Verhalten vor. Hier wird mit Ausnahme eines sehr kleinen Bereiches mit niedriger NWP (kleiner 0,2) und NJP (kleiner 10%) jeder externe Job abgelehnt und jeder lokale Job zunächst dem Partner angeboten.

In beiden Fällen führt das Verhalten dazu, dass die Entscheidungen, wo ein Job schließlich ausgeführt werden soll, nur noch von der Entscheidung des Trainingspartners abhängt, der selbst AWF als Transferstrategie einsetzt. Dies deutet darauf hin, dass AWF bei EASY für die zu trainierende Site bereits eine so gute Scheduling-Performanz erreicht, dass dies ein gezieltes Verhaltenstraining überflüssig macht. In beiden Fällen findet eine sehr einseitige Verschiebung des Workloads statt, die nur noch von dem AWF einsetzenden Trainingspartner beschränkt wird, da dieser — bei entsprechender Situation — externe Jobs immer noch ablehnt bzw. bei eigenen Jobs nur extern anfragt, wenn er den Job nicht selbst bearbeiten will.

In der Übertragung auf die 6-Monats-Traces ist die „Gleichgültigkeit“ der trainierten Sites fatal, da nun beide Partner eine Regelbasis einsetzen, welche das Gridweite Scheduling zuvor weitestgehend dem AWF-Partner überlassen hat. Dies führt zwangsläufig zu einer Verstärkung der einseitigen Workload-Verlagerung, wenn der kleinere der beiden Partner seine Jobs immer abgeben möchte und der größere der beiden alles lokal ausführt.

Das Resultat ist eine Überlastung der größeren Site, welche sich unter Umständen (vgl. Experiment KTH mit CTC in Tabelle 7.8) auch negativ auf den kleineren Delegationspartner auswirken kann, da seine abgegebenen Jobs zu viel Zeit in der Queue des Delegationspartners verbringen.

Der zuvor beschriebene kontraproduktive Anpassungsprozess findet offenbar nur bei Anwendung von EASY *während* des Trainings statt. Es besteht die Vermutung, dass die bei FCFS gelernten Regelbasen ein vorteilhaftes Delegationsverhalten für die Sites beinhalten, das während des Lernens mit EASY nicht erlernt wird.

Daher wurde zusätzlich noch der Versuch unternommen, die zuvor mit FCFS trainierten Regelbasen auf Robustheit bzgl. des lokalen Scheduling-Algorithmus zu testen.

Tabelle 7.9 zeigt die daraus resultierenden Ergebnisse der Übertragungsexperimente. Für den überwiegenden Teil der Experimente führt die Anwendung durchaus zu positiven Ergebnissen gegenüber „Nicht-Teilnahme“ am Grid.

Lediglich in zwei Fällen (KTH mit SDSC00 und KTH mit SDSC05) verschlech-

	KTH		SDSC00		CTC		SDSC03		SDSC05	
	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP	$AWRT_I$	VRP
KTH			-18	25	15	-11	20	-15	18	-30
SDSC00	20	-6			32	3	35	-4	35	-6
CTC	1	3	3	-3			5	16	16	-7
SDSC03	0	1	1	2	7	-6			3	11
SDSC05	-2	2	2	1	0	2	12	-7		

TABELLE 7.9: *Ergebnisse der paarweisen Anwendungsdurchläufe für alle fünf Workloads. Die Regelbasen wurden zuvor mit FCFS gelernt und nun mit EASY angewandt.*

tert sich einer der beiden Partner. Im ersten Fall ist dies der großen Bereitschaft des KTH zur Übernahme von Arbeitslast geschuldet (25% der lokalen Arbeitslast wird dabei mehr vom SDSC00 übernommen als umgekehrt.)

Das Versagen im zweiten Fall hingegen stellt keine wirkliche Besonderheit dar, da der viel größere SDSC05 es sowieso sehr schwierig hat, von einer Kooperation mit dem KTH zu profitieren und auch bei Anwendung von AWF und EASY auf den Übertragungsdaten von sechs Monaten keine Verbesserung gegenüber der lokalen Ausführung erreicht (vgl. Tabelle 7.1 auf S. 68).

Aus den gemachten Beobachtungen lassen sich mehrere Schlüsse ziehen. Der wichtigste ist wohl, dass sich das vorgestellte Lernverfahren nicht zielführend verhält, wenn auf LRMS-Ebene während des Trainings EASY eingesetzt wird. Beim Training mit FCFS ergeben sich Regelbasen, die auch bei einem Wechsel des LRMS-Algorithmus von FCFS zu EASY für den Übertragungszeitraum immer noch in vielversprechenden Verbesserungen resultieren.

Im Sinne eines für alle Partner im Grid vorteilhaften Austauschs (also möglichst auch vorteilhaft für größere Sites) hat sich bei den Übertragungsexperimenten mit FCFS eindeutig gezeigt, dass das Lernverfahren eine bessere Performanz liefert als die einfache Anwendung von AWF.

Bei den Übertragungsexperimenten mit EASY-Backfilling kann durch Anwendung FCFS-gelernter Regelbasen in den meisten Fällen zwar ebenfalls eine Verbesserung gegenüber rein lokalem Scheduling erreicht werden, jedoch ist die erreichte Performanz gegenüber AWF in annähernd genau so vielen Fällen schlechter wie besser. Da der Lernansatz gegenüber der Heuristik einen viel größeren Aufwand (paarweises Lernen von Regelbasen) besitzt, lohnt sich dieser — angesichts der Ergebnisse für FCFS — nur bei dem Fehlen von Laufzeitschätzungen im System.

7.2.5 Effiziente Implementierung einer Lokationsstrategie

Ungeachtet der verwendeten Transferstrategie wird mit Vergrößerung des Grids auf mehr als zwei Sites auch eine Lokationsstrategie nötig, welche bei jeder lokalen Jobeinreichung bestimmt, mit welcher Priorität entfernte Partner zwecks Übernahme des Jobs befragt werden.

Hierbei sollte nach Möglichkeit die Site zuerst gefragt werden, welche auch den größten erwarteten Nutzen bei Abgabe des Jobs verspricht. Die Berechnung dieses

erwarteten Nutzens muss aufgrund der restriktiven Informationsstrategie wie bei der Transferstrategie ebenfalls nur auf Basis lokal berechenbarer Größen erfolgen.

Da die Transferstrategie bereits hinsichtlich der AWRT optimiert wurde und ein Delegationspartner aus Sicht einer Site insbesondere durch schnelle Abarbeitung der delegierten Jobs an Nutzen gewinnt, wurde für die Realisierung der Lokationsstrategie auf die AWRT delegierter Jobs zurückgegriffen.

Existieren im Grid mehrere potentielle Delegationspartner, so berechnet sie für jeden dieser Partner separat eine AWRT. Diese wird im Gegensatz zur ursprünglichen Definition weder auf der Menge lokal eingereichter Jobs der Site (τ_k) noch auf der Menge der berechneten Jobs (π_k) ermittelt, sondern lediglich auf der Menge von Jobs, welche die Site zuvor zu eben jenem Partner delegiert hat.

Bei dem Aufruf der Lokationsstrategie werden die Sites aufsteigend nach ihrer delegierten AWRT sortiert, so dass zunächst der Partner mit der geringsten Ausprägung gefragt wird, da diese nach der vergangenen Erfahrung die schnellste Abarbeitung von Jobs verspricht.

Der Vollständigkeit halber sei noch erwähnt, dass auch die AWRT abgegebener Jobs mit zunehmender Menge betrachteter Jobs unempfindlich gegenüber Änderungen der AWRT einzelner Jobs wird. Um aber trotzdem auf Änderungen der Servicequalität von Delegationspartnern reagieren zu können, wurde die Metrik auf die letzten 50 abgegebenen Jobs gefenstert[25].

Die Anwesenheit mehrerer Partner im Grid hat in unserem Fall (paarweises Training von Fuzzy-Regelbasen) allerdings nicht nur den Bedarf nach einer Lokationsstrategie zur Folge. Darüber hinaus muss auch sichergestellt werden, dass für die Kommunikation mit jedem Partner (Transferstrategie) auch die korrekte — zuvor trainierte — Regelbasis eingesetzt wird, da die gelernten Regelbasen sich, wie zuvor in Abschnitt 7.2.4 beschrieben, in Abhängigkeit der Größe der Trainings-Site bzw. des Trainingspartners sehr stark unterscheiden.

Bei Untersuchung der Lokationsstrategie war es interessant, wie sich die Performanz der einzelnen Sites mit zunehmender Größe des Grids in Bezug auf lokales Scheduling sowie im Vergleich zur Grid-weiten Anwendung von AWF entwickelt. Zudem war zu untersuchen, in welchem Maße Arbeitslastverschiebungen von welcher zu welcher Site einer Grid-Konstellation stattfinden und wie sich diese ändern, wenn durch Vergrößerung des Grids bei jeder Austauschentscheidung mehr Verhandlungspartner verfügbar sind.

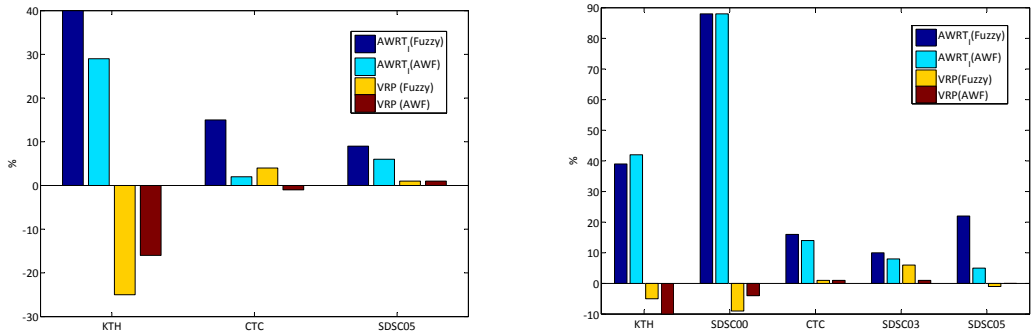
Im Rahmen dieser Forschungsfragen hat sich herausgestellt, dass sich die An-

[25]Dieser Wert wurde in mehreren Stichproben als vorteilhaft ermittelt, da ein sehr kleiner Wert wie z.B. 10 aufgrund der starken Schwankungen der AWRT zu Ergebnissen führt, die vergleichbar mit Zufallsexperimenten sind. Ein sehr großer Wert z.B. jenseits von 100 Jobs führt zu dem gleichen Effekt wie beim Verzicht auf ein Fenster — die unflexible Konvergenz gegen einen bestimmten Delegationspartner.

wendung erlernten Verhaltens auch in größeren als paarweisen Setups lohnt und die erreichte Performanz der einzelnen Sites in etwa den Resultaten der besten Trainingsdurchläufe entspricht. Mit Zunahme der Parteien im Grid kommt es zudem automatisch zu einer gleichmäßigeren Verteilung der Last bei gleichzeitigem Erhalt des zuvor angesprochenen Performanzverhältnisses.

Die im Vergleich zu AWF verhältnismäßig schlechte Übertragung von FCFS-gelernten Regelbasen auf den Übertragungszeitraum unter Anwendung von EASY warf außerdem die Frage auf, ob sich das aufwendige Lernverfahren erst mit der Vergrößerung des Grids rentiert.

Die nachfolgenden Experimente haben in diesem Zusammenhang gezeigt, dass sich der Mehraufwand für das Lernen von Regelbasen bei lokaler Anwendung von EASY erst recht dann nicht lohnt, wenn eine Übertragung der erlernten Regelbasen auf größere Grids vorgenommen wird.



(a) Grid mit drei Teilnehmern.

(b) Grid mit allen fünf Teilnehmern.

Abbildung 7.8: *Zwei Beispiele für Grids mit mehreren Teilnehmern bei Anwendung der zuvor paarweise gelernten Regelbasen und AWF im Vergleich. In allen Konfigurationen wurde FCFS als lokaler Scheduling-Algorithmus genutzt.*

Abbildung 7.8 zeigt die prozentualen Ergebnisse für die AWRT_I und die Änderung in der bearbeiteten Arbeitslast (VRP) gegenüber der lokalen Ausführung der Workloads. Gegenübergestellt sind jeweils die Ergebnisse für die Anwendung der Fuzzy-Regelbasen und dem Grid-weiten Einsatz von AWF als Transferstrategie bei den jeweiligen Übertragungsexperimenten auf sechs Monaten.

In dem 3-Site-Grid (Abbildung 7.8(a)) zeigt sich ein eindeutiger Performanzvorteil des Lernansatzes gegenüber AWF. Hier profitieren alle drei Sites während der sechsmonatigen Simulation von den zuvor erlernten Regelbasen. Die KTH-Site erreicht eine AWRT_I von 39% bei der Anwendung von Fuzzy-Regeln gegenüber 29% bei Anwendung von AWF.

Der CTC erreicht mit einer AWRT-Verbesserung um 15% gegenüber lokaler Ausführung eine viel größere Verbesserung als mit der AWF-Heuristik und auch die SDSC05-Site kann sich mit 9% AWRT_I noch um weitere drei Prozentpunkte gegenüber der AWF-Performanz steigern.

Im größeren Grid (Abbildung 7.8(b)) verliert lediglich die KTH-Site 2 Prozentpunkte Performanz bei Anwendung trainierter Regelbasen gegenüber AWF, was sich dadurch erklären lässt, dass die größeren Sites im gelernten Austausch mit der KTH-Site geschickter zu ihrem eigenen Vorteil vorgehen.

Beim Vergleich mit den Resultaten der paarweisen Anwendungsexperimente (vgl. Tabelle 7.7 auf S. 83) fällt auf, dass die hier erreichten $\mathrm{AWRT_I}$-Werte annähernd den Maximalwerten der paarweisen Übertragung der beteiligten drei Sites entsprechen. So erreichte der KTH im 2-Site-Grid mit dem CTC eine $\mathrm{AWRT_I}$ von 39%, der CTC mit dem SDSC05 15% und der SDSC05 mit dem CTC 9%.

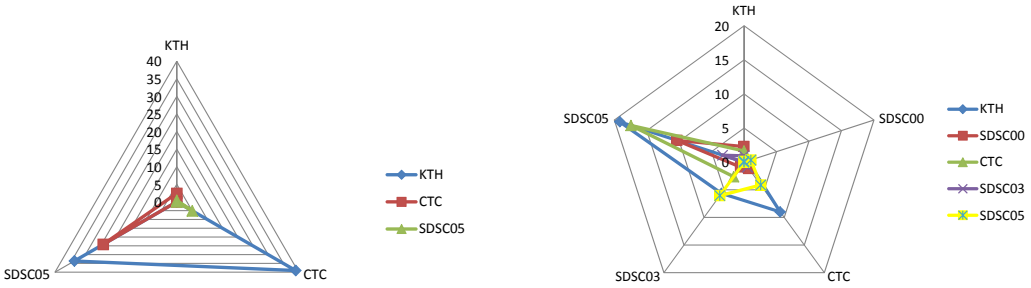
Dieses Phänomen setzt sich mit Ausnahme der neu hinzukommenden SDSC03-Site auch bei der Vergrößerung des Grids auf fünf Sites (Abbildung 7.8(b)) fort. Auch hier erreichen alle übrigen Sites das Maximum an $\mathrm{AWRT_I}$ der paarweisen Übertragungen.

Die SDSC03-Site, welche in der paarweisen Simulation mit dem SDSC05-Workload auf 24% AWRT-Verbesserung kommt, schafft in dem größeren Grid immerhin noch 10% Verbesserung gegenüber lokaler Ausführung.

Bemerkenswert ist allerdings, dass die annähernd gleichbleibende Performanz von KTH und CTC bzw. die sogar noch gesteigerte Performanz des SDSC05 bei Erweiterung des Grids von drei auf fünf Teilnehmer mit einer gleichmäßigeren Verteilung der Last einhergeht.

Konnte die KTH-Site ihre 40% $\mathrm{AWRT_I}$ im kleineren Grid nur über Mehrabgabe von 25% der eigenen Last (VRP $= -25$) erreichen, so ist eine annähernd gleichgroße Performanz ($\mathrm{AWRT_I} = 39\%$) im größeren Grid auch mit einer VRP von -5% möglich.

Darüber hinaus profitieren vor allem die größeren Sites (CTC, SDSC03 und SDSC05) von einem vorherigen Lernen der Regelbasen gegenüber dem Einsatz von AWF, wohingegen der KTH einen Prozentpunkt an Performanz verliert.



(a) Grid mit drei Teilnehmern.

(b) Grid mit allen fünf Teilnehmern.

Abbildung 7.9: *Prozentuale Arbeitslastverschiebung zwischen den Delegationspartnern in % für beide Lokationsstrategie-Experimente bei Anwendung der zuvor erlernten Regelbasen im 6-Monats-Übertragungsexperiment. Die gezeigte Verschiebung entspricht nicht der VRP der Sites sondern schlüsselt die prozentuale Abgabe an einzelne Partner auf.*

Abbildung 7.9 visualisiert die — bei der Simulation der vergrößerten Grids durchgeführten — Arbeitslastverschiebung im Detail.

So verteilt der KTH in Abbildung 7.9(a) insgesamt ca. 73% seiner lokalen Arbeitslast auf den CTC (39%) und den SDSC05 (34%). Die Umverteilung der beiden anderen Sites fallen weitgehend einseitiger aus, da die KTH-Site wenig Fremdlast aus dem Grid übernimmt.

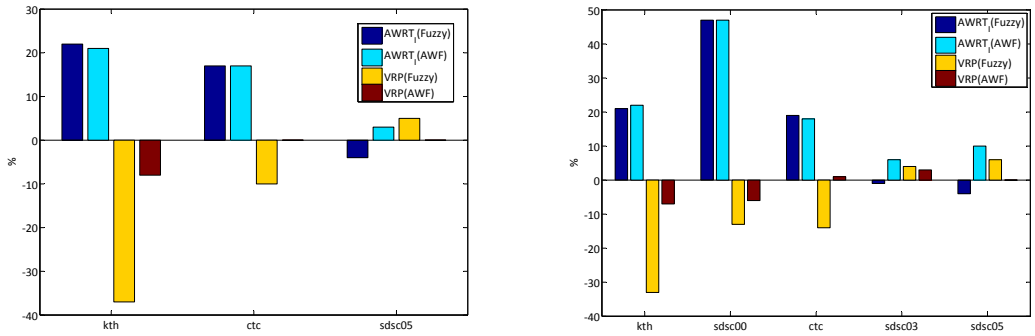
Deswegen tauschen CTC und SDSC05 mit 24% (CTC→SDSC05) und 5% (SDSC05→CTC) in erster Linie miteinander anstatt mit dem KTH.

Dieser wenig überraschende Trend lässt sich auch in dem 5-Site-Setup in Abbildung 7.9(b) nachvollziehen, bei dem sich für die SDSC05-Site in erster Linie die nächstkleineren Partner SDSC03 und CTC als bevorzugte Delegationspartner darstellen. Die SDSC05-Site selbst wiederum dient allen anderen Partnern als bevorzugter Tauschpartner, da diese zwischen 19% (KTH) und 3% (SDSC03) ihrer eigenen Last an die SDSC05-Site delegieren.

Obwohl der KTH annähernd die gleiche AWRT-Performanz erreicht wie noch beim 3-Site-Grid, delegiert er in der Summe nun nicht mehr 73% sondern lediglich 34% der eigenen Last. Dies deutet deutlich auf die stark schwankende Last dieser Site und die damit verbundenen Vorteile durch den Lastaustausch hin.

Je mehr Delegationspartner sich im Grid befinden, desto wahrscheinlicher ist es, dass die Site ihre Jobs bei Hochlast auch delegieren kann, was automatisch zu einer Lastreduktion und damit rückgekoppelt zu einer Reduzierung weiterer Delegationsversuche führt.

Bei der Übertragung der FCFS-gelernten Regelbasen auf den Anwendungszeitraum mit EASY-Backfilling als lokalen Scheduling-Algorithmus fällt auch bei den größeren Setups auf, dass AWF für den Austausch von vornherein die bessere Wahl darstellt.



(a) Grid mit drei Teilnehmern.

(b) Grid mit allen fünf Teilnehmern.

Abbildung 7.10: *Zwei Beispiele für Grids mit mehreren Teilnehmern bei Anwendung der zuvor mit FCFS paarweise gelernten Regelbasen und AWF im Vergleich. In allen Anwendungsexperimenten wurde EASY-Backfilling als lokaler Scheduling-Algorithmus genutzt. In der Darstellung wurden alle Größen in Relation zu den Ergebnissen bei lokalem Scheduling berechnet.*

Abbildung 7.10 zeigt die zuvor mit FCFS erlernten Regelbasen in ihrer Performanz innerhalb der beiden größeren Grids und unter Anwendung von EASY. Zwar

weisen die Simulation in Einzelfällen (z.B. für den KTH im 3-Site-Grid oder den CTC im 5-Site-Grid) noch geringe Performanzsteigerungen gegenüber AWF auf, jedoch ist auch in diesem Fall der Mehraufwand für das Lernen nicht gerechtfertigt. Schon allein deswegen, weil insbesondere die größeren Sites (SDSC03 und SDSC05) bei Anwendung der Fuzzy-gelernten Regelbasen schlechter werden als lokal.

7.2.6 Coevolutionäres Lernen von Regelbasen

Der bisherige Lernansatz mit klassischem EA hat sowohl ein konzeptionelles als auch ein praktisches Problem. Konzeptionell kritisch ist, dass das Verhalten einer Site während des paarweisen Lernens stets gegen einen mit AWF statisch konfigurierten Lernpartner optimiert wird.

Einerseits ist diese unveränderliche Umwelt vorteilhaft für den stetigen Lernfortschritt. Andererseits wird das Verhalten einer Site dadurch in erster Linie gegen das AWF-Verhalten des Partners optimiert, welches nur im begrenzten Umfang das spätere Verhalten desselben Partners während der Robostheitsübertragung widerspiegelt, das dieser zuvor separat erlernt hat. Dieser Effekt konnte vor allem bei der Optimierung von Regelbasen unter lokaler Anwendung von EASY beobachtet werden.

Das praktische Problem liegt in der hohen Anzahl von Lerndurchläufen begründet. So benötigte das in Abschnitt 7.2.5 evaluierte 5-Parteien-Grid zehn unabhängige Trainingsdurchläufe, damit jede Site eine Regelbasis für die Transferstrategie mit jeder anderen Site besaß.

Motivation für das nachfolgende Lernkonzept, welches auch bereits im Rahmen von gemeinsamen Forschungsarbeiten durch Fölling et al. [41, 46, 45] veröffentlicht wurde, ist zum einen die Beschleunigung des Lernverfahrens und zum anderen, das Erlernen von Regelbasen unter wechselnden Umweltbedingungen zu untersuchen.

Hierzu wird statt dem paarweisen Lernen von Regelbasen in einer Evolutionsstrategie ein *Coevolutionärer Algorithmus* verwendet. Gegenüber rein evolutionären Algorithmen unterscheiden sich jene in erster Linie dadurch, dass sie anstatt einer einzelnen Population mit Individuen, die jeweils eine komplette Lösung des Problems beinhalten, mehrere Populationen gleichzeitig optimieren.

Die Fitnessberechnung eines einzelnen Individuums hängt dabei nicht mehr nur von den Parametern eines einzelnen Individuums ab, sondern ergibt sich in der Interaktion mit Individuen anderer Populationen.

Weiterhin lassen sich Coevolutionäre Algorithmen nach Paredis [98] grundsätzlich in zwei verschiedene Klassen unterteilen — *Kompetitiv* oder *Kooperativ*.

Der Unterschied zwischen den beiden Klassen liegt in der Fitnessberechnung der Individuen während des Evolutionsprozesses. Bei kooperativen COEAs lässt sich ein Gesamtproblem in mehrere Unterprobleme (Spezies mit eigener Population) unterteilen. Diese werden unabhängig voneinander optimiert. Die Fitness eines Individuums entspricht der Fitness der Gesamtlösung bei Interaktion mit (wenigen) Vertretern der übrigen Spezies.

Grundsätzlich empfiehlt sich dieses Verfahren nach Potter & De Jong [100] für Optimierungsprobleme, bei denen sich verbesserte Lösungen der Einzelpopulatio-

nen grundsätzlich auch positiv auf die übrigen Populationen auswirken.

Mit kompetitiven COEAs hingegen werden in erster Linie konkurrierende Spezies mit antiproportionaler Fitness optimiert. Hierbei führt die Verbesserung der einen Spezies in der Regel zur Verschlechterung einer anderen. Die Fitnessberechnung der Individuen innerhalb einer Spezies erfolgt dabei in Form von Turnieren gegen Vertreter anderer Spezies. Die Auswahl der entsprechenden Turnierpartner kann hierbei entweder nach gezielten Kriterien oder auch rein zufällig erfolgen.

Bei unserem Problem (Optimierung von Regelbasen in einem dezentralen Grid) verfolgen die einzelnen Sites in erster Linie egoistische Ziele. Sie möchten durch die Teilnahme am Grid den Service für ihre eigene Nutzergemeinschaft steigern.

Die vorherigen Ergebnisse haben auf der einen Seite gezeigt, dass dieser Egoismus in vielen Fällen dazu führt, dass sich einige Sites (mit weniger lokalen Ressourcen) auf Kosten der Grid-Partner (mit mehr lokalen Ressourcen) verbessern.

Auf der anderen Seite gibt es aber auch einen kooperativen Faktor, bei dem kleinere Sites von einem guten Scheduling auf den größeren Partnern profitieren und selbst schlechter werden, wenn sie den Grid-Partner durch zu viel Auslagerung von Arbeitslast überfordern.

Maßgeblich für die Fitnessberechnung einer Regelbasis ist stets nur die egoistische Leistung für die eigene Nutzergemeinschaft. Dieses Prinzip soll auch bei der Modellierung einer Coevolutionären Optimierung beibehalten werden. Daher erfolgt diese in Form eines kompetitiven COEAs.

Training von Regelbasen

In der Kodierung von Regeln und Regelbasen unterscheidet sich der COEA nicht von dem zuvor diskutierten klassischen EA. Auch die Rekombination und Variation von Individuen einer Spezies erfolgt nach den gleichen Regeln und Parametern wie bereits in Abschnitt 7.2.4 erläutert.

Es sei zudem darauf hingewiesen, dass die Spezies im Hinblick auf ihre Fortentwicklung voneinander isoliert sind. Lediglich die Fitnessberechnung erfolgt in Form der turnierorientierten Interaktion mit den jeweiligen anderen Spezies.

Zu Beginn der Optimierung wird nun nicht eine einzige Population von μ Individuen erzeugt sondern jeweils eine Population für jede Site, die an der Optimierung des Grids mit insgesamt K Sites teilnimmt. Während bei der ersten Generation noch zufällige Turniertupel gebildet werden, erfolgt die Bildung in den späteren Generationen auf Basis der lokalen Fitness.

Wie bereits bei dem klassischen EA bilden die μ besten Individuen (größte Fitness bzw. kleinste AWRT) innerhalb jeder Spezies die Elternpopulation. Diese Eltern müssen sich nun allerdings in einer sich verändernden Umwelt behaupten.

Wie in Abbildung 7.11 dargestellt, bilden die Eltern in der Reihenfolge ihrer

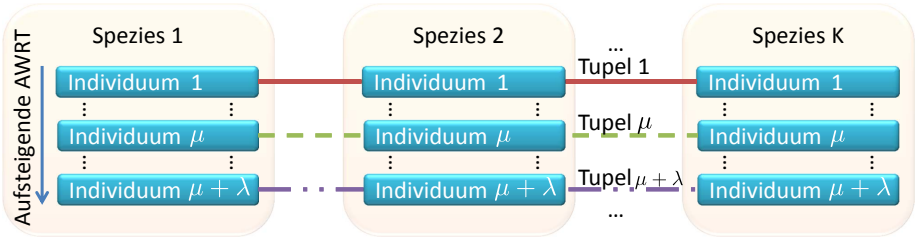


ABBILDUNG 7.11: *Turnierplan für die Fitnessberechnung der Spezies.*

Fitness zunächst μ Tupel. Durch die Bildung von Turnierpaaren mit vergleichbar guten Individuen (die Besten aus jeder Population gegeneinander, die Zweitbesten gegeneinander, usw.) wird der Wettbewerbsdruck auf die Elternindividuen erhöht. So können sich auf lange Sicht nur Individuen in einer Population halten, die sich nicht nur gegen einen sondern viele unterschiedliche Grid-Partner durchsetzen können. Lösungen, die zu vergleichsweise egoistischem Verhalten führen, stellen häufig zwar die beste Lösung innerhalb ihrer Population dar, sterben allerdings nach der Simulation mit egoistischen Lösungen der anderen Populationen.

Die übrigen λ Tupel werden aus den Kindern gebildet und führen sowohl zur Fortentwicklung einer jeden Population als auch zur Erprobung unterschiedlicher Grid-Paarungen.

Da während der Coevolution meist mehr als zwei Sites gleichzeitig trainiert werden, ist auch die Lokationsstrategie Teil der Optimierung und basiert ebenfalls auf der AWRT abgegebener Jobs, wie in Abschnitt 7.2.5 erläutert.

Evaluation und Vergleich mit klassischem EA

Bei der Evaluation des coevolutionären Ansatzes hat sich gezeigt, dass er trotz geringerem Lernaufwand zu einer vergleichbar guten Performanz für die einzelnen Sites führt wie das klassische paarweise Training der Parteien. Zusätzlich bietet er auch im Nachhinein die Möglichkeit zur Auswahl von Regelbasen, die im Sinne eines kooperativen Grids zu gerechteren Gewinnverteilungen führen können. Diese Resultate werden im Folgenden näher erläutert.

Während sich die Fitness eines Individuums beim klassischen EA aufgrund der unveränderlichen Umwelt zwischen zwei Durchläufen nicht unterscheidet[26], ist dies beim COEA aufgrund der wechselnden Regelbasen der Trainingspartner sehr wohl der Fall.

Dies resultiert in einer vergleichsweise starken Fluktuation der Elternpopulation einer Spezies. Es werden ständig neue Eltern selektiert, sterben wieder aus oder werden durch Variation weiterentwickelt.

Im Gegensatz zum klassischen EA ergibt sich die beste gefundene Lösung daher nicht einfach am Ende der Optimierung mit dem besten Individuum. Je nach betrachtetem Kriterium ergibt sich eine gute Regelbasis (hinsichtlich egoistisch

[26] Sofern das Individuum selbst nicht verändert wird.

kompetitivem aber auch kooperativem Verhalten) ebenfalls im Laufe früherer Generationen, wird jedoch aufgrund der Turnierbildung und dem sich über die Zeit verändernden Selektionsdruck verlernt.

Daher werden *nach* coevolutionärer Optimierung über 150 Generationen nicht einzelne Individuen mit egoistisch maximaler Fitness gesucht, sondern vorteilhafte Paarungen von Regelbasen, die im Zusammenspiel eine gute Grid-weite Performanz erreicht haben.

Bei Untersuchung verschiedener Kriterien für diese Lösungsselektion haben sich insbesondere zwei Kriterien als recht vorteilhaft erwiesen und werden daher im Folgenden näher diskutiert.

Bei dem ersten Auswahlkriterium werden alle Paarungen hinsichtlich ihrer kumulativen Fitness untersucht. Ein Tupel von Regelbasen ist also dann besonders gut, wenn die Summe der AWRT-Werte aller Sites möglichst gering ist[27].

Da insbesondere kleinere Sites stärker von ihrem egoistischen Verhalten profitieren können, wird die kumulative Fitness im zweiten Ansatz über gewichtete AWRT-Werte der Sites gebildet. Als AWRT-Gewichtung wurde in diesem Fall die Site-Größe M_k einer Site k genutzt. Auf die Art und Weise fallen Verbesserungen auf einer größeren Site stärker ins Gewicht als Verbesserung einer kleinen Site.

Für einen aussagekräftigen Vergleich dieser beiden Ansätze und des COEA mit dem klassischen EA sowie dem Austausch von Jobs über AWF wurden alle zufallsbasierten Lernverfahren in fünf Wiederholungen für das 3-Site-Setup (KTH, CTC und SDSC05) ausgeführt und in der Übertragung auf die sechsmonatige Simulationszeit gegenübergestellt.

Abbildung 7.12 zeigt die Ergebnisse dieses Vergleichs hinsichtlich der prozentualen AWRT-Verbesserung. Neben den Werten für die einzelnen Läufe sind auch die arithmetischen Mittel aller Läufe pro Site und pro Ansatz eingezeichnet.

Bei keinem der Läufe ist die AWRT einer Site schlechter geworden als unter rein lokaler Ausführung. Gleichzeitig hat kein klassischer EA-Lauf schlechter abgeschnitten als AWF. Besonders die CTC-Site profitiert stark von dem Lernverfahren, da sie sich mit Verbesserungen zwischen 10% bis 14% relativ gesehen viel stärker gegenüber der AWF-Performanz verbessern kann als die anderen Sites.

Beim coevolutionären Verfahren treten je nach Site und Auswahlkriterium stärkere Schwankungen in der Performanz auf. Mitunter führt dies in seltenen Fällen auch zu einem schlechteren Ergebnis für eine Site als bei AWF.

Dies betrifft z.B. die Läufe vier und fünf unter der gewichteten kumulativen Fitness. Hierbei erzielt die KTH-Site nur noch 18% und 28% $\mathrm{AWRT_I}$, da in diesem Fall Lösungstupel gewählt wurden, die durch die starke Gewichtung der

[27] Es sei an dieser Stelle noch einmal darauf hingewiesen, dass diese kumulative Fitness erst nach Abschluss der Optimierung berechnet wird und keinen Einfluss auf die Selektion der Individuen während der Optimierung hat.

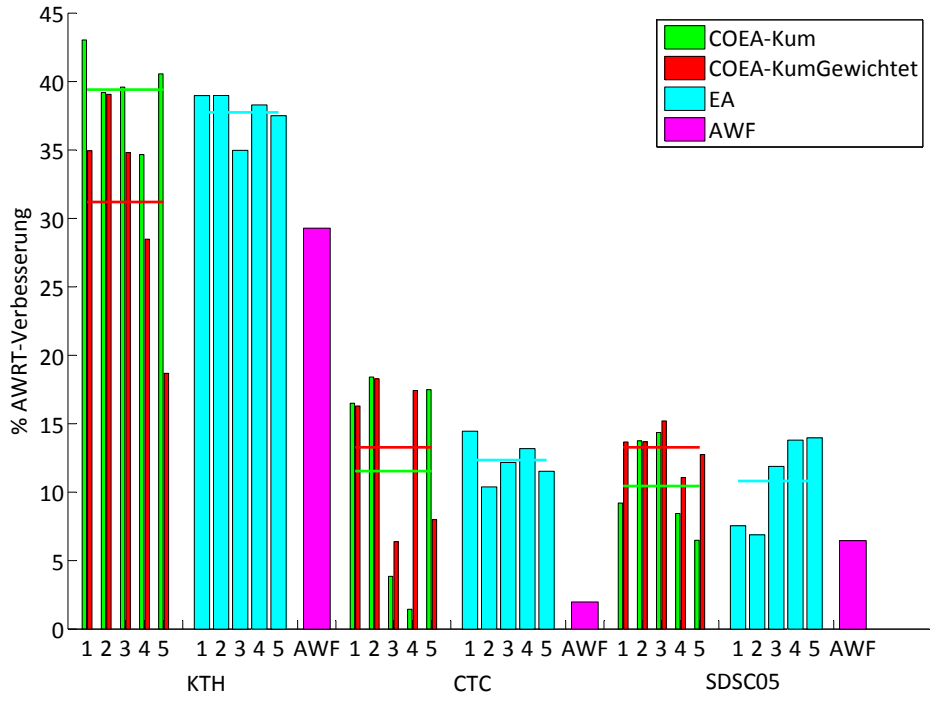


ABBILDUNG 7.12: *Prozentuale AWRT-Verbesserungen für alle drei Verfahren COEA, EA und AWF im Vergleich bei Übertragung der gelernten Regelbasen auf sechs Monate und bei Anwendung von FCFS. In dieser Abbildung wurden die $AWRT_I$-Ergebnisse relativ zur lokalen Ausführung berechnet.*

SDSC05-Fitness in erster Linie für diese Site von Vorteil sind.

Bei Gegenüberstellung der beiden Tupel-Auswahlkriterien — der kumulativen Fitness und ihrer gewichteten Form — zeigt sich, dass die Gewichtung für eine kleine Site wie KTH zu einer Verschlechterung der Performanz bei gleichem COEA-Lauf führt, wohingegen die größte SDSC05-Site davon profitiert. Die Auswahl gewichteter Lösungen motiviert also vor allem größere Sites, sich an dem Grid zu beteiligen.

Für die mittelgroße CTC-Site kann diesbezüglich keine Aussage gemacht werden, da es mit den COEA-Läufen 4 und 5 zwei Läufe gibt, in denen einmal die gewichtete Auswahl zu einem sehr viel besseren Ergebnis gegenüber der ungewichteten Auswahl führt und einmal genau umgekehrt. Arithmetisch über alle Läufe gemittelt liegt aber auch hier die gewichtete Lösung vorn.

Der Vergleich zwischen COEA und klassischem EA wird dadurch erschwert, dass die COEA-Lösungen während der Übertragung zwischen den Läufen viel stärker in ihrer Performanz schwanken als die EA-Lösungen. Dabei sind die COEA-

Lösungen in insgesamt 11 Fällen besser als die beste erreichte EA-Lösung der gleichen Site[28]. In 7 Fällen sind sie allerdings schlechter als die schlechteste EA-Lösung.

Auch hier hilft der Blick auf den Mittelwert der einzelnen Läufe. Für jede Site ist eines der Auswahlverfahren des COEA statistisch gesehen besser und das jeweils andere schlechter als die klassischen EA-Läufe im Mittel. Es ist wenig überraschend, dass es sich dabei immer genau um eben jenes Auswahlverfahren handelt, welches eher kleinere Sites bevorzugt (ungewichte Summe) oder größere Sites (gewichtete Summe). Es kann also nicht generell gefolgert werden, dass eines der beiden Verfahren COEA oder EA besser ist als das andere. Vielmehr liegen die Unterschiede der beiden Verfahren im Lernen selbst.

Beim klassischen EA muss jede Site — allerdings unabhängig von den anderen Partnern — ein egoistisches paarweises Lernen gegen alle Partner vornehmen. Bis auf den Austausch der Lern-Workloads im Vorfeld einer Kooperation ist dazu nichts weiter notwendig.

Beim COEA hingegen reicht eine einzige Optimierung aus, die zwar mit der Größe des Grids in ihrer Komplexität steigt, aber trotzdem auf diesen einen Lauf beschränkt ist. Zudem kann durch die Auswahl eines bestimmten Lösungstupels nach Optimierung noch Einfluss darauf genommen werden, welche Sites in stärkerem Maße von der Kooperation profitieren sollen. Dies setzt aber voraus, dass die Delegationspartner prinzipiell nicht nur ihre egoistischen Ziele verfolgen, sondern auch an einer fairen Kooperation interessiert sind.

Während die klassische EA-Lösung sich durch hohe Stabilität in der Übertragung auszeichnet, liegen die Vorteile des COEA vor allem in der sehr geringen und besser skalierenden Rechenzeit für die Optimierung.

Nichtsdestotrotz zeichnen sich beide Verfahren dadurch aus, dass die Rechenzeit im praktischen Einsatz der zuvor trainierten Regelbasen nicht ins Gewicht fällt, da innerhalb der Transferstrategie pro Job lediglich ein Konsequenzwert der Regelbasis ausgerechnet werden muss. Die Regelbasen selbst bleiben statisch.

7.3 Abschluss der Aktivitätendelegation

Im vorangegangenen Kapitel wurden algorithmische Verfahren zur Ressourcenerweiterung über Aktivitätendelegation in dezentralen Computational Grids unter restriktiven Informationsbedingungen untersucht.
Dazu wurde im Vorfeld auf die aktuellen Forschungen in diesem Bereich eingegangen und die Stärken und Schwächen der beschriebenen Ansätze diskutiert.

In der anschließenden Beschreibung eigener Ansätze zur Aktivitätendelegation wurde das Basis-Maschinenmodell erweitert um Modellannahmen für Computational Grids (Homogenität von Ressourcen, Kommunikationszeiten) sowie zusätzliche

[28] Bei der Zählung wurden beide Auswahlverfahren des COEA zusammengezählt.

Bewertungsgrößen für den Austausch von Arbeitslast zwischen autonomen Sites.

Mit einer AD-Strategie auf Basis von Transfer- und Lokationsstrategie wurde die Entscheidungsfindung eines jeden GRMS einer Site näher beschrieben. Infolgedessen wurden mit Accept When Fit (AWF), Fuzzy-EA und Fuzzy-COEA drei Realisierungen für Transferstrategien vorgestellt und sowohl in paarweisen Grids als auch (insbesondere für den COEA-Ansatz) im Zusammenspiel mit einer AWRT-basierten Lokationsstrategie in größeren Grids miteinander verglichen.

Hierbei hat sich herausgestellt, dass die Performanz der drei Verfahren abhängig ist von dem unterliegenden verwendeten lokalen Scheduling-Algorithmus (FCFS/EASY), obwohl keines der Verfahren gezielt auf Laufzeitschätzungen zurückgreift. Wenn FCFS eingesetzt wird (z.B. beim Fehlen von Laufzeitschätzungen), eignen sich in erster Linie die durch die beiden EA-Verfahren optimierten Fuzzy-Regelbasen zur Realisierung des Austauschverhaltens einer Site. Beim Einsatz von EASY hingegen wird die direkte Verwendung von AWF für den Austausch empfohlen.

Ein weiterer wichtiger Unterschied zwischen den Verfahren liegt in dem Aufwand zur Bestimmung des Verhaltens. Während AWF als einfache Heuristik keines Lernaufwands bedarf, unterscheiden sich die anderen beiden Verfahren erheblich in ihrem Aufwand.

So müssen bei einem klassischen EA alle Sites eine Regelbasis gegen alle anderen Sites erlernen, um die maximale Performanz zu erreichen, was im Rahmen größer werdender Grids schlecht skaliert, auch wenn die Optimierung unter egoistischen Gesichtspunkten (und hochparallel) von jeder einzelnen Partei durchgeführt werden kann.

Das Lernen mit einem coevolutionären Algorithmus hingegen verlangt nur eine einzelne Optimierung aber auch eine höhere Kooperationsbereitschaft zwischen den Parteien, da die nachträgliche Auswahl von Regelbasen unter der Menge vorteilhafter Paarungen großen Einfluss darauf hat, in welchem Maße welche Partei vom Grid profitiert.

Ein signifikanter Performanzunterschied, der generell das eine oder das andere EA-Verfahren befürwortet, konnte nicht nachgewiesen werden.

Kapitel 8

Ressourcendelegation in dezentralen Computational Grids

Der konzeptionelle Unterschied zwischen der Aktivitätendelegation und der Ressourcendelegation liegt darin begründet, dass eine Site bei Delegation einer Aktivität in der Regel keinen direkten Einfluss mehr auf die tatsächliche Ausführung der Aktivität auf den Fremdressourcen nehmen kann. Ist eine Aktivität einmal delegiert, so wird sie praktisch nach dem „Best effort"-Prinzip auf den Fremdressourcen zur Ausführung gebracht[1]. Die delegierende Site kann höchstens durch explizite Ausführungsgarantien ein entsprechendes Startfenster fordern. Bei der Ressourcen-

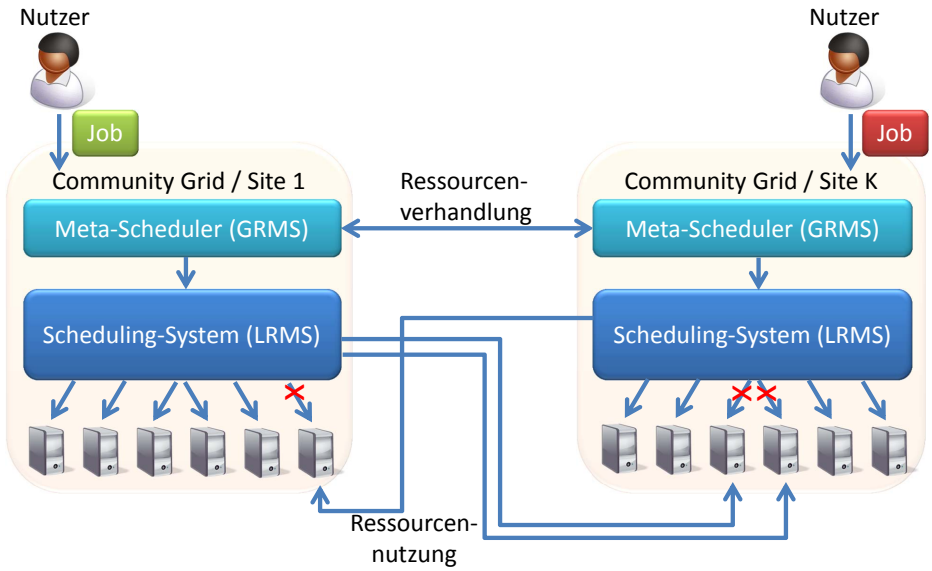


Abbildung 8.1: *Erweiterung des Basismodells um die Ressourcendelegation.*

[1]Fremde Jobs könnten z.B. auf Basis eines LRMS mit mehreren Queues in einer gesonderten Queue mit niedriger Priorität platziert werden und so im schlimmsten Fall sehr lange auf ihre Ausführung warten.

delegation hingegen werden zwischen den autonomen Schedulern Verhandlungen zur temporären Exklusivnutzung fremder Ressourcen durchgeführt. Wie in Abbildung 8.1 dargestellt, werden ausgeliehene Ressourcen nach Verhandlung kurzzeitig in das LRMS der ausleihenden Site eingebunden. Gleichzeitig werden ausgeliehene Ressourcen für den Zeitraum der Delegation nicht von der bereitstellenden Site genutzt.

Aus Sicht der leihenden Site bietet dies nach erfolgreicher Delegationsverhandlung eine größere Planungssicherheit, da sie nun selbst bestimmen kann, zu welchem Zeitpunkt sie welche Jobs auf den geliehenen Ressourcen zur Ausführung bringt.

Des Weiteren wird die Verantwortlichkeit im Hinblick auf die Zugriffskontrolle von der verleihenden Site hin zu der leihenden Site verschoben. Bei der Aktivitätendelegation müssen auf der ausführenden Seite Nutzer-Accounts mit entsprechenden Rechten vorliegen, damit auch die Aktivitäten von Nutzern des Delegationspartners ausgeführt werden können. Im Falle der Ressourcendelegation sorgt die ausleihende Site bezüglich der ihr bekannten Nutzer für die Zugriffskontrolle, indem sie den Zugriff auf geliehene Ressourcen ebenfalls nur den Nutzern gewährt, die auch auf die lokalen Ressourcen zugreifen können. Dies reduziert den Administrationsaufwand, da jede Site auf ihr eigenes Sicherheitskonzept (Zertifikate, Rechtemanagement) zurückgreifen kann.

8.1 Algorithmen für die Ressourcendelegation

Im Gegensatz zur Aktivitätendelegation gibt es bei der Ressourcendelegation weit weniger algorithmische Forschungen, da diese erst mit der Entwicklung der technischen Voraussetzungen attraktiv geworden ist.

So bedurfte es z.B. auf Verhandlungsebene zunächst der Möglichkeit, SLAs zur temporären Dienstnutzung erstellen zu können. Dies dient einerseits der Absicherung des Ressourcennutzers, so dass dieser in der Lage ist, eine Art elektronischen Mietvertrag abzuschließen. Andererseits dienen SLAs auch der Absicherung der delegierenden Partei, da sie über die SLAs stets (z.B. im Falle eines Missbrauchs) verfolgen kann, in welchem Zeitraum Ressourcen nicht unter der eigenen sondern der Fremdkontrolle einer anderen Site standen.

Auch die Möglichkeit zur dynamischen Ein- und Ausgliederung von Ressourcen in Grid-Middlewares bzw. lokalen Ressourcen Management Systemen musste geschaffen werden. Diese technische Anbindung von Fremdressourcen wurde insbesondere durch die Fortschritte in der Virtualisierung vereinfacht, da es so nun möglich ist, virtuelle Zugangspunkte zum Fremdsystem (vgl. Abschnitt 6.2) bereitzustellen, die den Administrationsaufwand für beide Seiten minimieren (Rechtemanagement, Protokolle, Beschreibungssprachen für Jobs usw.).

Ziel der nachfolgenden Forschungen war es, Algorithmen für die Ressourcendelegation zu finden, welche unter ähnlichen Modellbedingungen — wie zuvor bereits die Aktivitätendelegation — zu einer Verbesserung der Scheduling-Qualität durch den Austausch von Rechenkapazität führen.

Eine algorithmische Lösung, welche diese Voraussetzungen am ehesten erfüllt, ist beispielsweise in der Arbeit von Iosup et al. [65] gegeben.
In dem dort vorgestellten Ansatz wird für jeden Job ein eigener *Jobmanager* erstellt, welcher mit einem lokalen *Sitemanager* bezüglich der Ressourcenanforderungen des Jobs (parallele Maschinen) in Verhandlung tritt. Kann Letzterer die Ressourcen für die Anfrage nicht bereitstellen, so ist er in der Lage, die Anfrage an einen bekannten Sitemanager weiterzuleiten. Wenn sich innerhalb dieser (durch eine maximale Länge beschränkten) Verhandlungskette ein Sitemanager findet, welcher über genügend freie Ressourcen verfügt, so werden die Ressourcen über die Kette hinweg direkt dem Jobmanager zugesprochen.

In dem skizzierten Szenario treten die Sitemanager lediglich als Vermittler zwischen einer jeden Jobinstanz und den einzelnen Ressourcen auf. Von einer wirklichen Rekonfiguration der lokalen Ressourcen einer Site kann dabei nicht die Rede sein, da sich die Scheduling-Domäne einer Site dadurch nicht ändert. Streng genommen handelt es sich bei dem Ansatz um eine Aktivitätendelegation, da die Ressourcen nach erfolgreicher Abarbeitung eines einzelnen Jobs direkt zurückgegeben werden. Zudem verwalten die Sites ihre lokalen Ressourcen nicht selbst und bestimmen auch nicht die Ausführungsreihenfolge lokal eingereichter Jobs. Stattdessen kontrollieren sie die Reihenfolge und Weiterleitungswege für Ressourcenanfragen einzelner Jobs, die sich innerhalb des Grids verbreiten.

Die Entscheidung, an welche benachbarten Sitemanager eine Anfrage bei Nichterfüllung weitergeleitet wird, wird auf Basis von Lastinformationen getroffen, welche die Sites untereinander austauschen. Dies entspricht allerdings einer klaren Verletzung des restriktiven Informationsmodells, welchem sich — nach eigener Aussage — auch die Autoren des Papers verschrieben haben.

In einer vorangegangenen eigenen Veröffentlichung [44] zu dem Thema Ressourcendelegation in Grids wurde zwar Wert auf die Wahrung der restriktiven Informationspolitik gelegt. Allerdings wurden die Ressourcen auch in diesem Fall nur für einzelne Jobs geliehen und nicht für spätere Jobs weiterverwendet. Daher dient die Veröffentlichung lediglich als Grundlage für das im Anschluss eingeführte Modell.

Es erfolgt zunächst eine kurze Einführung in das angenommene Modell und verwendete Bewertungsgrößen, bevor im weiteren Verlauf eine detaillierte Beschreibung und Analyse der entworfenen Strategien vorgenommen wird.

8.1.1 Modell und Bewertungsgrößen

Das Modell zur RD setzt auf das zuvor in Abbildung 8.1 (Seite 99) erweiterte Basismodell auf, in dem autonome Scheduler auf Grid-Ebene miteinander verhandeln. Die Sites sind dabei direkt mit jeder anderen Site verbunden und können Anfragen zur Ressourcendelegation unabhängig stellen, akzeptieren und ablehnen.

Die Verhandlungen sollen weiterhin mit einem restriktiven Informationsmodell ohne unnötigen Austausch von dynamischen Lastinformationen geschehen. Darü-

ber hinaus sollte das Ressourcendelegationsproblem möglichst auf GRMS-Ebene gelöst werden und somit das LRMS weitestgehend unberührt lassen. Folgende Funktionalitäten müssen vom LRMS unterstützt werden:

Ressourcenanbindung: Das LRMS muss in der Lage sein, externe Ressourcen zur Laufzeit einzugliedern und am Ende der Leihzeit wieder auszugliedern.

Reservierung: Das LRMS muss in der Lage sein, lokale Ressourcen im eigenen Ausführungsplan für die Dauer einer Delegation mit einer Reservierung zu belegen, damit diese im entsprechenden Zeitraum nicht von eigenen Jobs belegt werden. Alternativ könnten die Ressourcen auch komplett für die Dauer der Delegation aus dem System entfernt werden.

Gezielte Entnahme: Das GRMS muss in der Lage sein, einzelne — nicht bereits gestartete Jobs — aus dem LRMS zu entnehmen, um auf Basis derer eine Ressourcendelegationsanfrage zu stellen.

Abbrechen mit Neuplanung: Das GRMS muss in der Lage sein, Jobs, die auf geliehenen Ressourcen ausgeführt werden, abbrechen zu können, wenn dies z.B. aufgrund des Leihzeitraumendes nötig sein sollte.

Darüber hinaus werden die Grid-weiten Ressourcen weiterhin als homogen angenommen und sowohl Verhandlung, Bereitstellung von delegierten Ressourcen sowie die Auslagerung von Jobs in die entfernten Ressourcen bedürfen im Modell keines Zeitaufwands.

Auch wenn geliehene Ressourcen temporär in die Scheduling-Domäne einer Site eingebunden werden, so ist eine Multi-Site-Ausführung paralleler Jobs über Site-Grenzen hinweg (z.B. gleichzeitig auf geliehenen und eigenen Ressourcen) analog zum Modell der Aktivitätendelegation nicht vorgesehen.

Die Bewertung der entwickelten Strategien findet über die zuvor etablierte prozentuale AWRT-Verbesserung ($AWRT_I$, vgl. Abschnitt 7.2.1 auf S. 64) statt, welche als Referenzexperimente sowohl rein lokale Simulationen von Sites als auch Resultate anderer Lastaustauschverfahren zulässt.

8.1.2 Eine konfigurierbare Strategie zur Ressourcendelegation

Zu Beginn der Konzeption einer Strategie zur Ressourcendelegation (RD-Strategie) muss zunächst festgelegt werden, welcher Auslöser dafür sorgt, dass diese aktiviert wird, also ein Bedarf an der Einbindung zusätzlicher Ressourcen besteht.

Die vorangehenden Strategien zur Aktivitätendelegation wurden alle durch die lokale Eingabe von Jobs aktiviert, da diese einen Zuwachs an Arbeitslast darstellen, der noch vor der Weitergabe an das LRMS vom GRMS behandelt werden kann.

Auch im Falle der Ressourcendelegation wird der zusätzliche Bedarf an Rechenleistung vornehmlich durch die Hinzugabe neuer Jobs in das System motiviert.

Alternativ wäre es auch möglich gewesen, die Ressourcendelegation komplett unabhängig von Jobeinreichungen z.B. in einem bestimmten Zeitintervall zu starten.

Auf der einen Seite würde dies möglicherweise die Rechenzeit reduzieren, auf der anderen Seite verlöre das System an Dynamik, da sich die Änderungen in der Ressourcenkonfiguration auf die Art und Weise nur sprunghaft und nicht entlang der Lastverläufe orientieren würden.

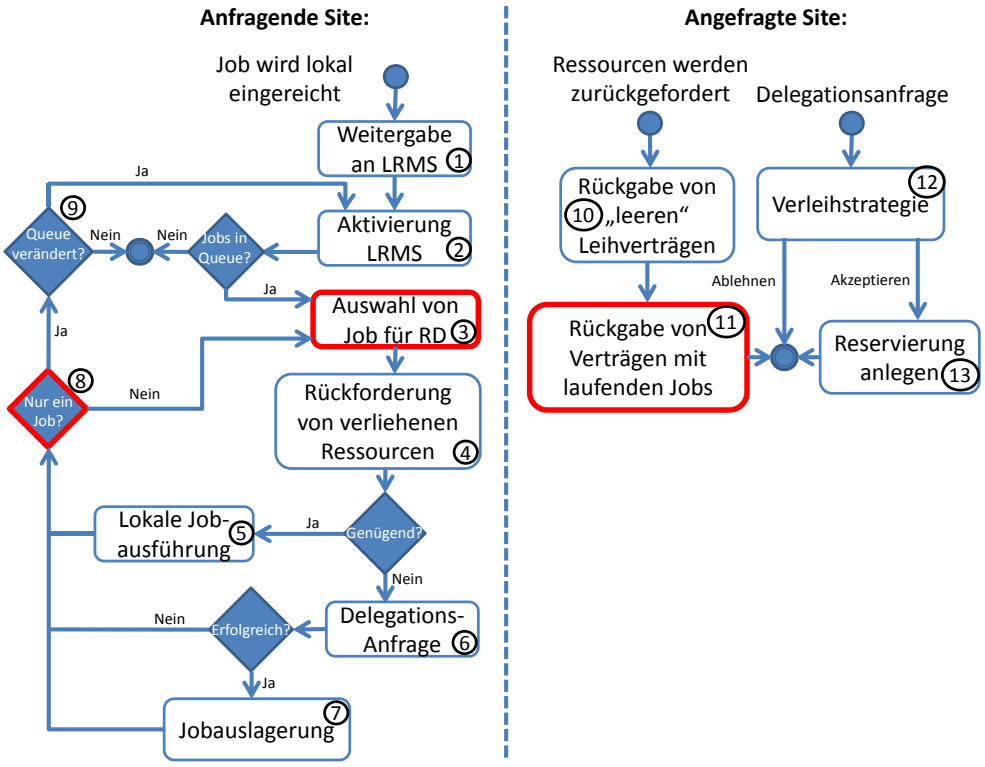


Abbildung 8.2: *Allgemeiner Ablauf einer Strategie zur Ressourcendelegation. Die rot umrandeten Aktivitäten bzw. Bedingungen stellen konfigurierbare Teile der Strategie dar.*

Daher ist der Auslöser unserer RD-Strategie mit dem lokalen Einreichen eines neuen Jobs in die Site gegeben. Dabei wird der Job, wie in Abbildung 8.2 gezeigt, zunächst an das LRMS übergeben und somit in die Queue eingereiht ((1)). Anschließend wird das LRMS aktiviert und somit das lokale Scheduling durchgeführt ((2)). Hierbei können Jobs sowohl auf lokalen oder bereits geliehenen Ressourcen ausgeführt werden. Sollten sich im Anschluss keine Jobs mehr in der Queue befinden, so ist auch keine Veränderung an der lokalen Ressourcenkonfiguration notwendig und die Strategie wird beendet.

In dem Fall, dass noch Jobs in der Queue sind, muss der eigene Ressourcenraum der Site vergrößert werden. Dazu wird zunächst festgestellt, in welchem Maße die lokalen Ressourcen in Abhängigkeit der momentanen Lastsituation rekonfiguriert werden sollen. Auch wenn prinzipiell die Auswahl ganzer Teilmengen von Jobs als

Grundlage für die Ressourcenanfrage möglich ist, wurden in dieser Arbeit immer nur einzelne Jobs dafür ausgewählt (③). Allerdings galt es zu untersuchen, unter welchem Auswahlkriterium eine RD-Strategie zu besserer AWRT-Performanz führt. Die zwei Konfigurationsmöglichkeiten sind:

FCFS: Es wird der Job gewählt, welcher nach ursprünglicher Reihenfolge in der Queue an vorderster Stelle steht. Diese Auswahl stellt aus Sicht der Nutzer eine faire Variante dar, da auf diesem Wege kein Job vorgezogen wird.

MAX: Es wird der Job mit der größten Parallelität gewählt. Grundidee hierbei ist, dass dieser Job innerhalb der Queue am wahrscheinlichsten zu einer Blockierung der eigenen Ressourcen führt und somit durch seine Auslagerung auf Fremdressourcen ebenfalls die Ausführung nachfolgender Jobs beschleunigt werden kann.

Grundsätzlich existieren zwei verschiedene Paradigmen zur Ressourcendelegation. Im ersten Paradigma werden Delegationsanfragen mit einer festen Zahl paralleler Maschinen und einem festen Leihzeitraum gestellt und das Ergebnis in einem SLA festgehalten. Im zweiten hingegen werden die Anfragen mit einem potentiell unendlichen Leihzeitraum gestellt. Somit gehen ausgeliehene Ressourcen auf unbestimmte Zeit in die Scheduling-Domäne der leihenden Site über. Damit die abgebende Site die Ressourcen zurückbekommt, wenn diese benötigt werden, muss sie jederzeit in der Lage sein, den Vertrag ihrerseits zu kündigen und somit die verliehenen Ressourcen zurückzubekommen.

Aus algorithmischer und kollaborativer Sicht macht vor allem das letzte Paradigma Sinn. Dadurch werden die Ressourcenrückgaben nicht anhand zuvor geschätzter Leihzeiten vorgenommen, sondern immer dynamisch an die verändernde Last angepasst. Hat eine Site beispielsweise die Hälfte ihrer Ressourcen verliehen und steht dann einer erhöhten Last gegenüber, so kann sie durch Rückforderung der Ressourcen unter Umständen schneller an ihre Ressourcen gelangen, als wenn sie zuvor vereinbarte Vertragsfristen abwartet.

Der leihende Partner profitiert ebenfalls von der potentiell unendlichen Leihzeit, da er die geliehenen Ressourcen, die ursprünglich für die Auslagerung eines bestimmten Jobs geliehen wurden, solange für weitere lokal wartende Jobs einsetzen kann, bis diese zurückgefordert werden.

Aus diesen Gründen werden bei Rekonfiguration des Systems in Schritt ④ zunächst die eigenen verliehenen Ressourcen zurückgefordert, um den betrachteten Job doch noch lokal auszuführen. Dieser Schritt macht von vornherein nur Sinn, wenn die Summe von momentan freien Ressourcen und den insgesamt verliehenen Ressourcen mindestens der für den Job benötigten entspricht. Andernfalls kann dieser Schritt direkt übersprungen und eine Delegationsanfrage an eine fremde Site gestellt werden.

Sofern die vorherige Bedingung erfüllt ist und die Ressourcen im geforderten Maße zurückgegeben wurden (vgl. spätere Ausführungen zur Ressourcenrückgabe

in den Schritten (10) und (11)), kann der Job in Schritt (5) direkt auf den lokalen Ressourcen ausgeführt werden.

Ist dies nicht der Fall, so bleibt der Site zur Ausführung des betrachteten Jobs nichts anderes übrig, als die Anzahl an geliehenen Ressourcen zu vergrößern. So wie bei der Rückforderung nur so viele Ressourcen zurückgefordert werden müssen, wie zur Ausführung noch fehlen, so werden bei einer Delegationsanfrage ((6)) auch jene Ressourcen berücksichtigt, die bereits von einem potentiellen Delegationspartner geliehen wurden und im Moment frei sind. Die Delegationsanfrage erstreckt sich damit ebenfalls nur über die noch fehlenden Ressourcen, damit der Job nach Erfolg der Anfrage auf externe Ressourcen einer einzelnen Site ausgelagert ((7)) werden kann.

Im Anschluss wurde der Job entweder auf lokalen oder externen Ressourcen gestartet oder ist in der Queue verblieben. Unabhängig vom Ausgang steht die Strategie nun bei Punkt (8) vor der Entscheidung, ob entweder nach dem zuerst betrachteten Job abgebrochen werden soll oder ob die Schritte (4) bis (7) mit dem nächsten Job gemäß der in (3) gewählten Reihenfolge wiederholt werden sollen, bis das Ende der Queue erreicht wurde.

Direkt mit nächstem Job fortfahren: Diese Einstellung führt in der Regel zu sehr häufigen Rekonfigurationen, da eventuell jeder Job in der Queue zu einer Rekonfiguration führt.

Weiter zu Schritt (9): Bei dieser Alternative sollen zunächst die Auswirkungen der vorherigen Umkonfigurationen auf das lokale Scheduling überprüft werden. Schließlich können zurückgeforderte Ressourcen oder ein ausgelagerter Job, erheblichen Einfluss auf die wartenden Jobs in der Queue haben.

Gelangt die Strategie zu Schritt (9) z.B. durch die entsprechende Einstellung oder weil bereits alle Jobs in der Queue den Delegationskreislauf ((3) bis (8)) durchlaufen haben, wird überprüft, ob sich insgesamt eine Veränderung in der Queue ergeben hat. In diesem Fall wird nochmals das LRMS aktiviert, um auch alle Jobs, die nun eventuell lokal ausgeführt werden könnten, noch zur Ausführung zu bringen. Wenn sich durch den Kreislauf inklusive LRMS-Aktivierung, Rückforderung und Leihen nichts mehr an der Queue verändert hat, ist die Strategie beendet.

Bei der angefragten Site können nach den vorherigen Ausführungen zwei verschiedene Arten von Anfragen eintreffen. Bei der ersten Kategorie handelt es sich um Rückforderungen von bereits geliehenen Ressourcen. Hierbei wird seitens der anfragenden Site eine Zahl freizugebender Ressourcen bestimmt.

Da das technische Delegationsprotokoll des DGSI-Projekts (vgl. Abschnitt 6.2) keine Alternative zulässt[2] und ansonsten auch eine aufwändigere Form der Protokollierung notwendig wäre, können SLAs der Ressourcendelegation immer nur

[2]Zwar wurden die hier vorgestellten Strategien zur Ressourcendelegation lediglich simulativ evaluiert, die grundlegende Motivation zur Erforschung der Algorithmen stammte allerdings aus dem Projekt.

im Ganzen aufgelöst werden. Eine Entnahme einzelner Ressourcen aus den zusammenhängenden Verträgen ist somit nicht vorgesehen.

Es obliegt der angefragten Site in Schritt (10), unter allen Verträgen **ohne** momentan ausgeführte Jobs (im Folgenden *leere* Verträge genannt) auszuwählen, welche von diesen beendet werden.

In unserem Fall versucht die Site möglichst viele Fremdressourcen für das eigene Scheduling zu erhalten[3] und setzt daher zur Bestimmung einen Algorithmus zur Lösung des Teilsummenproblems ein. Schließlich ist eine Menge von Verträgen gesucht, die sich in der Summe mindestens über so viele Ressourcen erstrecken, wie von der anfragenden Site zurückgefordert wurden. Unter allen Teilmengen, die diese Bedingung erfüllen, wird diejenige bevorzugt, welche eine minimale Ressourcenanzahl besitzt[4].

Wenn die leeren Verträge nicht ausreichen, um die angeforderten Ressourcenanzahl freizugeben, müssen auch Verträge **mit** laufenden Jobs berücksichtigt werden ((11)). An dieser Stelle besitzt die konfigurierbare RD-Strategie sogar zwei verschiedene Parameter, welche Einfluss auf die Rückgabe der Ressourcen nehmen.

Zum einen muss festgelegt werden, ob die Verträge augenblicklich aufgelöst, die Ressourcen zurückgegeben und die laufenden Jobs damit abgebrochen werden sollen.

Zum anderen muss auf der Menge aller Verträge mit laufenden Jobs eine Ordnung hergestellt werden, da es unter Umständen sinnvoller ist, bestimmte Verträge vor anderen aufzulösen. Für die Vertragsauflösung gelten demnach die zwei folgenden Alternativen:

Direkt auflösen: Diese Einstellung begünstigt die anfragende Site, da sie die geforderten Ressourcen stets sofort zurückbekommt. Sie führt aber auch dazu, dass bereits gestartete Jobs noch einmal neu gestartet[5] werden müssen, was durch die Mehrfachbearbeitung insgesamt zu mehr Last im Grid führt.

[3] Diese Herangehensweise erscheint aus taktischer Sicht und unter dem gegebenen Modell sinnvoll. Wenn sie für die geliehenen Ressourcen jedoch bezahlen müsste, so machte es wenig Sinn, ungenutzte Fremdressourcen zu behalten. In unserem Fall liegt der Fokus allerdings klar auf der Optimierung des Scheduling-Verhaltens innerhalb eines kollaborativen Grids ungeachtet monetärer Aufwendungen für das Leihen von Ressourcen.

[4] Beim Teilsummenproblem ist eine Menge von ganzen Zahlen $I = \{i_1, \cdots, i_n\}$ gegeben. Gesucht ist eine Teilmenge dieser Zahlen mit maximaler Summe, die eine vorgegebene Schranke c nicht überschreiten darf. Bei unserem Problem handelt es sich um einen Spezialfall des Teilsummenproblems. Angenommen die zurückgeforderte Ressourcenzahl sei mit r gegeben und die Menge I besteht aus den Leihgrößen der Verträge, so lässt sich mit $S = \sum_{i \in I} i$ zunächst die Summe aller geliehenen Ressourcen berechnen. Die Lösung zu unserem Problem entspricht genau den Verträgen die **nicht** zur Lösungsmenge des Teilsummenproblems gehören, wenn dieses mit der Schranke $c = S - r$ gelöst wird.

[5] Der Neustart von Jobs ist natürlich nur in dem Fall möglich, wenn diese *idempotent* sind, also ein wiederholter Aufruf zum gleichen Ergebnis bzw. Systemzustand führt wie ein einmaliger Aufruf. Dies ist für Batch-Jobs in der Regel gegeben.

Verzögert auflösen: Bei dieser Einstellung werden bereits freie Ressourcen des Vertrages direkt geblockt, so dass sie nicht mehr von neuen Jobs genutzt werden können. Der Vertrag bleibt so lange bestehen, bis alle betroffenen Jobs zu Ende gelaufen sind. Erst dann erfolgt die Rückgabe aller Ressourcen. Die anfragende Site bekommt dabei nicht unmittelbar alle angeforderten Ressourcen zurück, wodurch bei ihr unter Umständen auch ein Leerlauf entstehen könnte. Dafür gibt es keine Mehrfachbearbeitung von Jobs.

Für die Bestimmung einer Ordnung der Verträge macht es Sinn, zu betrachten, wie viel Arbeitslast die aktuell in einem Vertrag laufenden Jobs bereits abgearbeitet haben. Dazu wird für jeden Vertrag zunächst das kumulative Ressourcenprodukt seiner Jobs berechnet, welches sich über die Parallelität eines jeden Jobs und die bis zu diesem Zeitpunkt bereits abgearbeitete Laufzeit[6] erstreckt. Die Sortierung der Verträge kann nun aufsteigend oder absteigend anhand dieser Größe vorgenommen werden:

Aufsteigend: Verträge mit kleinem laufenden Ressourcenprodukt werden eher beendet. Insbesondere in dem Fall, dass Verträge sofort aufgelöst werden, kann diese Sortierung weiterhin von Vorteil sein, da das bereits abgearbeitete Ressourcenprodukt auf jeden Fall noch einmal neu bearbeitet werden müsste, würden die Jobs abgebrochen.

Absteigend: Verträge mit großem laufenden Ressourcenprodukt werden eher beendet. Diese Einstellung könnte sich ebenfalls als vorteilhaft erweisen, insbesondere wenn die Verträge nicht sofort beendet werden, da Jobs, die schon über eine längere Zeit laufen, mit höherer Wahrscheinlichkeit bald beendet werden.

Gemäß der Sortierung werden so viele Verträge aufgelöst bzw. zur Auflösung vorgemerkt, bis die angeforderte Anzahl an zurückzugebenen Ressourcen erreicht ist.

Die Verleihstrategie ((12)) bei Delegationsanfragen selbst ist verhältnismäßig einfach. Immer wenn genügend Ressourcen für die Erfüllung der Anfrage zur Verfügung stehen, wird diese akzeptiert und die Ressourcen werden mit einer entsprechenden Reservierung in Schritt (13) für die lokale Ausführung gesperrt. Anschließend oder bei Ablehnung der Anfrage ist die Strategie beendet.

Insgesamt ergeben sich durch die vier binär konfigurierbaren Parameter $2^4 = 16$ verschiedene Kombinationen, die nun im folgenden Abschnitt anhand ihrer AWRT-Performanz untersucht werden.

[6] Die tatsächliche Laufzeit ist aufgrund der Online-Charakteristik des Problems nicht bekannt.

8.1.3 Evaluation der RD-Strategie

Die nachfolgende Evaluation dient zunächst der Feststellung, welche der vier Parameter in welcher Ausprägung zu einem vorteilhaften Scheduling-Verhalten der Sites führen. Hierbei wird sich zeigen, dass mit der „Vertragsauflösung“ und der „direkten Fortführung der Delegationsschleife“ lediglich zwei der vorgestellten Parameter einen maßgeblichen Einfluss auf das System besitzen.

Zusätzlich wird sich zeigen, dass der Lastaustausch über Ressourcendelegation in der besten gefundenen Parameterausprägung in jedem Fall zur Verbesserung der Performanz beider Sites gegenüber lokaler Ausführung führt und sich bei Anwendung von FCFS in etwa zwischen den Leistungen von Fuzzy-AD und AWF positioniert.

Die nähere Betrachtung der Performanz einer RD-Strategie mit EASY wird zudem klarmachen, dass diese zu einer gerechteren Verteilung der Gewinne hin zu den größeren Sites führt als die Anwendung von AD-Strategien.

Für die Evaluation der 16 verschiedenen Konfigurationen wurden diese analog zur Evaluation von AWF bei der Aktivitätendelegation (vgl. Abschnitt 7.2.3) in paarweisen Simulationen aller fünf Workloads analysiert. Dabei wurden sie auf den Workload-Abschnitten zu fünf und sechs Monaten sowohl unter Anwendung von FCFS als auch EASY für das lokale Scheduling ausgeführt.

In der Bewertung müssen sich alle Strategien im Rahmen eines Turnierverfahrens hinsichtlich der bei der Simulation erreichten AWRT-Ergebnisse messen. Daher wird für jedes Setup (Site-Paar, Laufzeit, lokaler Scheduling-Algorithmus) die Summe aus den AWRT-Verbesserungen (AWRT_I) gegenüber lokaler Ausführung beider Sites berechnet.

Im Anschluss wird für jede Strategie in Abhängigkeit der Verbesserungssumme ein Punktwert zwischen 0 und 10 Punkten ermittelt. Dadurch bekommt die beste Strategie 10 Punkte und die schlechteste 0 Punkte.

Die Summe aller Punkte, die eine Strategie erzielt, gibt Aufschluss darüber, wie vorteilhaft sie für das Scheduling im Vergleich zu den anderen Strategien ist.
Abbildung 8.3 zeigt beispielhaft die erreichten Gesamtpunkte der einzelnen Strategien bei Anwendung von FCFS für das lokale Scheduling und Simulation auf den fünfmonatigen Workload-Abschnitten Die erreichbaren Punktzahlen liegen aufgrund der zehn Site-Paare zwischen 0 und 100 Gesamtpunkten.

Als erstes fällt auf, dass alle Konfigurationen, bei denen Verträge erst verzögert aufgelöst werden (rechte Hälfte der Konfigurationen), besser sind als diejenigen, welche die Verträge direkt auflösen und laufende Jobs abbrechen. Eine Erläuterung hierfür könnte sein, dass das Grid-weite Scheduling unter der verzögerten Rückgabe von Ressourcen insgesamt besser funktioniert, da die Wiederholung eigener Jobs nicht nur schlecht für die eigene Scheduling-Performanz ist, sondern auch die Bereitschaft zur Delegation von Ressourcen an eine Partner-Site senkt.

Darüber hinaus ist auch beim zweiten Parameter — der Frage nach direkter Fortsetzung der inneren Delegationsschleife mit dem nächsten Job — ein Niveau-

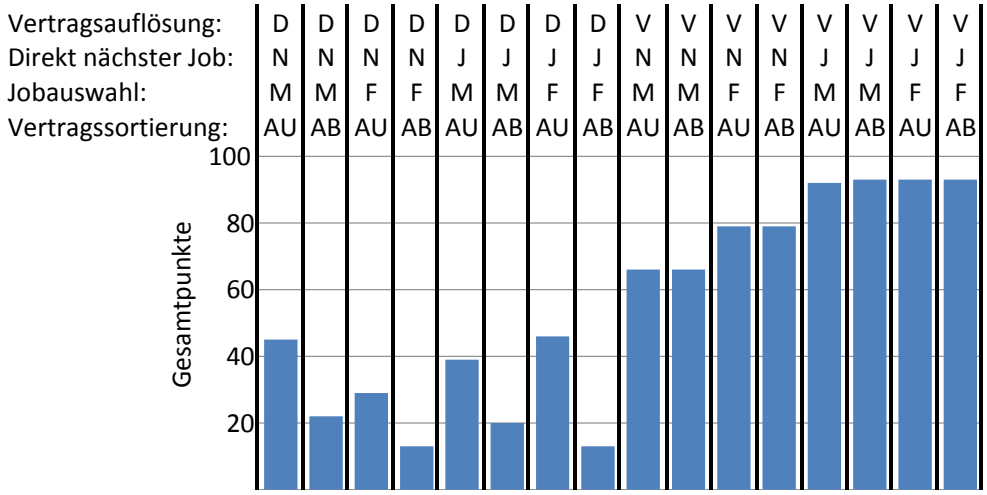


Abbildung 8.3: *Beispiel für ein Turnierergebnis bei der Untersuchung aller* 16 *Strategiekonfigurationen. Betrachtet werden alle zehn möglichen Site-Paare unter lokaler Anwendung von FCFS auf den fünfmonatigen Workload-Sequenzen. Die darüberstehende Legende gibt Aufschluss über die entsprechende Konfiguration. Vertragsauflösung: (D)irekt, (V)erzögert; Direkt nächster Job: (J)a, (N)ein; Jobauswahl: (M)ax , (F)CFS; Vertragssortierung: (AU)fsteigend, (AB)steigend.*

Unterschied zu erkennen. Insbesondere bei den acht besten Konfiguration (ohne direkte Vertragsauflösung) setzen sich diejenigen durch, welche tatsächlich erst die gesamte Queue abarbeiten, bevor erneut das lokale Scheduling durchgeführt wird. Der Grund für die bessere Performanz könnte darin liegen, dass bei wiederholter Abarbeitung der inneren Delegationsschleife die Rekonfiguration insgesamt dynamischer und stärker angepasst an die Last geschieht.

Ebenfalls deutlich wird der Einfluss des vierten Parameters — der Sortierung von Verträgen nach ihrem bereits ausgeführten Ressourcenprodukt (absteigend oder aufsteigend). Dazu werden jeweils zwei benachbarte Konfigurationen in Abbildung 8.3 verglichen. Der Parameter hat offensichtlich nur dann einen ausschlaggebenden Einfluss, wenn die Verträge direkt aufgelöst werden (linke Hälfte der Konfigurationen).

In dem Fall ist stets diejenige Konfiguration besser, welche ihre Verträge aufsteigend sortiert und damit zuerst die Verträge auflöst, die ein geringeres abgearbeitetes Ressourcenprodukt besitzen. Bei näheren Untersuchungen hat sich herausgestellt, dass der Grund für diese Dominanz darin besteht, dass Verträge mit größerem laufenden Ressourcenprodukt oftmals ebenfalls diejenigen sind, die sich über eine größere Anzahl an Ressourcen erstrecken. Werden diese Verträge bei der Rückgabe bevorzugt, so erhält die zurückfordernde Site, welche die Ressourcen aufgrund hoher Last offenkundig benötigt, auch mehr Ressourcen zurück und kann diese auch für andere wartende Jobs nutzen. Werden die Verträge verzögert zurückgegeben, so verschwindet auch der vorteilhafte Effekt, da die Ressourcen für die originäre Site nicht direkt wieder verfügbar sind.

Weitaus indifferenter scheint der Parameter zu sein, welcher bestimmt, ob bei

einer gegebenen Queue wartender Jobs zunächst der größere Job Grundlage für die Ressourcendelegation sein soll oder ob nach der ursprünglichen Queue-Reihenfolge (FCFS) verfahren werden soll. Insbesondere bei den besten vier Strategien, welche die Verträge verzögert zurückgeben und die innere Delegationsschleife wiederholt durchlaufen, scheint die Jobauswahl fast gar keine Auswirkung auf das Ergebnis mehr zu haben.

Der Einfluss der einzelnen Parameter lässt sich auch noch deutlicher darstellen, wenn man ihre Ausprägungen ebenfalls gezielt in einem Turnier bewertet. Hierbei werden die Punkte, die zuvor konkreten Strategien zugewiesen wurden, direkt in einer Bewertung einzelner Parameterausprägungen zusammengefasst[7]. Dieses Verfahren wurde ebenfalls für beide Workload-Abschnitte und lokale Scheduling-Algorithmen durchgeführt. Die Ergebnisse der Untersuchung wurden in Tabelle 8.1 vereinfacht dargestellt. In den Ergebnissen zeigt sich deutlich,

Trace/Algo	Jobauswahl		Direkt nächster Job		Vertragsauflösung		Vertragssortierung	
	Max	FCFS	Nein	Ja	Direkt	Verzögert	Aufsteigend	Absteigend
5 Monate/FCFS	+	-	-	+	-	+	+	-
6 Monate/FCFS	-	+	-	+	-	+	+	-
5 Monate/EASY	+	-	-	+	-	+	+	-
6 Monate/EASY	=		-	+	-	+	+	-
Zusammengefasst	+/-		-	+	-	+	+	-

TABELLE 8.1: *Ergebnisse der Punktbewertung der einzelnen Ausprägungen der vier Parameter in Abhängigkeit von Workload-Abschnitt und lokalem Scheduling-Algorithmus. Ein „+“ bedeutet, dass die Ausprägung im gegebenen Szenario besser als die gegensätzliche Ausprägung ist. „=“ kennzeichnet die exakte Gleichheit beider Ausprägungen nach Punkten und ein „+/-“ in der Zusammenfassung visualisiert, dass es keine eindeutig bessere Ausprägung über alle betrachteten Simulationen gibt.*

unter welchen Ausprägungen die Parameter für die innere Schleife sowie die Vertragsauflösung und -sortierung vorteilhafter für den Ressourcenaustausch sind. Der Jobauswahlparameter hingegen variiert allerdings stark, so dass mal die eine und mal die andere Ausprägung besser ist oder beide die exakt gleiche Bewertung erhalten.
Zusammenfassend hat sich gezeigt, dass in beiden betrachteten Zeiträumen und darüber hinaus auch für beide lokalen Scheduling-Algorithmen immer die gleiche Strategie (F, J, V, AU) Gewinner des Turniers ist. Deckungsgleich mit den vorherigen Beobachtungen durchläuft diese Strategie wiederholt die innere Delegationsschleife und bricht Verträge bei Rückforderung verzögert ab, wobei zuerst diejenigen mit einem kleinen bearbeiteten Ressourcenprodukt aufgelöst werden.

[7]Beispiel: Die beste Strategie (F, J, V, AU) bekommt 10 Punkte. Dann werden für diese Strategie jeweils 10 Punkte für die einzelnen Ausprägungen F, J, V und AU gerechnet. Jeder Parameterausprägung werden also die Gesamtpunkte zugeordnet, welche die Strategien unter ihrer Anwendung erlangt haben.

Aufgrund des (sehr geringen) Einflusses der Jobauswahl (vgl. Abbildung 8.3) setzt die Strategie auf Einhaltung der ursprünglichen Jobreihenfolge (FCFS-Auswahl).

	Versuch	KTH $AWRT_I$	SDSC00 $AWRT_I$	CTC $AWRT_I$	SDSC03 $AWRT_I$	SDSC05 $AWRT_I$
KTH	Lokal		21	36	31	31
	AWF		42	13	-4	1
	EA		14	-5	-6	-4
SDSC00	Lokal	71		84	83	84
	AWF	62		45	-7	7
	EA	28		-18	20	18
CTC	Lokal	4	3		13	15
	AWF	6	14		5	9
	EA	1	0		9	-1
SDSC03	Lokal	4	3	18		18
	AWF	9	9	16		21
	EA	2	-2	-4		-7
SDSC05	Lokal	4	4	12	23	
	AWF	2	1	11	17	
	EA	3	-2	4	2	

TABELLE 8.2: *Performanz der besten RD-Strategie (F,J,V,AU) im Vergleich zur lokalen Ausführung ohne Austausch und im Vergleich zur Aktivitätendelegation mit AWF und dem evolutionären Fuzzy-System. Alle Ergebnisse wurden auf der sechsmonatigen Workload-Sequenz unter Anwendung von FCFS ermittelt.*

Die Performanz dieser Strategie bei paarweiser Simulation wird in Tabelle 8.2 unter Anwendung von FCFS als lokalen Scheduling-Algorithmus in Form der prozentualen AWRT-Verbesserung ($AWRT_I$) sowohl im Vergleich zu lokalem Scheduling als auch im Vergleich zu den zwei Aktivitätendelegationsverfahren AWF (vgl. Abschnitt 7.2.3) und Fuzzy-EA (vgl. Abschnitt 7.2.4) dargestellt.

Um einen fairen Vergleich — insbesondere gegenüber den aufwendig erlernten EA-Regelbasen — durchzuführen, wurden alle Experimente auf dem Übertragungszeitraum von sechs Monaten durchgeführt.

Die Ergebnisse zeigen zunächst einmal, dass die Ressourcendelegation in allen paarweisen Setups ein probates Mittel zur Performanzsteigerung darstellt, wenn sie mit den Ergebnissen des lokalen Schedulings verglichen wird. Je nach Setup belaufen sich die Verbesserungen auf 3% $AWRT_I$, die beispielsweise von CTC und SDSC03 gegenüber einer viel kleineren SDSC00-Site erreicht werden, bis zu sehr großen 84%, um welche sich die letztere Site im umgekehrten Fall verbessert.

Auch im Vergleich zu AWF schneidet die Strategie zur Ressourcendelegation gut ab. Mit Ausnahme der Setups (KTH mit SDSC03) sowie (SDSC00 mit SDSC03) erweist sich die RD-Strategie als performanter, wobei sie sogar in der Lage ist, die Schwierigkeiten des Lastaustauschs zwischen der KTH- und der SDSC00-Site[8] aufzulösen.

[8]Gemeint ist die bereits in Abschnitt 7.2.3 angesprochene Blockade von kleinen durch hochparallele Jobs, die insbesondere bei dem Setup KTH/SDSC00 weniger effizient durch Aktivitätendelegation aufgelöst werden kann.

Im direkten Vergleich mit der Fuzzy-AD-Performanz erweist sich generell keines der beiden Verfahren als effizienter.

So fehlt beispielsweise auch ein klares Muster von Performanzverschiebungen. Zwar könnte auf Basis der Ergebnisse mit KTH als Austauschpartner die Aussage getroffen werden, dass eine Umverteilung der Performanz von der kleineren zur größeren Site stattfindet, wenn die RD-Strategie anstatt der Fuzzy-AD-Strategie eingesetzt wird.

Doch auch diese Aussage ist nicht generell gültig, da es im Falle des SDSC00 in zwei Fällen (mit SDSC03 und SDSC05) zu genau dem gegenteiligen Effekt kommt — einer noch stärken Verbesserung der kleineren Site auf Kosten der größeren Partner.

Mit Ausnahme der zuvor angesprochenen beidseitigen Verbesserungen von KTH und SDSC00 im Vergleich zu den anderen Verfahren ist dies in keinem anderen Setup gelungen.

	Versuch	KTH $AWRT_I$	SDSC00 $AWRT_I$	CTC $AWRT_I$	SDSC03 $AWRT_I$	SDSC05 $AWRT_I$
KTH	Lokal		9	14	13	11
	AWF		2	-1	-8	-4
	EA		23	-1	-8	-8
SDSC00	Lokal	21		34	40	37
	AWF	7		8	0	-3
	EA	1		2	8	4
CTC	Lokal	4	3		15	14
	AWF	2	2		1	-1
	EA	3	1		10	-2
SDSC03	Lokal	2	3	7		8
	AWF	2	2	5		3
	EA	2	2	0		6
SDSC05	Lokal	2	3	6	12	
	AWF	3	1	5	3	
	EA	4	1	6	-1	

TABELLE 8.3: *Performanz der besten RD-Strategie (F,J,V,AU) im Vergleich zur lokalen Ausführung ohne Austausch und im Vergleich zur Aktivitätendelegation mit AWF und dem evolutionären Fuzzy-System*[9]*. Alle Ergebnisse wurden auf der sechsmonatigen Workload-Sequenz unter Anwendung von EASY ermittelt.*

Interessanter ist in dieser Hinsicht ein Vergleich der Ergebnisse unter Anwendung von EASY-Backfilling als lokalen Scheduling-Algorithmus. Wie Tabelle 8.3 verdeutlicht, kommt es hier häufiger zu einer Verbesserung beider Delegationspartner bei Anwendung der RD-Strategie im Vergleich zu den AD-Strategien. Beispiele hierfür sind neben KTH mit SDSC00 auch SDSC00 mit CTC, SDSC00 mit SDSC03 und CTC mit SDSC03 in beiden Fällen (gegenüber AWF und EA) sowie SDSC00 mit SDSC05 und SDSC03 mit SDSC05 in jeweils einem Vergleich.

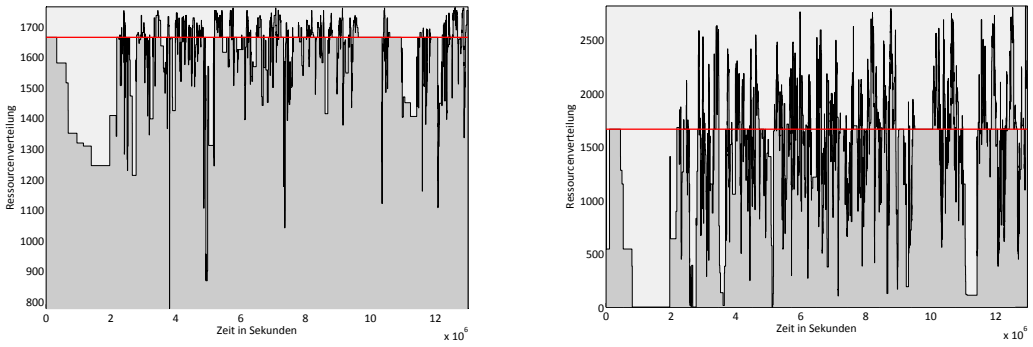
Darüber hinaus kommt es (mit Ausnahme von SDSC03 mit SDSC05 bei Verwendung der EA-Lösung) durch eine RD-Strategie stets zu einer Stärkung der

[9]Da es bei der Optimierung des Fuzzy-Systems unter Anwendung von EASY zu Fehlanpassungen kommt (vgl. Abschnitt 7.2.4) wurden für den Vergleich die Übertragungsergebnisse der mit FCFS gelernten Regelbasen mit anschließender Anwendung von EASY genutzt.

größeren Sites. Somit scheint das RD-Verfahren bei EASY-Backfilling für größere Sites vorteilhafter zu sein als die Vergleichsverfahren der Aktivitätendelegation.

Es sei darauf hingewiesen, dass die Evaluation verschiedener Parameterausprägungen der entworfenen allgemeinen RD-Strategie zwar einem Trainingsvorgang ähnelt, es sich grundsätzlich jedoch nicht um ein Training im Sinne der EA-Verfahren aus der Aktivitätendelegation handelt. Die gefundene Parametrisierung bestimmt lediglich, welche Eigenschaften eine erfolgreiche RD-Strategie besitzen sollte. Der anschließende Einsatz — auch auf neuen Workload-Daten (Sites, Zeiträume) — benötigt kein weiteres Training, ist also eher mit AWF vergleichbar.

Im Vergleich mit AWF schneidet die Ressourcendelegation insgesamt deutlich besser ab und führt bei Anwendung von EASY in vielen Fällen auch für beide Sites zur Verbesserung der Performanz. Dies wird ohne ein aufwendiges Trainingsverfahren erreicht.



(a) KTH mit SDSC05 (b) SDSC03 mit SDSC05

Abbildung 8.4: *Zwei Beispiele für die Rekonfiguration des Grids über die Zeit. Beide Simulationen wurden mit FCFS als lokalen Scheduling-Algorithmus und auf den jeweiligen fünfmonatigen Traces ausgeführt. Die rote Linie gibt das Gleichgewicht zwischen den beiden Sites an.*

Um einen besseren Überblick über die Dynamik des Gesamtsystems während der Ressourcendelegation zu gewinnen, ist es sinnvoll, sich die zeitlichen Verläufe der Ressourcenrekonfigurationen näher anzusehen.

Diese werden beispielhaft für zwei Setups in Abbildung 8.4 dargestellt.

Hierbei sind die Ressourcen der beiden Sites bildlich übereinander gestapelt. Demnach visualisiert die rote Linie die Trennung zwischen den 1664 lokalen Ressourcen des SDSC05 und den 100 Maschinen des Partners KTH in Abbildung 8.4(a) bzw. 1152 Maschinen des Partners SDSC03 in Abbildung 8.4(b).

Die dargestellte Funktion visualisiert die Leihverhältnisse zwischen den Sites. Liegt sie exakt auf der roten Linie, so besitzen beide Sites in der Summe aus lokal verfügbaren und geliehenen Ressourcen die gleiche Anzahl wie sie insgesamt lokal besitzen. Liegt die Funktion darunter, hat sich die kleinere Site (also KTH oder SDSC03) mehr Ressourcen von der SDSC05-Site geliehen als umgekehrt.

In beiden Setups gibt es zu Beginn der Simulation zunächst eine Einschwingphase, in der die kleinere Site sich sukzessive mehr Ressourcen von der SDSC05-Site leiht. Im ersten Setup führt dies dazu, dass sich die KTH-Site mit 414 Ressourcen zeitweise in ihrer Größe verfünffacht, bevor die SDSC05-Site zum ersten Mal damit beginnt, die Ressourcen zurückzufordern. Grund dafür ist ein plötzlicher Anstieg des Workloads beim SDSC05-Trace nach ca. 23 Tagen ($2 \cdot 10^6$ Sekunden).

Diese anfänglich große Leihbereitschaft führt im zweiten Setup sogar dazu, dass temporär die kompletten Ressourcen der SDSC05-Site an die SDSC03-Site übergehen. Bei näherer Betrachtung des SDSC05-Traces wird deutlich, dass im Zeitraum zwischen dem 3. und dem 23. Tag keine neuen Jobs auf der SDSC05-Maschine eingehen und die Ressourcen daher bei Bedarf ohne weiteres an die kleinere Maschine übergehen können.

Mit dem weiteren Verlauf der Simulationen kommt es zu einem regen Austausch von Ressourcen zwischen den Sites.

Da es in vielen vorangegangenen Untersuchungen z.B. zur Aktivitätendelegation oftmals zu einer einseitigen Verlagerung der Last bzw. zu einem Performanzgewinn der kleineren Site auf Kosten der größeren Site gekommen ist, liegt die Vermutung nahe, dass die größere Site nicht oft in der Lage ist, sich Ressourcen von der kleineren Site zu leihen.

Umso überraschender ist die Erkenntnis, dass die SDSC05-Site auch im Extremfall mit KTH als Delegationspartner (Abbildung 8.4(a)) zeitweise in der Lage ist, sich dessen Ressourcen gesamt zu leihen. Umgekehrt tritt dieser Fall in der Simulation nicht ein. Die größte geliehene Menge an Ressourcen (874) besitzt die KTH-Site nach ca. 45 Tagen ($3{,}9 \cdot 10^6$ Sekunden). Dies liegt allerdings nicht an der fehlenden Leihbereitschaft der größeren Site, sondern vielmehr daran, dass die Leihanfragen stets nur lastorientiert durchgeführt werden und die KTH-Site an dieser Stelle keinen weiteren Nutzen aus dem Leihen weiterer Ressourcen ziehen kann.

Die hohe Lastorientierung erkennt man auch an der Zahl der Rekonfigurationen während des Simulationszeitraums. Damit sind systemübergreifend alle Änderungen an den Ressourcenräumen gemeint. Im ersten Setup, welches weitaus weniger Jobs enthält, kommt es durchschnittlich nur alle 1,6 Stunden zu einer Rekonfiguration und auch in der Stunde mit den häufigsten Delegationen (28 an der Zahl) liegt die durchschnittliche Zeit zwischen zwei Rekonfigurationen bei über zwei Minuten.

Im zweiten Setup steigt die Zahl der Jobs im System und damit auch die Zahl an Rekonfigurationen. Nun findet durchschnittlich alle 44 Minuten eine Rekonfiguration statt und das unterliegende LRMS muss insgesamt schneller in der Lage sein, um auf Änderungen in der Ressourcenlandschaft zu reagieren, da es hier in der belebtesten Stunde zu insgesamt 65 Rekonfigurationen kommt.

8.1.4 Auswahlstrategie bei mehreren Partnern

Analog zur Lokationsstrategie bei der Aktivitätendelegation (vgl. Abschnitt 7.2.5) werden mit zunehmender Anzahl von Sites im Grid auch bei der Ressourcendelegation Verfahren für die Priorisierung von Delegationspartnern nötig. Im Folgenden wird eine Analyse möglicher Umsetzungen für solche Auswahlstrategien vorgenommen, die bei detaillierter Betrachtung der Ergebnisse jedoch zu der Erkenntnis führen wird, dass die Auswahlstrategie bei Anwendung der Ressourcendelegation eine weitaus geringere Rolle spielt als die Lokationsstrategie in der Aktivitätendelegation. Eine viel wichtigere Erkenntnis wird sein, dass die Ressourcendelegation bei Anwendung von FCFS und besonders bei Verwendung von EASY die weitaus attraktivere der beiden Delegationsarten im Hinblick auf die erreichten AWRT-Verbesserungen darstellt, wenn die Größe des Grids zunimmt. Zudem findet im Rahmen der Evaluation eine nähere Untersuchung des Rekonfigurationsverhaltens während eines 5-Site-Grids statt, die aufdeckt, dass es unter der gegebenen RD-Strategie vorteilhaft für die Sites ist, ihre eigenen lokalen Ressourcen nach und nach durch Fremdressourcen aus verschiedenen Quellen zu ersetzen, da diese eine höhere Flexibilität in der Nutzung aufweisen.

Anders als bei der Lokationsstrategie unterteilt sich in unserem Fall (unendliche Delegation mit aktiver Rückforderung) eine Auswahlstrategie für die Ressourcendelegation in zwei verschiedene Komponenten. Schließlich gibt es hierbei auch zwei voneinander unabhängige Interaktionsmöglichkeiten mit den Partnern des Grids. Stehen einer Site nicht genügend Ressourcen für die Ausführung eines Jobs zur Verfügung (vgl. Schritt (3) der allgemeinen RD-Strategie in Abbildung 8.2 auf S. 103), so versucht sie zunächst, bereits verliehene Ressourcen zurückzufordern. Nur wenn die Rückforderungen eigener Ressourcen nicht genügen, kommt es zu einer Delegationsanfrage für Fremdressourcen. In beiden Fällen muss entschieden werden, in welcher Reihenfolge die Partner im Grid befragt werden sollen.

Für beide Aspekte einer Auswahlstrategie wurden zwei einfache Ansätze untersucht:

Sortierung nach Site-Größe: Bei dieser Parametrisierung werden die entfernten Sites nach ihrer Größe[10] sortiert. Dies kann entweder auf- oder absteigend geschehen.

Sortierung nach Zahl der verliehenen Ressourcen: In diesem Fall wird entweder auf- oder absteigend nach der Gesamtzahl ausgeliehener Ressourcen an die entsprechende Site sortiert.

Insgesamt gibt es damit vier verschiedene Ausprägungen für jeden Teil der Auswahlstrategie.

[10] Zur Erinnerung: Die Größe bzw. Gesamtzahl an lokalen Ressourcen stellt im restriktiven Informationsmodell die einzige (statische) Information dar, die zwischen den Sites ausgetauscht wird.

In der Summe ergeben sich so 16 unterschiedliche Kombinationen für die Parametrisierung einer Auswahlstrategie, die analog zur allgemeinen RD-Strategie in Form eines Turnierverfahrens miteinander verglichen wurden. Alle Evaluationen wurden dabei auf Basis der zuvor erfolgreichsten RD-Strategie-Parametrisierung (F, J, V, AU) vorgenommen. Lediglich die Auswahlstrategie-Parameter wurden variiert.

Wie zuvor hängt die erreichte Punktzahl einer Parametrisierung von der Summe der AWRT-Verbesserungen aller Sites während der Simulation ab. Im Gegensatz zum vorherigen Turnierverfahren wurden bei diesem Turnier allerdings statt den zehn paarweisen Setups die beiden Setups aus Abschnitt 7.2.5 auf Seite 86 zur Untersuchung der Lokationsstrategie eingesetzt — das 3-Site-Setup aus KTH, CTC und SDSC05 und das 5-Site-Setup mit allen Sites.

Weiterhin wurde gezielt nach einer Parametrisierung gesucht, die auch für beide betrachtete lokale Scheduling-Algorithmen funktioniert. Somit ergibt sich die erreichte Gesamtpunktzahl einer Parametrisierung aus der Summe der erreichten Punkte in vier Wettkämpfen mit je 10 erreichbaren Punkten. Der Wertebereich erreichbarer Gesamtpunkte ist in diesem Fall mit 0 bis 40 gegeben.

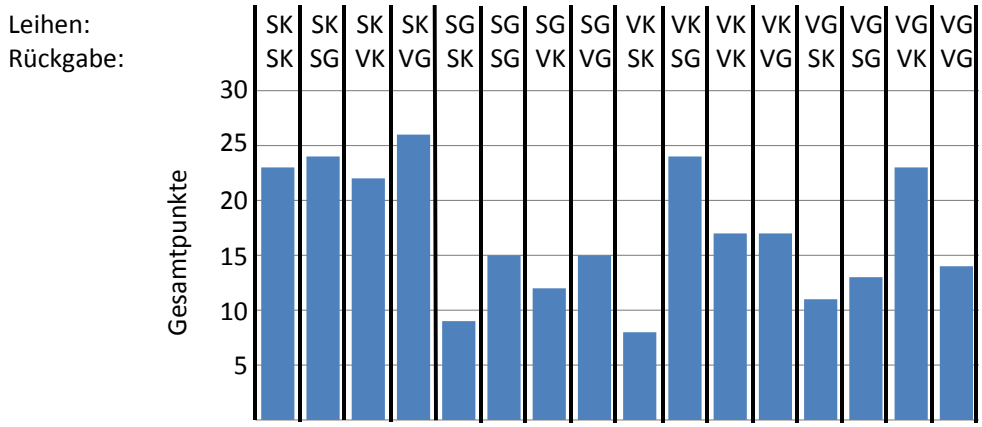


Abbildung 8.5: *Vollständiges Turnierergebnis bei der Untersuchung aller* 16 *Strategiekonfigurationen für die Lokationsstrategie unter zwei Setups (*3 *Sites,* 5 *Sites) und FCFS bzw. EASY als lokalen Scheduling-Algorithmus. Simuliert wurde auf den fünfmonatigen Teilsequenzen. Die darüberstehende Legende gibt Aufschluss über die entsprechende Konfiguration. SK: Aufsteigende Sortierung nach Site-Größe; SG: Absteigende Sortierung nach Site-Größe; VK: Aufsteigende Sortierung nach verliehenen Ressourcen; VG: Absteigende Sortierung nach verliehenen Ressourcen*

Abbildung 8.5 zeigt das Resultat der Turnierbewertung. Da die ersten vier Simulationen allesamt Ergebnisse unter den TOP6 darstellen und darüber hinaus auch die nach Punkten beste Parametrisierung liefern, lässt sich feststellen, dass das beste Leihverhalten darin besteht, die Partner aufsteigend nach der Site-Größe zu

sortieren (SK)[11]. Somit versucht eine Site zunächst einmal, Ressourcen von einem kleineren Partner zu leihen. Gelingt dies nicht, wird der nächstgrößere gefragt usw.

In Kombination mit diesem Leih-Verhalten hat sich das Rückgabe-Verhalten (VG) durchgesetzt, so dass eine Site zunächst bei dem Partner um Rückgabe bittet, der momentan am meisten Ressourcen geliehen hat.

Allerdings ist die ermittelte Strategie lediglich im relativen Vergleich aller Auswahlstrategien untereinander förderlich für die Scheduling-Performanz.

Bei einer näheren Betrachtung der AWRT-Absolutwerte der einzelnen Sites bei Anwendung der besten Strategie im Vergleich mit 100 Evaluationen zufälliger Auswahlstrategien wurde deutlich, dass die Auswahlstrategie — ganz im Gegensatz zur Lokationsstrategie der Aktivitätendelegation — keinen signifikanten Einfluss auf die AWRT-Verbesserungen der Sites gegenüber lokaler Ausführung hat.

Die erreichten $AWRT_I$-Werte schwankten dabei für die kleineren Sites um ca. $+/-5\%$ und für größere eines Setups um $+/-2\%$, was hinsichtlich der erreichten absoluten Verbesserungen gegenüber lokalem Scheduling unerheblich ist.

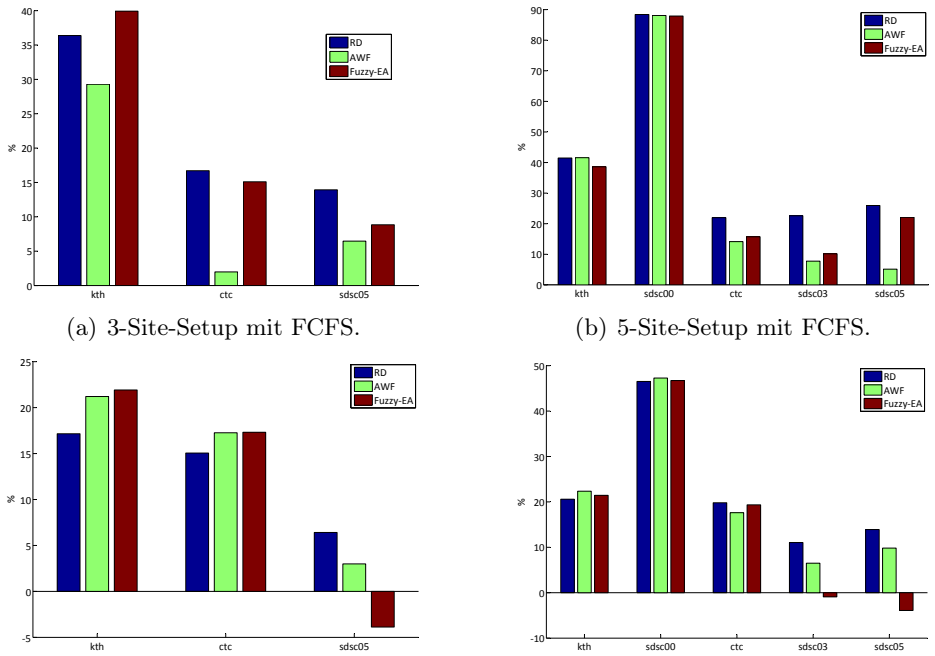
Dies zeigt sich z.B. an dem in Abbildung 8.6 visualisierten Vergleich der besten gefundenen RD-Strategie (inklusive der zuvor bestimmten besten Konfiguration der Auswahlstrategie) mit den Aktivitätsdelegationsstrategien AWF und Fuzzy-EA[12]. In allen dargestellten Fällen erzielt das Verfahren zur Ressourcendelegation gute Verbesserungen gegenüber der lokalen Ausführung. Schwankungen um wenige Prozentpunkte — hervorgerufen durch eine alternative Auswahlstrategie — würden an diesem Ergebnis kaum etwas ändern.

Besonders auffällig ist, dass sich bei Betrachtung größerer Grids noch stärker zeigt, dass die Ressourcendelegation gegenüber den Aktivitätendelegationsverfahren zu einer faireren Verteilung der Verbesserungen führt. So erreichen die kleineren Sites eines Setups (KTH in Abbildung 8.6(a)) bzw. (KTH und SDSC00 in Abbildung 8.6(b)) bei Anwendung der Ressourcendelegation mit lokalem FCFS eine geringere oder vergleichbare Performanz wie bei Anwendung des Fuzzy-Ansatzes. Für die größeren Sites (ab CTC) kommt es zu noch stärkeren Verbesserungen. Im Falle des 5-Site-Setups beträgt sie für alle beteiligten Sites sogar über 20% $AWRT_I$ gegenüber lokaler Ausführung.

Diese Beobachtung kann auch bei der lokalen Anwendung von EASY-Backfilling gemacht werden. Da der Fuzzy-Ansatz in besagtem Setup sogar negative Resultate für die größeren Sites erreicht, müssen die Ergebnisse eher mit der Performanz von AWF unter EASY verglichen werden.

[11] Der erste Buchstabe der Abkürzung steht für das Sortierkriterium, wohingegen der zweite Buchstabe die Reihenfolge beschreibt.
Sortierkriterium: (S)itegröße oder (V)erliehene Ressourcen.
Reihenfolge: (K)lein zuerst oder (G)roß zuerst.

[12] Auch in diesem Fall stammen die Regelbasen für die EASY-Anwendung aus der EA-Optimierung mit FCFS.



(a) 3-Site-Setup mit FCFS.

(b) 5-Site-Setup mit FCFS.

(c) 3-Site-Setup mit EASY.

(d) 5-Site-Setup mit EASY.

ABBILDUNG 8.6: *Vergleich der $AWRT_I$-Ergebnisse im Bezug auf die lokale Ausführung. Dargestellt sind zwei verschiedene Setups mit drei und fünf Sites bei lokaler Anwendung von FCFS und EASY. Gegenübergestellt sind die Ergebnisse von der besten gefundenen Ressourcendelegationsstrategie (RD), der Anwendung von AWF als Aktivitätendelegationsverfahren und Lastaustausch über EA-optimierte Fuzzy-Regelbasen.*

Auch hier zeigt sich eine Verschiebung der Verbesserungen von kleineren zu größeren Sites. So erreichen KTH und CTC 17% bzw. 15% $AWRT_I$ im 3-Site-Setup (vgl. Abbildung 8.6(c)), doch haben sie so gegenüber der AWF-Performanz zwei bis fünf Prozentpunkte $AWRT_I$ eingebüßt. Dies geht gleichzeitig mit einer Verdopplung des $AWRT_I$-Wertes für die SDSC05-Site einher.

Noch deutlicher wird dieser Trend mit zunehmender Größe des Grids. In dem 5-Site-Setup erreichen die drei kleinsten Sites (KTH, SDSC00 und CTC) annähernd die gleiche Performanz — unabhängig vom eingesetzten Lastaustauschverfahren. Die zusätzlichen 5% Performanz der beiden größeren Sites sind insofern bemerkenswert, zumal sie unter Anwendung von EASY erreicht wurden, welches bereits bei rein lokalem Scheduling zu guten Ergebnissen führt.

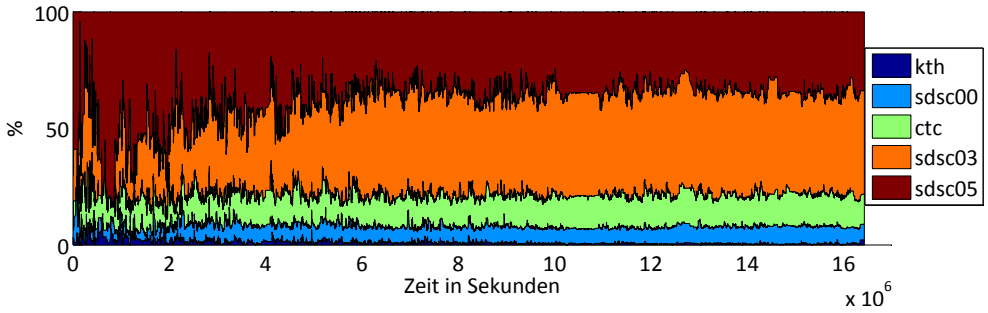


Abbildung 8.7: *Prozentuale Verteilung der systemweiten Ressourcen auf die einzelnen Sites über die Zeit von sechs Monaten bei Anwendung von FCFS als lokalen Scheduling-Algorithmus und der besten gefundenen RD-Strategie inklusive bester Auswahlstrategie. Startverteilung: KTH:* 2,9%*; SDSC00:* 3,7%*; CTC:* 12,4%*; SDSC03:* 33,2%*; SDSC05:* 47,9%

Für die nähere Analyse des Rekonfigurationsverhaltens der einzelnen Sites zeigt Abbildung 8.7 beispielhaft die prozentuale Verteilung der systemweiten Ressourcen über die Zeit. Für jede Site und jeden Zeitpunkt der sechsmonatigen Simulation ist hierbei angegeben, über wie viel Prozent der Gesamtressourcen sie momentan verfügt — unabhängig davon, ob es sich hierbei um lokale oder hinzugeliehene Ressourcen handelt.

Ausgehend von der Startverteilung der Ressourcen (z.B. 100 Maschinen des KTH entsprechen 2,9% der Gesamtressourcen), kommt es im ersten Viertel der Simulation (ca. $4 \cdot 10^6$ Sekunden) noch zu starken Schwankungen der prozentualen Anteile.

Besonders extrem sind diese Rekonfigurationen in den ersten 12 Tagen (ca. $1 \cdot 10^6$ Sekunden) der Simulation, in denen die SDSC05-Site zeitweise auf ein Viertel (12%) ihrer ursprünglichen Größe (47,9%) reduziert wird und die SDSC03-Site all ihre Ressourcen dem Grid zur Verfügung stellt.

Mit dem Ende der Einschwingphase, stellt sich bezüglich des Leihverhältnisses ein Gleichgewicht ein, in dem die Sites einen relativ unveränderten durchschnittlichen Anteil an den Gesamtressourcen halten[13].

Im Vergleich zu den anfänglichen Startverteilungen gibt es hierbei Sites, die ihre Ressourcenlandschaft dauerhaft vergrößert haben und jene, die im Durchschnitt mit weniger Ressourcen auskommen, als ihnen lokal zur Verfügung stehen.

Hinsichtlich der durchschnittlichen Verteilung haben sich KTH und SDSC05 verkleinert. Mit durchschnittlich 1,2% der Gesamtressourcen hat die KTH-Site verhältnismäßig am meisten bei der Teilnahme am Grid geopfert und trotzdem mit 40% AWRT_I gegenüber lokaler Ausführung eine gute Performanz erreicht. Die SDSC05-Site hat mit einer durchschnittlichen Reduzierung ihres Anteils von 47,9%

[13]Ein Grund für die vorliegenden Einschwingphase liegt darin, dass die Sites zu Beginn keinerlei laufende oder wartende Jobs besitzen. Erst mit der kontinuierlichen Eingabe von Arbeitslast in das System, besitzen sie genügend sogenannten *Backlog* also eine Menge von Jobs, die sich in den Queues ansammelt und eine Rekonfiguration der Ressourcen erfordern. Dabei beeinflusst die systemweite Verteilung der Jobs auch die Anteile, die jedes System an den Gesamtressourcen des Grids besitzt.

um 9,9 Prozentpunkte auf 38% zwar relativ gesehen weniger geopfert, stellt jedoch in Hinsicht auf die dem Grid bereitgestellte Rechenkapazität den wertvollsten Anbieter dar.

Nutznießer der Ressourcendelegation sind in erster Linie die SDSC00-Site, welche ihre durchschnittliche Site-Größe von 3,68% auf 6,89% fast verdoppelt, aber auch die SDSC03-Site, die ihren Ressourcenanteil mit 40,35% im Durchschnitt um ca. 249 zusätzliche Ressourcen erhöht.

Die CTC-Site erhöht ihren Ressourcenanteil durchschnittlich um ca. 40 Ressourcen, was bei 430 lokalen Ressourcen immerhin einer Vergrößerung um ca. 9% darstellt.

Zusätzlich lässt sich für jede Site auch ein zeitlicher Verlauf darstellen, aus welchen Quellen sie ihre Ressourcen bezieht.

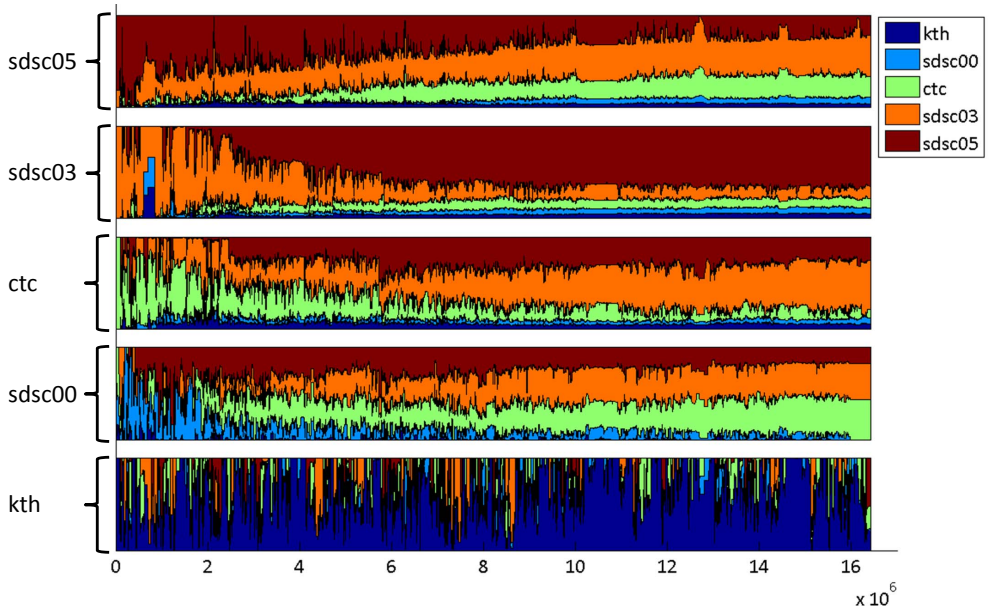


Abbildung 8.8: *Prozentualer Anteil an verfügbaren Ressourcen nach Quelle — separat für jede Site des 5-Site-Setups bei sechsmonatiger Ausführung der besten RD-Strategie mit lokalem FCFS auf allen Sites.*

Abbildung 8.8 zeigt diesen Verlauf separat für jede der fünf Sites des vorherigen Setups (sechs Monate, FCFS)[14]. Hierbei startet eine Site zu Beginn lediglich mit ihren lokalen Ressourcen und mit der Zeit werden diese anteilig von Fremdressourcen verdrängt.

[14]Es sei darauf hingewiesen, dass es sich hierbei um eine relative Darstellung der lokalen Ressourcenverteilung handelt, die nichts über die Gesamtzahl verfügbarer Ressourcen einer Site aussagt, sondern einen Überblick über die Herkunft der verfügbaren Ressourcen geben soll.

Das überraschende Ergebnis der Untersuchung ist, dass nahezu alle Sites (mit Ausnahme der KTH-Site) ihre lokalen Ressourcen mit zunehmender Simulationszeit durch Fremdressourcen der Delegationspartner ersetzen.

So sind bei der SDSC05-Site am Ende der Simulation lediglich 21% der verfügbaren Ressourcen lokal. Mit 44% vom SDSC03 und 23% vom CTC stellen diese beiden Partner allerdings einen mehr als dreimal so großen Teil der Ressourcen des SDSC05.

Noch extremer ist die Ersetzung bei der SDSC00-Site, die am Ende gar keine lokalen Ressourcen mehr verwaltet, sondern für ihr Scheduling lediglich auf Ressourcen von CTC, SDSC03 und SDSC05 zurückgreift.

In erster Linie gibt es zwei Gründe für den beobachteten Effekt. Zum einen ist die Benutzung von Fremdressourcen unter dem gegebenen Modell (keine Transferzeiten für Jobs und Daten, homogene Ressourcen) nicht nachteilig gegenüber der bevorzugten Nutzung von eigenen Ressourcen.

Zum anderen begünstigt die beste ermittelte RD-Strategie gewissermaßen die Nutzung von Fremdressourcen, auch wenn dies auf den ersten Blick widersprüchlich erscheint. Schließlich werden für einen nicht lokal ausführbaren Job zunächst die eigenen verliehenen Ressourcen zurückgefordert, bevor im Anschluss — falls nötig — eine Delegationsanfrage an potentielle Delegationspartner gerichtet wird. Dieser Ablauf lässt zunächst vermuten, dass der Anteil an lokalen Ressourcen im eigenen Ressourcenraum größer sein müsste.

Allerdings sind die Rückforderungen bei der ermittelten RD-Strategie mit einem nicht zu vernachlässigenden Nachteil verbunden. Da bereits laufende Jobs auf den ausgeliehenen Ressourcen nicht abgebrochen werden, bekommt die zurückfordernde Site die Ressourcen oft verzögert zurück, was trotz aller Rückforderungsversuche in Delegationsversuchen dieser Site resultiert.

Zugesicherte Fremdressourcen sind allerdings sofort verfügbar und können direkt mit Last belegt werden. Setzt sich dieser Prozess nur lange genug fort, so werden die lokalen Ressourcen aufgrund ihrer Unattraktivität nach und nach von Fremdressourcen verdrängt.

8.2 Abschluss der Ressourcendelegation

Im Rahmen der Schilderungen zur Ressourcendelegation des vorangegangenen Kapitels wurden in erster Linie algorithmische Aspekte dieser Delegationsart untersucht, die zur Minimierung der AWRT von Grid-Sites führen. Dazu wurden zunächst die für den Austausch wichtigen Modellannahmen (restriktives Informationsmodell, keine Datentransferzeiten, homogene Ressourcen und keine Multi-Site-Ausführung) wiederholt.

Im nächsten Schritt wurde eine konfigurierbare RD-Strategie vorgestellt, bei der sich Sites für potentiell unbeschränkte Leihzeit Ressourcen leihen und diese nach einer Rückforderung der leihenden Partei zurückgeben. Für die Konfiguration

der Strategie wurden vier verschiedene Parameter untersucht, welche Einfluss auf das Delegationsverhalten der Site auf Grid-Ebene besitzen.

Die anschließende Bewertung der durch alle Permutationen der Parameterausprägungen entstehenden Konfigurationen hat zu einer Strategie geführt, welche die Jobs in der Queue nach FCFS-Sortierung abarbeitet und in hohem Maße auf eine Rekonfiguration des eigenen Ressourcenraums setzt.

Zusätzlich hat es sich als besonders vorteilhaft für die Performanz des gesamten Grids erwiesen, wenn Jobs, die auf geliehenen Ressourcen laufen, bei Rückforderung vor der Rückgabe ordnungsgemäß beendet anstatt abgebrochen werden.

Im Vergleich mit Lastaustauschverfahren der Aktivitätendelegation hat sich gezeigt, dass die Ressourcendelegation unabhängig vom lokal verwendeten Scheduling-Algorithmus zu guten Ergebnissen hinsichtlich der AWRT führt.

Ist sie bei FCFS noch besser als AWF und vergleichbar gut wie EA-optimierte Fuzzy-Regelbasen der Aktivitätendelegation, so weist sie bei EASY eine deutlich robustere Performanz gegenüber den Fuzzy-Regelbasen auf und sorgt im Vergleich mit AWF für eine fairere Verteilung des mit dem Lastaustausch verbundenen Verbesserungspotentials.

Ausgehend von der vorherigen Ergebnissen wurden auch Auswahlstrategien für die Ressourcendelegation mit zunehmender Grid-Größe untersucht, die hinsichtlich der erreichbaren AWRT der einzelnen Sites nur einen geringen Einfluss besitzen.

Eine genauere Untersuchung des Rekonfigurationsverhaltens der Sites in einem 5-Site-Setup hat zudem gezeigt, dass es sich für die meisten Sites unter dem gegebenen Modell lohnt, ihren Ressourcenraum aus flexibleren Fremdressourcen unterschiedlicher Partner aufzubauen, anstatt die eigenen Jobs auf lokalen Ressourcen auszuführen.

Abgesehen von der vielversprechenden Performanz im Vergleich zur Aktivitätendelegation besitzen die vorgestellten Verfahren zur Ressourcendelegation einige Eigenschaften, die bei tatsächlicher produktiver Anwendung berücksichtigt werden müssen.

Zum einen sind dies Anforderungen an das LRMS, da das Verfahren zur Ressourcendelegation durch Entnahme von Jobs aus der Queue und das Anbinden fremder bzw. Reservieren lokaler Ressourcen nicht mehr ausschließlich auf Grid-Ebene (GRMS, vgl. Einführung von Kapitel 8) agiert.

Zum anderen ist dies das notwendige Vertrauen zwischen den Parteien, wenn Ressourcen mit potentiell unendlicher Zeit verliehen und auch auf Rückforderung nur verzögert — ohne Abbrechen von Jobs — zurückgegeben werden.

Teil III

Hybrides Cloud Scheduling

Kapitel 9

Einführung in das Cloud Computing

Auch wenn es sich beim Cloud Computing um ein verhältnismäßig junges Paradigma der Ressourcennutzung handelt, so existiert bereits jetzt eine große Anzahl verschiedener Definitionen, welche beispielsweise auf das zugrundeliegende Dienstmodell (Infrastruktur, Entwicklungsplattform, Softwarebetrieb) oder die Ausrichtung (privat, öffentlich, gemeinschaftsorientiert, hybrid) einer Cloud eingehen. Verbreitete Akzeptanz finden hierbei vor allem die Definitionen des amerikanischen *National Institute of Standards and Technology (NIST)* [85, 3]. In diesen wird Cloud Computing als „[...] Modell für den universellen, bequemen und bedarfsorientierten Netzwerkzugriff auf einen geteilten Pool konfigurierbarer Rechenressourcen (z.B. Netzwerke, Server, Speicherplatz, Applikationen und Dienste) [...]“[1] bezeichnet, welcher zudem „[...] schnell und mit minimalem Aufwand oder anbieterseitigem Eingriff [...]“[2] erfolgt.

Charakteristisch für das Modell ist dabei sowohl der Zugriff auf geteilte Ressourcen über das Netzwerk, der eben möglichst selbstständig durch den Nutzer erfolgen soll, als auch die große Flexibilität in der Nutzung der Ressourcen. So sind die Nutzer über spezielle (meist anbieterspezifische) Verwaltungssoftware in der Lage, vollständige Rechencluster in der Cloud zu betreiben, die sich automatisch an die Rechenlast anpassen. Diese bedarfsorientierte Skalierbarkeit wird ständig überwacht und erlaubt somit sehr feingranulare Modelle für die Abrechnung der Ressourcennutzung, so dass der Nutzer am Ende nur für die Ressourcen bezahlt, die er tatsächlich genutzt hat.

Aus diesen Gründen eignet sich das Cloud Computing sehr gut für eine Auslagerung von Arbeitslast der eigenen permanent betriebenen Rechenressourcen. Grundsätzlich existieren drei verschiedene, in der Regel aufeinander aufbauende Dienstmodelle. Wie in Abbildung 9.1 zu sehen, bietet das unterste Dienstmodell *Infrastructure as a Service (IaaS)* dem Nutzer die größte Kontrolle über die verfüg-

[1] [85], Kap. 2, S. 2, Übersetzung aus dem Englischen.
[2] ebd.

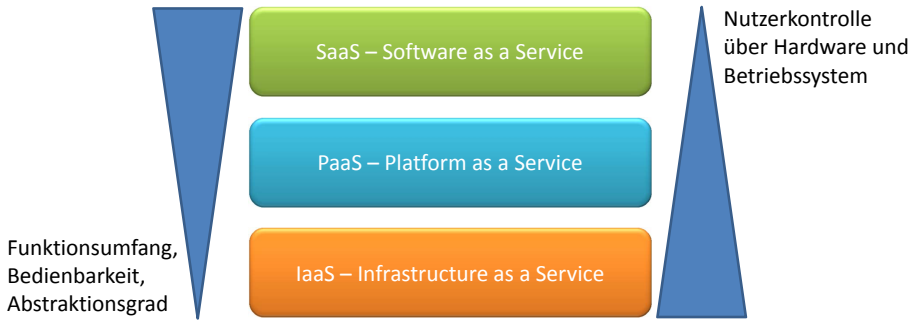


Abbildung 9.1: *Dienstmodelle einer Cloud*

bare Hardware bzw. Konfiguration des eingesetzten Betriebssystems. Gleichzeitig ist der Abstraktionsgrad von der unterliegenden Hardware bzw. die Verfügbarkeit höherer Funktionen verhältnismäßig gering. Bei IaaS kann ein Nutzer über klar definierte Schnittstellen (z.B. Browser-gestützt oder durch Programmierschnittstellen[3]) virtuelle Maschinen erstellen. Dabei hat er beispielsweise die Wahl zwischen unterschiedlichen Gastbetriebssystemen und Ressourcenkonfigurationen (Art und Anzahl von Prozessoren, Größe des Arbeitsspeichers, Netzwerkanbindung).

Diese Leistungen schlagen sich ebenfalls in der an der Nutzungsdauer orientierten Preisgestaltung nieder. Nach der Bereitstellung dieser VMs ist der Nutzer sich selbst überlassen, was z.B. die Installation zusätzlicher Software (Datenbanken, eigene lizensierte Software, Programmierumgebungen) betrifft.

Bei dem nächsthöheren Dienstmodell *Platform as a Service (PaaS)* nimmt der Abstraktionsgrad zu. Der Fokus liegt hier auf der Bereitstellung einer Entwicklungsplattform für die Umsetzung und den Betrieb eigener Software. Diese Plattform bietet je nach Anbieter Werkzeuge für unterschiedliche Programmiersprachen, Datenbanken, persistenten Speicher und Methoden zur Ausfall- und Datensicherheit. Die Stärke einer solchen Plattform liegt dann zum einen in der einfachen abstrahierten Nutzung dieser Werkzeuge, die es dem Entwickler möglich macht, seine Applikationen in einer wohldefinierten Testumgebung mit geringen Kosten zu implementieren.

Zum anderen liegt sie in der hohen Skalierbarkeit während des späteren Betriebs. So gibt es bei PaaS-Anbietern Mechanismen zur automatischen Kopplung der Arbeitslast, die bei der späteren Nutzung der Applikation durch eine schwankende Nutzerzahl entsteht, und der flexiblen Anbindung bzw. Freigabe von Ressourcen zur Bewältigung der Arbeitslast. Dies kann z.B. die Zuteilung von zusätzlichem Speicherplatz zu einer Datenbank oder die Instanziierung weiterer virtueller Maschinen für die Lastverteilung umfassen. Die detaillierte Konfiguration des unterliegenden Systems auf Infrastrukturebene ist dabei weder für den Entwickler noch für die Nutzer der Applikation von vorrangigem Interesse, solange Dienstgü-

[3] Application Programming Interface (API).

tekriterien (wie Zugreifbarkeit, Zugriffsgeschwindigkeit oder Datensicherheit), die durch die aufgesetzte Plattform bereitgestellt werden sollen, erfüllt werden.

Im höchsten Dienstmodell *Software as a Service (SaaS)* wird Anwendungssoftware für die direkte Nutzung zur Verfügung gestellt. Auf diesem Abstraktionsgrad beschränken sich die Kontrollmöglichkeiten nach Mell & Grance [85] lediglich auf begrenzte nutzerspezifische Einstellungen der Anwendungssoftware. Alle darüber hinausgehenden Anpassungen der Plattform- und Infrastrukturschicht werden ausnahmslos vom Anbieter vorgegeben.

Im Hinblick auf die Nutzung einer Cloud für die bedarfsorientierte Erweiterung lokaler HPC-Ressourcen ist in dieser Arbeit in erster Linie das IaaS-Dienstmodell von Interesse. Schließlich geht es auf technischer Ebene vor allem darum, virtualisierte Ressourcen in das eigene lokale Scheduling-System zu integrieren. Das betrifft die Anpassung von angebotenen VM-Abbildern an die eigene Umgebung (Betriebssystem, Bibliotheken, Simulationssoftware), die Erstellung virtueller Netze und Verbindung mit dem lokalen Netz, sowie die Bekanntmachung der fremden Maschinen beim lokalen Scheduler. Aus diesen Gründen ist eine umfassende Konfiguration der einzubindenden virtuellen Maschinen auf Infastrukturebene unabdingbar und daher beziehen sich die nachfolgenden Beschreibungen stets auf das IaaS-Dienstmodell.

Neben den zuvor erwähnten Dienstmodellen ist auch die Ausrichtung einer Cloud in Bezug auf Absicherung und Zugangswege von Interesse. Neben einigen Spezialformen definieren Badger et al. [3] zwei sehr gegensätzliche Formen einer Cloud. Die erste Form, die sogenannte *Private Cloud*, stellt hierbei die restriktivste Form einer Cloud dar. Zwar ist der Cloud-typische virtualisierte Ressourcenpool auch hierbei über ein Netzwerk miteinander verbunden, er ist jedoch ausschließlich für Nutzer innerhalb einer einzelnen Organisation bzw. innerhalb eines lokalen Netzes (LAN) zugänglich. Ein Zugang von außerhalb (z.B. über ein Weitverkehrsnetz) kann über ein virtuelles privates Netzwerk (VPN) realisiert werden. Im Gegensatz zum physischen Betrieb der lokalen Ressourcen erleichtert die Virtualisierung hierbei die Wartung derselben (einfacher Austausch von Hardware ohne Neuinstallation von Software) und ermöglicht eine Konsolidierung der Dienste (mehrere Gastbetriebssysteme pro physischer Ressource). Gleichzeitig werden den Nutzern so eine Möglichkeit zur Anpassung ihres Betriebssystems (Anpassung des eigenen VM-Abbilds) sowie automatische Methoden für die Datensicherheit (Erstellung und redundante Speicherung von Snapshots) gegeben. Das in dieser Arbeit verwendete und in Abschnitt 2.2 erläuterte virtualisierte MPP-System stellt eine solche private Cloud dar.

Im Gegensatz dazu steht die *Public Cloud*. Hierbei greifen (verhältnismäßig viele) Nutzer in der Regel über ein Weitverkehrsnetz auf die Cloud zu. Charakteristisch für diese Form einer Cloud ist eine hohe Fluktuation von virtuellen Maschinen und Nutzern aus Betreibersicht und ein hohes Maß an Skalierbarkeit bis hin zu dem Eindruck potentiell unendlicher Ressourcen aus Nutzersicht. Der sehr einfache Zu-

gang[4] und die hohe Skalierbarkeit auf der einen Seite gehen aufgrund technischer Gegebenheiten mit einer limitierte Kontrolle der Daten durch den Nutzer einher. So zeichnet sich eine Public Cloud laut Badger et al. [3] auch dadurch aus, dass die Arbeitslast des Nutzers jederzeit ohne sein Wissen zwischen unterschiedlichen physischen Standorten verschoben werden und auch eine Löschung von Daten seinerseits nicht tatsächlich zu einer Löschung der Daten in der Cloud führt.

Bei einer *Hybrid Cloud* wiederum handelt es sich um einen Zusammenschluss unterschiedlicher Cloud-Formen zu einer gemeinsamen Ressourcenlandschaft. So können die Stärken einer privaten Cloud-Umgebung (z.B. restriktiver Zugang einer limitierten Nutzergemeinschaft, Resistenz gegenüber Ausfall des Weitverkehrsnetzes oder höhere Datensicherheit) mit den Vorzügen einer öffentlichen Cloud hinsichtlich der Skalierbarkeit kombiniert werden. Während das Cloud Scheduling innerhalb einer einzelnen Public bzw. Private Cloud im Kern die Zuteilung von physischen Ressourcen zu virtuellen Maschinen, den konsolidierten Betrieb mehrerer virtueller Maschinen innerhalb eines physischen Servers oder Mechanismen zur Ausfallsicherheit wie Platzierungsstrategien für die Speicherung redundanter Snapshots virtueller Maschinen umfasst, kommen durch den Zusammenschluss mehrerer Clouds weitere Aspekte zum Scheduling hinzu. So muss die Auslagerung von Arbeitslast aus einer privaten (lokalen) Cloud in die öffentliche Cloud gesteuert werden. Dieses sogenannte *Cloud Bursting* [3] wird z.B. eingesetzt um Managementaufgaben (Backup, Datenwiederherstellung) aus dem eigentlichen Produktivbetrieb herauszulösen oder um in Zeiten hoher Last — wie in unserem Anwendungsfall — eine Erweiterung der lokalen Ressourcen zur Lastverteilung und Performanzsteigerung durchzuführen. Um zu betonen, dass es in dieser Arbeit vornehmlich, um die Kosten-Nutzen-effiziente Auslagerung von Jobs auf Cloud-Ressourcen geht, wurde der Begriff *Hybrides Cloud Scheduling* zur Klassifizierung der Ansätze gewählt.

[4]Bei vielen Anbietern (z.B. Amazon) reicht bereits das Anlegen eines Accounts und die Hinterlegung von Kreditkartendaten und nach wenigen Minuten ist ein virtualisiertes Netzwerk mit mehreren Maschinen erstellt.

Kapitel 10

Grundlagen des hybriden Cloud Schedulings

Für das Verständnis des hybriden Cloud Schedulings ist es notwendig, zunächst auf die technischen Voraussetzungen für die Realisierung hybrider Clouds einzugehen und gängige Aspekte bzw. existierende Werkzeuge zur praktischen Umsetzung vorzustellen. Im Anschluss daran werden technische und algorithmische Vorarbeiten unterschiedlicher Forschergruppen bei der Ausführung wissenschaftlicher Anwendungen auf der Cloud und insbesondere im Betrieb hybrider Infrastrukturen näher erläutert.

10.1 Technische Voraussetzungen

Die von Mell & Grance [85] für das Cloud Computing typische Bereitstellung von Cloud-Ressourcen mit minimalem Aufwand für Nutzer und Anbieter wird in der Praxis durch anbieterspezifische Werkzeuge realisiert. So bieten nahezu alle IaaS-Anbieter ihren Nutzern webbasierte Portale an, in denen sie sich nach Anlegen eines eigenen Nutzerzugangs selbst Instanzen virtueller Maschinen unterschiedlicher Instanztypen (die sich auf die unterliegende Hardware beziehen) und Abbilder (die sich auf Betriebssystem und vorinstallierte Softwarebibliotheken beziehen) erstellen können. Übliche zusätzliche Verwaltungsmöglichkeiten sind dabei:

- das Starten und Stoppen von Instanzen. Das ist vor allem deswegen wichtig, da bei vielen Anbietern nur für eine laufende Instanz Kosten anfallen.
- die Netzwerkkonfiguration von Instanzen. Auf diese Art und Weise können gesamte Cluster mit eigenem Subnetz aus einzelnen VM-Instanzen erzeugt werden.
- der Abruf von Monitoring-Daten zur Auslastung der Instanzen, zum Speicherverbrauch, zur Netzwerkkommunikation oder zu entstehenden Kosten durch den Betrieb der Instanzen. Dadurch hat der Nutzer stets im Blick,

wie effizient seine gemieteten Instanzen genutzt werden und welche Kosten er dafür aufwenden muss.

- die automatische Rekonfiguration. Manche Anbieter bieten auf diesem Wege bereits die Möglichkeit an, eine lastorientierte Erweiterung des virtuellen Clusters vorzunehmen bzw. zu konfigurieren.

Gerade im Hinblick auf eine automatisierte Verwaltung von Cloud-Ressourcen sind die Programmierschnittstellen (API) von Interesse, die es Entwicklern ermöglichen, die zuvor genannten Verwaltungsmöglichkeiten auch über eigene Programme zu steuern. Prominente Beispiele für solche APIs sind jene von Amazon[1] oder der Firma Rackspace[2].

In der Regel sind diese Schnittstellen stark auf den entsprechenden Anbieter zugeschnitten, so dass man bei der direkten Nutzung einer API zur Verwendung von Cloud-Ressourcen Gefahr läuft, für die Zukunft an einen bestimmten Anbieter gebunden zu sein. Dieser Sachverhalt wird auch als *Cloud lock-in*-Problem bezeichnet. Aus diesem Grund gibt es zahlreiche unterschiedliche Ansätze und Architekturimplementierungen, welche sich zum Ziel gemacht haben, eine einheitliche Schnittstelle zu möglichst allen IaaS-Anbietern zu erstellen.

So erheben sowohl das Unternehmen *Eucalyptus Systems* mit ihrer Cloud Computing Software Plattform *Eucalyptus*[3], das Projekt *OpenNebula*[4] mit seiner gleichnamigen Software als auch die Kooperation aus Rackspace und der amerikanischen Weltraumbehörde *NASA* mit ihrer Softwarelösung *OpenStack*[5] den Anspruch, eine quelloffene Lösung für das Cloud lock-in-Problem anzubieten. Alle drei Softwarepakete erlauben es einem Entwickler jegliche Form von Clouds (öffentlich, privat, hybrid) entweder auf Basis lokaler Ressourcen oder innerhalb kommerzieller IaaS-Anbieter umzusetzen. Zusätzlich soll das vom *Open Grid Forum* standardisierte *Open Cloud Computing Interface (OCCI)*[6] eine umfassende Schnittstelle für alle der zuvor genannten Softwarelösungen bereitstellen.

Ein wichtiger Punkt bei der Nutzung von Cloud-Ressourcen stellen mögliche Sicherheitsanforderungen dar. So wird die Auslagerung oder Ergänzung der eigenen Infrastruktur erst dann interessant, wenn sich die Anforderungen der ausgelagerten Services oder Rechenjobs im Hinblick auf Datensicherheit, Datenschutz oder eine bestimmte Sicherheitsinfrastruktur erfüllen lassen. Die verwendeten Sicherheitskonzepte erstrecken sich dabei je nach Anbieter vom einfachen Zugang mit Benutzername und Passwort, über die Verwendung von RSA-Schlüsselpaaren und zertifikatsbasierte Sicherheitsinfrastrukturen bis hin zu *Virtual Local Area Networks (VLAN)* zur Einrichtung abhörsicherer Subnetze.

[1] http://aws.amazon.com/documentation/, Zugriff: 16. Januar 2013.
[2] http://docs.rackspace.com/, Zugriff: 16. Januar 2013.
[3] http://www.eucalyptus.com/, Zugriff: 16. Januar 2013.
[4] http://opennebula.org, Zugriff: 16. Januar 2013.
[5] http://www.openstack.org/, Zugriff: 16. Januar 2013.
[6] http://occi-wg.org/about/specification/, Zugriff: 16. Januar 2013.

Eine technische und arbeitslastreaktive Umsetzung der Erweiterung lokaler Ressourcen um Cloud-Ressourcen — wie in dieser Arbeit — konnte von Marshall et al. [84] auf Basis der *Nimbus Platform*[7] vorgestellt werden. Hierbei wurde eine dynamische Erweiterung aus einer prototypischen lokal betriebenen privaten Cloud [82] erweitert um zwei weitere Cloud-Anbieter (FutureGrid, Amazon EC2). Die Arbeit umfasst sowohl die Instanziierung von virtuellen Maschinen auf den ergänzenden Cloud-Plattformen als auch die sogenannte *Kontextualisierung*. Bei dieser wird eine neue VM-Instanz ausgehend von einem Basis-Image durch automatische Installation von Softwarepaketen für die spätere Ausführung der Nutzlast und die Anbindung an das lokale Batch-System in die eigene Ressourcenlandschaft integriert. Als Bestätigung ihres Konzepts und zur Messung der Performanz führen die Autoren dabei Messungen aus in denen sie Jobs unterschiedlicher Parallelität einreichen, die dann dynamisch auf einen neu erstellten Subcluster innerhalb einer fremden Cloud-Umgebung zur Ausführung gebracht werden.

Ein ähnlicher Ansatz wurde von Ostermann et al. [97] vorgestellt, bei dem eine bestehende Grid-Infrastruktur zur Abarbeitung von wissenschaftlichen Workflows um die Anbindung privater und öffentlicher Cloud-Ressourcen erweitert wird. Die Instanziierung weiterer VM-Instanzen erfolgt auch hier dynamisch auf Basis von Arbeitslaständerungen. Die technische Machbarkeit der dynamischen Erweiterung um Cloud-Ressourcen ist somit offenbar gegeben. Im Folgenden werden daher eine Reihe von praktischen Umsetzungen wissenschaftlichen Rechnens in der Cloud beschrieben.

10.2 Wissenschaftliches Rechnen in der Cloud

Mit der Aussicht auf eine Alternative zum Rechnen wissenschaftlicher Anwendung auf dem eigenen Rechencluster bzw. einer Grid-Umgebung hat sich eine neue Forschungsdisziplin ergeben. Beim *Scientific Cloud Computing* geht es zum einen darum, bestehende Cloud-Infrastrukturen unterschiedlicher Cloud-Anbieter auf ihre Nutzbarkeit für wissenschaftliches Rechnen (insbesondere HPC) zu überprüfen und zum anderen darum, die Vor- und Nachteile einer solchen Auslagerung wissenschaftlicher Anwendungen in Bezug auf Skalierbarkeit und entstehende Kosten zu untersuchen.

So haben Ostermann et al. [96] beispielsweise durch Evaluation der — zu dieser Zeit — angebotenen Amazon-Instanztypen im Hinblick auf ihre Performanz bei Ausführung von Standardbenchmarks für HPC-Umgebungen (LINPACK) nachgewiesen, dass ein Verbund von VM-Instanzen in der Cloud durchaus in der Lage ist, an die Effizienz eines klassischen lokalen HPC-Clusters heranzureichen. Probleme treten immer dann auf, wenn es sich um sehr I/O-intensive Aufgaben handelt, da bei der Erstellung eines Rechnerverbundes in der öffentlichen Cloud von Amazon lediglich eine Region (Europe-West, US-East etc.) ausgewählt werden kann und die Instanzen dann über physische Server an unterschiedlichen Standorten ver-

[7] http://www.nimbusproject.org/, Zugriff: 16. Januar 2013.

teilt werden. Da dieses Problem den IaaS-Anbietern durchaus bewusst ist, werden zunehmend auch spezialisierte HPC-Instanztypen angeboten, die eine vergleichbare Leistung wie aktuelle lokale HPC-Cluster-Installationen erreichen sollen (z.B. Amazon EC2 cc2.8xl[8]).

Darüber hinaus existieren diverse Machbarkeitsstudien für die Nutzung von öffentlichen Clouds. So wurde von Lu et al. [78] mit der Portierung der bioinformatischen Anwendung *BLAST* für den Abgleich von Gensequenzen auf Microsofts *Azure*-PaaS-Cloud der Vorteil für hochskalierbare Anwendungen aufgezeigt, die über wenig bzw. gar keine Kommunikation zwischen einzelnen parallelen Instanzen verfügen (*Embarrassingly Parallel*).

Ähnliche Ansätze wurden von Fenn et al. [40] anhand einer Wettermodellsimulation und von Rehr et al. [103] vorgenommen, wobei Letztere eine komplette SaaS-Lösung ihrer Software *FEFF* zur Röntgenspektroskopie inklusive grafischer Oberfläche für das Einreichen von Simulationen auf Basis von Amazons EC2 realisiert haben.

Bei dem Ansatz von Deelman et al. [20], einen Workflow zur Analyse von Teleskopaufnahmen auf der Amazon EC2 auszuführen, wurde aus Kostengründen lediglich auf eine Simulation der Cloud-Komponenten gesetzt. Dennoch gibt diese Arbeit einen guten Eindruck über die zu erwartenden Kosten, die bei Ausführung wissenschaftlicher Anwendungen im Vergleich mit dem Betrieb auf einer lokalen Cluster-Umgebung zu erwarten sind.

10.3 Existierende Ansätze des Cloud Schedulings

Neben der technischen und praktischen Portierung von wissenschaftlichen Anwendungen auf und Erweiterung lokaler Ressourcenräume um Cloud-Ressourcen, sind für das hybride Cloud Scheduling in erster Linie algorithmische Ansätze von Interesse, die darüber entscheiden wann und wie viele Cloud-Ressourcen zur Bearbeitung der Arbeitslast eingesetzt werden sollen.

Diesbezüglich untersuchten Marshall et al. [82] einfache Heuristiken für die Instanziierung von VMs auf Basis sequentieller Jobs. Der Fokus lag dabei auf der Analyse des Verhältnisses zwischen der genutzten Rechenzeit auf der Cloud und der verlorengegangenen Managementzeit für das Instanziieren bzw. Abschalten von virtuellen Maschinen. Eine Analyse der durch die Nutzung entstehenden Kosten wurde nicht durchgeführt.

Ausgehend von den zuvor genannten Heuristiken widmeten sich Marshall et al. [83] zwei Jahre später der Entwicklung kostensensitiver Leihverfahren, die unter anderem auch Abrechnungsintervalle öffentlicher IaaS-Anbieter berücksichtigten. Dabei lag der Fokus auf der Balancierung der Arbeitslast unter mehreren gleichzeitig verwendeten Cloud-Anbietern. Hierbei setzten sie auch Offline-

[8] Nach Presseberichten von Morgan [87] und Cloer [15] haben Vertreter von Amazon auf der Supercomputing Conference 2011 mit einem Verbund von 1064 der neuen Instanztypen bei dem LINPACK-Benchmark Platz 42 auf der TOP500-Liste erreicht.

Optimierungsverfahren ein, um vorteilhafte Lösungen bei der Verteilung ihres Eingabe-Workloads auf unterschiedliche Cloud-Anbieter zu ermitteln.

Genaud & Gossa [57] wiederum analysierten unterschiedliche Heuristiken im Hinblick auf die VM-Zuteilung zu extern auszuführenden Jobs (exklusive Zuteilung von VMs zu Jobs, Einreihen von Jobs in Queues bestehender VMs) und konnten dabei feststellen, dass das eigentlichen Mapping von Jobs auf die VMs immer zu ähnlichen Kosten führt. Daraus ergibt sich, dass Scheduling-Lösungen mit unterschiedlichen Kosten nur über eine zielgerichtete Auswahl lokaler Jobs zur Portierung in die Cloud erfolgen kann.

Mao et al. [81] erstellten auf Basis von Vergangenheitsdaten (Joblaufzeiten, Abarbeitungsgeschwindigkeiten von Jobs auf unterschiedlichen VM-Instanztypen) einen sogenannten VM-Instanzplan mit dem Ziel, möglichst alle Deadlines eines Online-Deadline-Scheduling-Problems zu erfüllen. Als Nachweis der Funktionsfähigkeit ihrer Architektur evaluierten sie ihr Gesamtsystem durch die Anbindung von Microsofts Azure Cloud und unter Einsatz einer Anwendung zur Bearbeitung von Satellitenaufnahmen. Die Analyse geschah vornehmlich im Hinblick auf die Reaktion ihres Systems auf Veränderungen der vorgegebenen Deadlines oder der Anzahl pro Zeitintervall submittierter Jobs. Eine wirkliche Online-Optimierung in Bezug auf ein günstiges Kosten-Nutzen-Verhältnis bei Auslagerung einer bestimmten Untermenge von Jobs lag allerdings auch in diesem Fall nicht vor.

Auch die von Dias de Assuncao et al. [21] veröffentlichte Arbeit setzt auf die Evaluation vieler unterschiedlicher Kombinationen (*Strategy Sets*) aus lokalen Scheduling-Algorithmen und Abgabestrategien mit dem Ziel, die Deadlines eines modellbasiert generierten Workloads einzuhalten. Im Vordergrund steht hierbei die Auswirkungen auf Kosten und Scheduling-Performanz (AWRT) sowie verpasste Deadlines der einzelnen Strategien, wenn Modellparameter des Workload-Modells wie die Zeit zwischen Jobeinreichungen, die durchschnittliche Joblaufzeit oder Jobparallelität variiert werden.

In dieser Arbeit soll es vielmehr darum gehen, das Optimierungspotential im Hinblick auf ein ausgewogenes Kosten-Nutzen-Verhältnis zu ermitteln und zu überprüfen in wie weit Online-Lernverfahren mit und ohne Integration von Expertenwissen hinsichtlich dieses Kriteriums an Offline-Lernverfahren heranreichen können. Zusätzlich soll sie Betreibern von Rechenzentren Ideen für die strategische Umsetzung einer Erweiterung ihrer lokalen Ressourcen (private Cloud) um öffentliche Cloud-Ressourcen geben, bei der es um die Zielsetzung geht, ein bestimmtes Verhältnis von Performanzgewinn und dafür aufzubringenden Kosten zu erreichen.

Kapitel 11

Eigene Ansätze für das hybride Cloud Scheduling

Die Einführung eigener hybrider Cloud Scheduling-Ansätze erfolgt in mehreren Schritten. Zunächst wird das aus Abschnitt 2.2 bekannte Modell eines MPP-System in Abschnitt 11.1 erweitert. Dies schließt beispielsweise die Spezifizierung modellabhängiger Einschränkungen und Annahmen sowie neue Metriken für die Berechnung der Scheduling-Performanz mit ein. Dabei werden getroffene Entscheidungen hinsichtlich der Annahmen bzw. Verwendung unterschiedlicher Strategieparameter direkt evaluiert und diskutiert, um eine ausreichend stabile Basis für die darauf folgenden Lernansätze zu gewährleisten.

Die in Abschnitt 11.2 untersuchte mehrkriterielle Optimierung von Cloud Scheduling-Entscheidungen dient im Anschluss als Orientierung für die in Abschnitt 11.3 beschriebenen Online-Lernverfahren.

Alle folgenden Untersuchungen basieren auf Modellannahmen und Ergebnissen, die bereits vorangehend von Fölling & Hofmann [47] veröffentlicht wurden.

11.1 Modell und Bewertungsgrößen für das hybride Cloud Scheduling

Die Erweiterung des lokalen Ressourcenraums erfolgt in Anlehnung an die Definition des homogenen MPP-Systems auf heterogenen Ressourcen (vgl. Abschnitt 2.2 auf S. 14). Das Scheduling-System einer privaten Cloud weist einer statischen Zahl virtueller Maschinen parallele Rechenjobs zu, welche zuvor durch die Nutzer eingereicht wurden. Wie in Abbildung 11.1 wird die private Cloud um eine öffentliche Cloud-Umgebung erweitert, die eine dynamisch veränderbare Zahl von virtuellen Maschinen für die Abarbeitung von Rechenjobs bereitstellt. Diese einseitige Auslagerung von Arbeitslast von der lokalen in die entfernte Cloud wird dabei von der sogenannten Leihstrategie gesteuert.

Analog zu den Annahmen bei der Erweiterung von Ressourcen durch Compu-

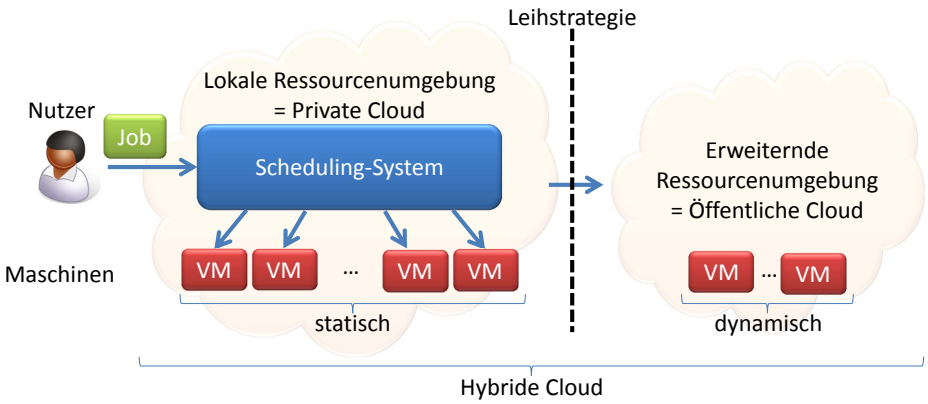


ABBILDUNG 11.1: *Hybrides Cloud-Modell*

tational Grids (vgl. Teil II) werden auch hier alle virtuellen Maschinen als homogen in ihrer Leistung angesehen.

Das gilt insbesondere über Cloud-Grenzen hinweg. Ein Rechenjob läuft demnach auf jeder Untermenge von virtuellen Maschinen (egal ob innerhalb der privaten oder öffentlichen Cloud) gleich schnell.

Ein solches Modell wird von Iosup et al. [66] als *source-like performance*-Modell beschrieben, bei dem sich die Geschwindigkeit der Abarbeitung von Rechenjobs an der Geschwindigkeit der lokalen Maschinen orientiert. Die Alternative bestünde in der Festlegung einer Funktion, die bei der Simulation eines Workloads jedem enthaltenen Job eine neue Bearbeitungsdauer zuweist, sollte er nicht auf den lokalen Ressourcen sondern in der öffentlichen Cloud ausgeführt werden. Eine solche Funktion ist in der Praxis nur schwer zu ermitteln, da die Bearbeitungsdauer einer Anwendung selbst bei gleicher Implementierung von einer großen Anzahl verschiedener Faktoren abhängt (Architektur der physischen Ressourcen; eingesetzte Virtualisierungslösung; Art und Länge der Eingabe; Grad der Parallelisierung, sofern dieser veränderbar ist; Konfiguration der Netzwerkinfrastruktur uvm.).

Des Weiteren wird für die Bereitstellung von virtuellen Maschinen keine *Setup*-Zeit modelliert — also Zeit für das Instanziieren und Booten einer virtuellen Maschine. Zum einen sind diese Zeiten in der Praxis sehr stark abhängig vom IaaS-Anbieter und instanziiertem Image und bewegen sich nach Iosup et al. [66] im Mittel zwischen 60 Sekunden[1] und ca. 9 Minuten[2]. Zum anderen existieren schon jetzt vielversprechende Ansätze [71, 18] für die Reduktion der Setup-Zeiten, so dass in naher Zukunft in Kombination mit leistungsfähigerer Daten- und Netzinfrastruktur bei den IaaS-Anbietern damit zu rechnen ist, dass diese keine große Relevanz mehr für das hybride Cloud Scheduling haben werden. Dies gilt insbesondere, da die durchschnittlichen Joblaufzeiten für die betrachteten Workloads zwischen 70 Minuten (SDSC00) und ca. 3 Stunden (CTC) liegen.

[1] Gilt für nahezu alle Amazon EC2-Instanzen.
[2] Bezieht sich auf Erhebungen beim Anbieter *GoGrid*.

Auf Basis derselben Argumente aus Abschnitt 7.2.1 (S. 63) wird auch in dem Cloud Scheduling Modell auf eine Berücksichtigung der Datentransfers zwischen den Cloud-Umgebungen verzichtet, die bei Auslagerung der Arbeitslast entstehen können. Darüber hinaus gilt weiterhin die Einschränkung, dass ein Rechenjob nicht über Cloud-Grenzen hinweg ausgeführt werden darf. Zwar würde eine virtuelle Infrastruktur mit source-like-performance-Modell diese *Multi-Site-Ausführung* prinzipiell ermöglichen, doch fände die Kommunikation zwischen parallelen Prozessen in dem Fall über Weitverkehrsnetze statt. Dies führt je nach Aufbau der Anwendung (z.B. bei Prozessen mit verstärkter Interprozesskommunikation) in der Realität zu erheblichen Leistungseinbrüchen und wird daher auch nicht vom Modell unterstützt.

Im Gegensatz zu den Erweiterungen über Computational Grids handelt es sich bei dem hier betrachteten hybriden Cloud Scheduling um einen einseitigen Austausch von Rechenjobs, durch den Kosten für die Ausführung der Jobs entstehen. Dies wiederum macht die Definition eines geeigneten Kostenmodells nötig.

11.1.1 Kostenmodell

Je nach Anbieter existieren unterschiedliche Kostenmodelle für die Nutzung ihrer Ressourcen. In der Regel lassen sich diese in drei verschiedene Klassen unterteilen.

So verlangen IaaS-Anbieter beispielsweise einen bestimmten Geldbetrag für eine laufende virtuelle Maschine. Dieser Betrag ist dabei abhängig von dem gewählten Image und dem unterliegenden Host-System und ist einer bestimmten Taktung (pro Stunde, pro Tag) unterworfen. Dabei ist es unerheblich, ob die virtuelle Maschine gerade produktiv genutzt wird, bootet, herunterfährt oder einfach nur eingeschaltet ist.

Die zwei anderen Klassen hängen mit den verbundenen Daten zusammen und beschreiben einerseits eine ebenfalls getaktete und vom Datenvolumen abhängige Gebühr für die Speicherung von Daten. Andererseits werden volumenabhängige Gebühren für den Transfer von Daten in die oder aus der kommerziellen Cloud verlangt.

Da Jobdaten in unserem Fall nicht Teil des Modells sind[3], fallen auch keine Kosten der letzten beiden Klassen an und finden demnach keine Berücksichtigung im Modell. Des Weiteren sind durch den Verzicht auf schwierig zu modellierende Setup-Zeiten und unter der Annahme, dass ein Cloud-übergreifendes homogenes Maschinenmodell eingesetzt wird, lediglich die Produktivzeiten der virtuellen Maschinen zu betrachten. Eine Berücksichtigung von anbieterspezifischen Taktungen wird aufgrund der generellen Anwendbarkeit der hier diskutierten Ansätze ebenfalls nicht vorgenommen.

Bei dem hier angenommenen Kostenmodell wird also von einer idealisierten Cloud-Umgebung mit sekundengenauer Abrechnung für die produktive Nutzung ausgegangen. Dadurch beschränken sich die, durch die Erweiterung um kommerzi-

[3]Die Datenvolumina der Jobs in den betrachteten Workloads sind nicht bekannt.

elle Cloud-Ressourcen, entstehenden Kosten auf das Ressourcenprodukt (vgl. Abschnitt 3.2.2 auf S. 23) der in die Cloud ausgelagerten Jobs in CPU-Sekunden.

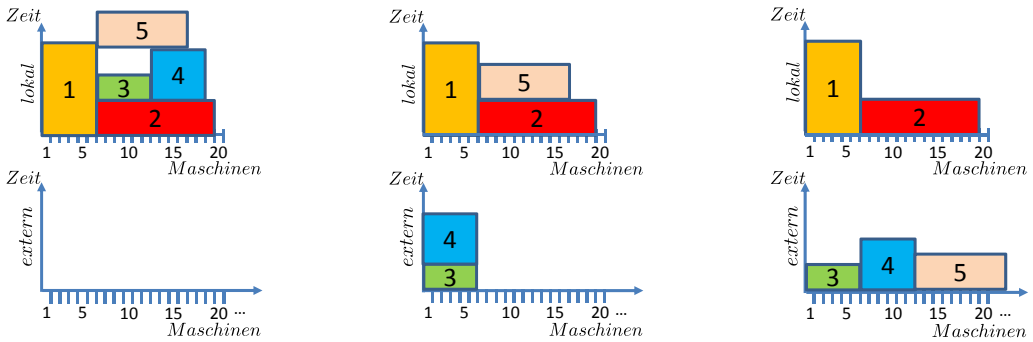
11.1.2 Die Bilanz als Kosten-Nutzen-Funktion

Ziel der Untersuchungen in diesem Teil der Arbeit ist die Optimierung des Kosten-Nutzen-Verhältnisses bei der arbeitslastabhängigen Erweiterung der lokalen Ressourcen. Neben den zuvor beschriebenen Kosten, die bei der Auslagerung von Jobs in die kommerzielle Cloud entstehen, muss also zusätzlich noch eine Definition des daraus resultierenden Nutzens erfolgen. Als Grundlage soll in dieser Arbeit die in Abschnitt 3.2.5 (S. 25) eingeführte Gesamtwartezeit (TWT) dienen.

Hierbei soll in Hochlastsituationen durch gezielte Auslagerung von Jobs in die Cloud eine Reduzierung der Wartezeit erreicht werden. Wann immer ein Job aufgrund belegter lokaler Ressourcen auf seine Ausführung warten müsste, entscheidet die Leihstrategie darüber, ob der Job umgehend innerhalb der öffentlichen Cloud gestartet werden soll, auch wenn dies zu weiteren Kosten führt.

In einer produktiven Umsetzung bedeutet dies, dass die Nutzergemeinschaft zu Spitzenzeiten insgesamt weniger lange auf den Beginn der Bearbeitung ihrer Jobs warten müsste. Gleichzeitig würde diese Leistungssteigerung des Scheduling-Services nicht über den Ausbau der lokalen physischen Ressourcen sondern durch eine flexible „pay-per-use“-Lösung erreicht werden. Die verbessere Wartezeit und damit verbundene Kosten müssen zur Berechnung eines Kosten-Nutzen-Verhältnisses stets relativ betrachtet werden, da sie sonst aufgrund unterschiedlicher Wertebereiche nicht miteinander vergleichbar sind.

Abbildung 11.2 liefert ein Beispiel für die relative Berechnung der beiden Entscheidungsmetriken.



(a) Extremfall mit lokalem Scheduling resultiert in TWT_{ref} und $C = 0$.

(b) Zwischenlösung mit eingeschränkter Auslagerung resultiert in $\text{TWT}_{beispiel}$ und $C_{beispiel}$.

(c) Extremfall mit unbegrenzter Auslagerung resultiert in TWT = 0 und C_{ref}.

ABBILDUNG 11.2: *Beispiel für unterschiedliche Schedules bei gleicher Jobmenge mit unterschiedlichen Ergebnissen in Gesamtantwortzeit (TWT) und Kosten (C).*

In dem Beispiel wird eine Menge zuvor eingereichter Jobs (1 bis 5)[4] unter drei verschiedenen Konfigurationen für die öffentliche Cloud geplant. Dabei wird sowohl für das lokale als auch das entfernte Scheduling der gleiche lokale Scheduling-Algorithmus eingesetzt (sowohl FCFS als auch EASY würden zu dem gleichen Ergebnis führen).

In Abbildung 11.2(a) werden die Jobs lediglich lokal ausgeführt. Dies führt zwar zu 0 Kosten aber zu einem Maximum an Gesamtwartezeit TWT_{ref}, da die Jobs 3 bis 5 auf ihre Ausführung warten müssen.

Der gegenteilige Extremfall in Abbildung 11.2(c) geht von einer unbegrenzten öffentlichen Cloud-Umgebung aus, in der alle potenziell wartenden Jobs direkt parallel gestartet werden können. Zwar liegt in diesem Szenario keine Wartezeit mehr vor, doch sind die daraus resultierenden Kosten C_{ref} maximal, da die Jobs 3 bis 5 nun ausgelagert werden.

In der Zwischenlösung in Abbildung 11.2(b) hingegen hat die Leihstrategie entschieden, dass die Menge auszulagernder Jobs zwar limitiert aber nicht leer ist und die beiden Jobs 3 und 4 sequentiell ausgeführt werden sollen. Dies führt zu den entsprechenden Werten für die Gesamtwartezeit $\mathrm{TWT}_{beispiel}$ und Kosten $C_{beispiel}$.

Die verbesserte Wartezeit (TWT_I) ergibt sich nun als die prozentuale Reduzierung der Wartezeit gegenüber der maximalen Wartezeit $\mathrm{TWT}_I = 100 - \frac{\mathrm{TWT}_{beispiel} \cdot 100}{\mathrm{TWT}_{ref}}$. Gleichzeitig lassen sich ebenfalls die prozentualen Kosten ($\%C$) mit Bezug zu den Maximalkosten (C_{ref}) berechnen. Auf Basis dieser beiden — auf denselben Wertebereich genormten — Größen lässt sich die Bilanz (BIL) einer Zwischenlösung durch die Gleichung (11.1) berechnen.

$$\mathrm{BIL} = \mathrm{TWT}_I - \%C \tag{11.1}$$

Die Gleichgewichtung beider Größen vorausgesetzt, indiziert eine positive Bilanz hierbei einen höheren Nutzen durch die Erweiterung als die relativen Kosten, die dadurch entstehen. Durch die Bilanz lassen sich nun unterschiedliche Cloud Scheduling-Entscheidungen miteinander vergleichen, wobei diejenige mit der höchsten Bilanz gleichzeitig die vielversprechendste Lösung darstellt.

Für die späteren Untersuchungen ist es wichtig, zu erwähnen, dass es sich bei der Bilanzfunktion um eine A priori-Aggregation (vgl. Abschnitt 5.2 auf S. 41) der beiden gegensätzlichen Bewertungsfunktionen aus Wartezeit und Kosten handelt, bei denen beide Faktoren gleichgewichtet eingehen. Für eine gezielte Analyse unterschiedlich gewichteter Kompromisslösungen[5] sei auf die mehrkriterielle Optimierung in Abschnitt 11.2 verwiesen.

[4] Die Jobs wurden in der Reihenfolge ihrer Benamung eingereicht und befinden sich bereits in der Queue.

[5] Dies könnte für den Betreiber einer privaten Cloud von Interesse sein, wenn dieser entweder eine stärkere Wartezeitreduzierung ungeachtet der dadurch entstehenden Kosten präferiert oder umgekehrt.

11.1.3 Diskretisierung der Cloud Scheduling Entscheidungen

Bewertungsfunktionen wie Gesamtwartezeit, Kostenfunktion oder die daraus resultierende Bilanzfunktion des vorherigen Abschnitts lassen sich immer nur auf bestimmten Zeitintervallen und den in diesem Zeitraum betrachteten Jobs berechnen. Eine Leihstrategie, die auf Basis solcher Größen Entscheidungen bzgl. des Leihverhaltens treffen soll, benötigt daher einen zeitlichen Auslöser für die Entscheidungsfindung. Auf die Art und Weise wird das Online-Problem für das Leihen von virtuellen Maschinen in ein deterministisches Scheduling-Problem mit festem Betrachtungszeitraum übersetzt. Ein solcher Ansatz findet aufgrund der besseren Skalierbarkeit manchmal auch in gängigen Batch-Systemen (vgl. Abschnitt 2.1 auf S. 10) statt, bei denen die eingesetzten Algorithmen nicht bei Eintritt jedes einzelnen Jobs sondern in Zeitintervallen (z.B. alle 30 Sekunden) ausgeführt werden.

Für die Wahl eines geeigneten Zeitintervalls gelten zwei gegensätzliche Kriterien. Zum einen sollten innerhalb der Intervalle genügend Jobs liegen, damit unterschiedliche Entscheidungen auch zu möglichst unterschiedlichen Ausprägungen der Bewertungsfunktionen führen, denn nur in dem Fall kann der später eingesetzte Ansatz zum Lernen vorteilhafter Entscheidungen zwischen einzelnen Aktionen unterscheiden. Zum anderen dürfen die Entscheidungsintervalle nicht zu groß werden, da sonst die Lerngeschwindigkeit und damit auch die Qualität der Entscheidungsfindung abnimmt, weil zu wenig Messpunkte auf einem Workload beschränkter Länge (in unserem Fall vereinheitlichte Aufzeichnung von 11 Monaten Länge; vgl. Abschnitt 4.1) bestehen.

Um eine fundierte Entscheidung treffen zu können, wurden alle fünf Lastaufzeichnungen bzgl. der Zahl von Jobeinreichungen bei drei verschiedenen gewählten Zeitintervallen untersucht. Tabelle 11.1 zeigt die bei der Untersuchung festgestellten Mittelwerte für die eingereichten Jobs und die Menge an Intervallen, in denen gar keine Jobs eingereicht wurden in Abhängigkeit der Intervallgröße und daraus resultierenden Anzahl von Intervallen. Gemessen wurden Intervalle der Länge von einem halben, einem und zwei simulierten Tagen. Nach Betrachtung der Ergebnisse wurde entschieden, dass die Intervalllänge für alle weiteren Untersuchungen mit einem Tag gewählt werden soll, da diese Einstellung einen guten Mittelweg aus der Anzahl unterschiedlicher Intervalle (340) und innerhalb der Intervalle eingereichter Jobs (zwischen 82 und 226 Jobs pro Intervall) darstellt.

Auch die Anzahl von Intervallen, in denen keine Jobs eingereicht werden und das Scheduling demnach nur von dem Backlog der vorhergehenden Intervalle abhängt, ist außer beim SDSC05[6] verhältnismäßig gering.

Des Weiteren besitzen viele Aufzeichnungen einen sich wiederholenden Tages- und Wochenrhythmus[7]. Die Unterteilung in halbe Tage würde daher eine gezielte

[6] Die vergleichsweise große Anzahl von leeren Intervallen hängt mit der sehr langen Einschwingphase zusammen. So werden beim SDSC05 in den ersten 22 Tagen lediglich 10 von insgesamt 74 903 Jobs eingereicht.

[7] Unterschiedliche Einreichungen nach Tageszeit und wenig Einreichungen während des Wochenendes.

Trace	Länge (Tage)	# Intervalle	∅ Jobs/Intervall	# Intervalle ohne Jobs
KTH-11	$\frac{1}{2}$	680	40,93	14
	1	340	82,86	5
	2	170	166,54	1
SDSC00-11	$\frac{1}{2}$	680	42,97	22
	1	340	86,94	6
	2	170	174,85	2
CTC-11	$\frac{1}{2}$	680	112,86	7
	1	340	226,72	2
	2	170	455,30	0
SDSC03-11	$\frac{1}{2}$	680	95,65	20
	1	340	192,30	7
	2	170	384,44	1
SDSC05-11	$\frac{1}{2}$	680	109,6	78
	1	340	220,01	34
	2	170	441,02	15

TABELLE 11.1: *Analyse der Jobeinreichungen bei Unterteilung des Entscheidungsraums in Intervalle unterschiedlicher Längen.*

Untersuchung von Tagesunterteilung nötig machen (z.B. vormittags/nachmittags oder 8 bis 20 Uhr und 20 bis 8 Uhr), wohingegen eine Unterteilung in zwei Tage zu einem alternierenden Verhalten für die zweitägigen Wochenenden bei einer 7-Tage-Woche führen würde. Es macht also nach konzeptionellen Gründen und auf Basis der Jobeinreichungen Sinn, die Entscheidungsintervalle mit einem simulierten Tag zu wählen.

Durch die Unterteilung des Entscheidungsraums in n Intervalle von einem Tag erhält man abhängig von dem Wertebereich eines Strategieparameters (SP)[8] eine feste Anzahl verschiedener Strategiepfade. Angenommen die Menge an wählbaren Parameterausprägungen sei mit $\mathbb{SP}$ gegeben. In diesem Fall würde sich die Anzahl an verschiedenen Strategiepfaden durch $|\mathbb{SP}|^n$ ergeben, wobei alle Pfade der Form aus Abbildung 11.3 entsprechen würden.

Hierbei muss sich die Leihstrategie in Abhängigkeit der Bewertungsfunktionen auf einem Intervall $[t_i, t_{i+1}]$ mit $i \in [0, n-1]$ für eine neue Konfiguration des Parameters sp_{i+1} zum Einsatz im nächsten Intervall entscheiden[9].

Am Ende jedes Strategiepfades lässt sich unabhängig von den gewählten Entscheidungen auf jeden Fall ein Bilanzwert berechnen, da stets auch die beiden

[8] Alle nachfolgend beschriebenen Leihstrategien basieren stets nur auf einem einzelnen Strategieparameter. Daher werden die Strategiepfade an dieser Stelle auch nur mit Hilfe eines einzelnen Parameters skizziert.

[9] Das bedeutet, dass für die erste Entscheidung sp_0 kein vorheriges Intervall als Bemessungsgrundlage hinzugezogen werden kann und diese statisch definiert werden muss.

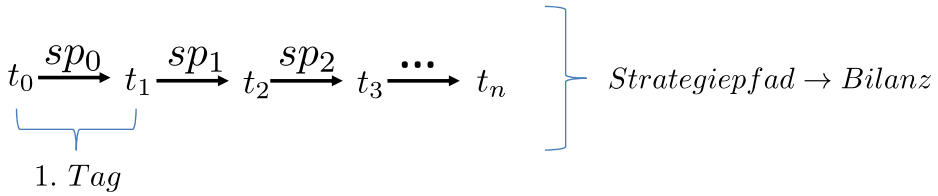


Abbildung 11.3: *Aufbau eines Strategiepfades.*

Extremfälle ganz ohne Ressourcenerweiterung und unbeschränkte Ressourcenerweiterung simuliert werden können.

11.1.4 Diskussion und Einführung verschiedener Strategieparameter

Die — durch eine sinnvolle Leihstrategie — erreichbaren Bilanzwerte sind stark abhängig vom eingesetzten Strategieparameter. Dies soll im Folgenden am Beispiel von zwei unterschiedlichen Strategieparametern untersucht werden.

Im ersten Fall entscheidet die Leihstrategie, ob ein Job ausgelagert werden soll oder nicht, mit Hilfe der maximalen Anzahl in der Austausch-Cloud aktiver virtueller Maschinen. Sollten für die Ausführung eines Jobs in der lokalen Umgebung also nicht genügend Ressourcen zur Verfügung stehen, so wird er nur dann ausgelagert, wenn diese Maximalzahl V nicht überschritten wird. Abbildung 11.4 gibt ein Beispiel für das Scheduling unter Einsatz dieses Parameters.

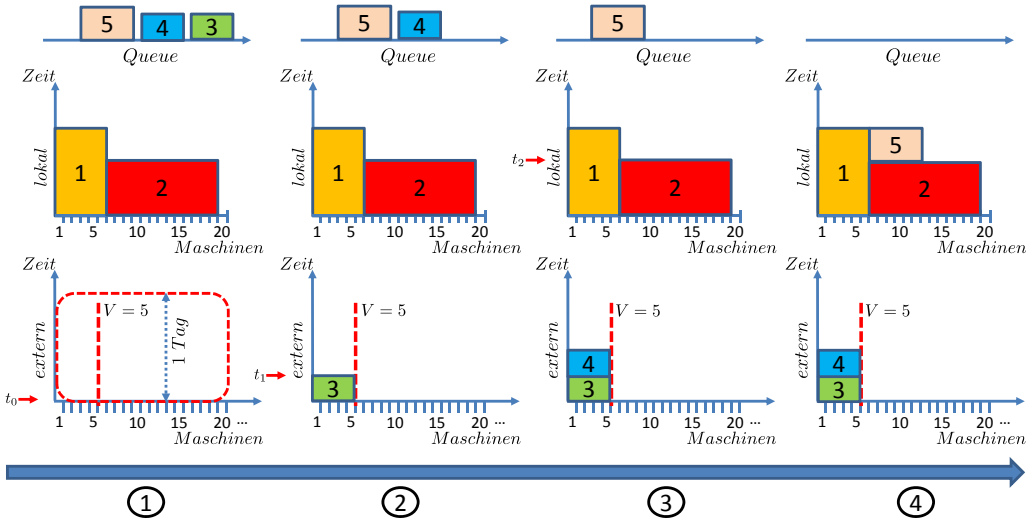


Abbildung 11.4: *Beispiel für das Scheduling mit dem Strategieparameter V.*

Hierbei sind vier verschiedene Situationen ① bis ④ gegeben und der Parameter für den Zeitraum von einem Tag auf $V = 5$ gesetzt. Zusätzlich ist die Rechenkapazität des lokalen Scheduling-Systems dargestellt, die sich aus dem Produkt

von einem Tag Entscheidungsintervall und der Anzahl lokaler Maschinen ergibt (rot-gestrichelt hervorgehoben). Durch die Festsetzung von $V = 5$ entspricht die maximal extern ausführbare Arbeitslast damit 25% der lokalen Rechenkapazität.

In Situation ① wird der Algorithmus zum markierten Zeitpunkt t_0 ausgeführt. Die Jobs 1 und 2 belegen zur Zeit 19 der insgesamt 20 lokal verfügbaren Maschinen. Für keinen der wartenden Jobs 3 bis 5 sind genügend lokale Ressourcen zur Ausführung gegeben und so wird innerhalb der Leihstrategie in Queue-Reihenfolge nach auszulagernden Jobs gesucht. Dies führt lediglich zur Auslagerung von Job 3, da im Anschluss das Maximum an aktiven virtuellen Maschinen in der externen Cloud erreicht ist.

In Situation ② ist Job 3 zum Zeitpunkt t_1 auf den externen Ressourcen abgeschlossen und so kann Job 4 in die Austausch-Cloud gebracht werden.

Die Fertigstellung von Job 2 zum Zeitpunkt t_2 führt in Situation ③ zur lokalen Ausführung von Job 5, der aufgrund seiner Parallelität von 6 keinesfalls in der Cloud hätte ausgeführt werden dürfen.

Der aus den Entscheidungen resultierende Schedule ist dann in Situation ④ dargestellt mit sequentieller Ausführung von Job 3 und 4 auf Fremdressourcen und lokalem Scheduling von Job 5 auf lokalen Ressourcen. Weiterhin sei angemerkt, dass durch die Ausprägung des Parameters $V = 5$ bei 20 lokalen Maschinen maximale Kosten von 25% der für den Tag verfügbaren Ressourcenkapazität hätten entstehen können.

Anstatt eine feste Anzahl von maximal geliehenen Maschinen anzugeben, orientiert sich der zweite untersuchte Strategieparameter genau an diesen relativen Maximalkosten. Abhängig von der lokalen statischen Konfiguration wird der Leihstrategie ein relatives Budget zur Verfügung gestellt, welches für die Auslagerung von Jobs ausgegeben werden darf. Dies führt bei Zuteilung der gleichen Maximalkosten und Jobsequenz wie im vorherigen Beispiel zu einem anderen Schedule.

In Abbildung 11.5 wird mit Situation ① zunächst wieder die Ausgangslage zum Zeitpunkt t_0 dargestellt. Zusätzlich ist neben der Queue sowie dem lokalen und externen Schedule eine weitere Zeile angegeben, welche die Entwicklung des verfügbaren Budgets visualisiert.

Zu Beginn entspricht das verfügbare Budget (B) demnach 25% der lokalen Rechenkapazität (wieder rot-gestrichelt hervorgehoben). Zusätzlich sind für die Jobs 3 bis 5 nun auch die geschätzten Laufzeiten angegeben, da diese maßgeblichen Einfluss auf das Scheduling mit dem Budget-orientierten Strategieparameter besitzen. Die Aktivierung der Leihstrategie führt nun zu einem Vergleich des Budgets mit den maximal zu erwartenden Kosten bei der Auslagerung von Jobs. Diese entsprechen je Job dem Produkt aus erwarteter Laufzeit und geforderten Maschinen. Nur wenn dieses Produkt kleiner oder gleich dem aktuellen Budget ist, so darf der Job ausgelagert werden und das freie Budget wird entsprechend verringert. In unserem Beispiel ist die Summe der Maximalkosten beider Jobs deckungsgleich mit dem aktuellen Budget. Daher werden beide Jobs nun parallel in der Austausch-Cloud gestartet und reduzieren das Budget auf 0.

Zum Zeitpunkt t_1 befindet sich der Scheduling-Prozess in Situation ② und so-

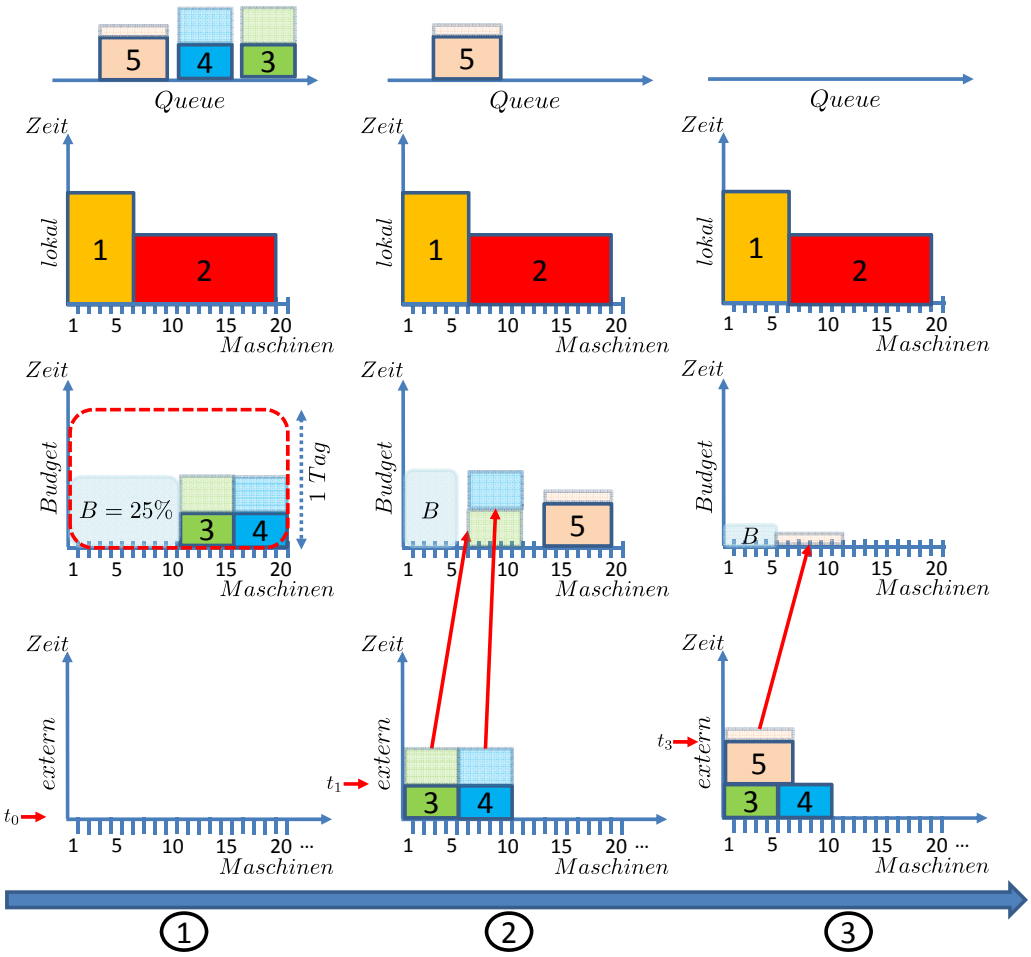


Abbildung 11.5: *Beispiel für das Scheduling mit dem Budget-Strategieparameter.*

wohl Job 3 als auch Job 4 beenden ihre Ausführung. Das zuvor reservierte Budget kann nun wieder freigegeben werden. Da sich Job 5 immer noch in der Queue befindet und auf seine Ausführung wartet, findet nun auch für diesen Job ein Vergleich seiner maximalen Kosten mit dem freigegebenen Budget statt, was in einer Auslagerung von Job 5 resultiert.

Der am Ende erstellte Schedule in Situation ③ weist nun mehrere Unterschiede zu dem Schedule unter einem vergleichbaren V-Strategieparameter auf. So wurden zum einen alle Jobs, die unter ausschließlich lokaler Ausführung hätten warten müssen, ausgelagert, was im Falle von Job 5 unter Verwendung des ersten Strategieparameters gar nicht möglich gewesen wäre. Zum anderen konnte durch die parallele Ausführung der Jobs 3 und 4 zusätzliche Wartezeit eingespart werden. Der Vollständigkeit halber ist auch die Freigabe des reservierten nicht verwendeten Budgets durch Job 5 zum Zeitpunkt t_3 dargestellt.

Als wichtiger konzeptioneller Unterschied zwischen den beiden Ansätzen fällt

auf, dass bei Verwendung des Budget-Parameters das Wissen über erwartete Laufzeiten für die eingegebenen Jobs vorhanden sein muss, da ansonsten nicht von einer Einhaltung des maximalen Budgets ausgegangen werden kann. Dies steht im Gegensatz zur Verwendung des V-Parameters, da die Entscheidung zur Auslagerung von wartenden Jobs in dem Fall lediglich von der Parallelität eines Jobs und der noch freien Parallelität in der erweiternden Cloud abhängt.

Anders als bei den vorgestellten Ansätzen zum Grid Scheduling aus Teil II wurde die restriktive Einstellungen gegenüber der Nutzung geschätzter Laufzeiten auf Austauschebene gelockert und stets EASY Backfilling eingesetzt. Dies führt für alle verwendeten Eingabedaten insgesamt zu einer verbesserten Scheduling-Qualität und reduziert damit auch Hochlastsituationen, die ein größeres Potential zur Einsparung von Wartezeit durch hybrides Cloud Scheduling beinhalten würden.

Würde stattdessen FCFS eingesetzt, so wäre insgesamt mit höheren Maximalkosten und einer höheren Gesamtwartezeit bei den jeweiligen Extremfällen (lokales Scheduling und unbegrenztes paralleles Leihen) zu rechnen. Die nachfolgenden Untersuchungen werden allerdings zeigen, dass der Einsatz von Austauschalgorithmen auch bei Verwendung von EASY Backfilling immer noch ein großes Potential für Verbesserungen bietet.

Da der Bilanzwert eines Strategiepfades immer nur auf Basis der zwei Extremfälle (ohne und mit unbeschränkter Ressourcenerweiterung) möglich ist, müssen vor der Analyse der Strategieparameter zunächst diese parameterunabhängigen Extremwerte ermittelt werden. Tabelle 11.2 gibt eine Übersicht über die erzielten Extremwerte.

Abkürzung	TWT_{ref} $(\cdot 10^6)$	C_{ref} $(\cdot 10^6)$
KTH-11	193	632
KTH-5	104	309
KTH-6	89	324
SDSC00-11	314	907
SDSC00-5	111	360
SDSC00-6	203	537
CTC-11	244	2 122
CTC-5	93	872
CTC-6	151	1 255
SDSC03-11	272	6 386
SDSC03-5	165	2 948
SDSC03-6	107	3 444
SDSC05-11	329	6 803
SDSC05-5	71	2 018
SDSC05-6	253	3 773

TABELLE 11.2: *Maximale Gesamtwartezeit (in Sekunden) und Maximalkosten (in CPU-Sekunden), die durch die Verwendung von EASY-Backfilling für das lokale Scheduling und entweder einer uneingeschränkten Nutzung von Leihressourcen oder ohne Erweiterung der lokalen Ressourcen entstehen.*

Um herauszufinden, welcher der beiden Strategieparameter hinsichtlich der Bilanzfunktion das größte Potential besitzt, wurde ein einfaches Verfahren für die Erstellung von Strategiepfaden (vgl. Abschnitt 11.1.3) eingesetzt. Dieses basiert im

Grunde auf einer Abtastung aller statischen Konfigurationen auf allen Lastaufzeichnungen. Es kann deswegen von einer statischen Konfiguration gesprochen werden, da für einen untersuchten Pfad gilt, dass alle Intervalle mit der gleichen Ausprägung des Strategieparameters simuliert werden ($sp_0 = sp_1 = sp_2 = \cdots = sp_{n-1}$). Im Falle des V-Strategieparameters wurden hierbei — in Abhängigkeit der lokalen Maschinengröße (M) — Konfigurationen von $V = 0$ bis $V = M$ untersucht. Das heißt, die lokalen Ressourcen können bei einer Erweiterung maximal verdoppelt werden.

Analog dazu wurde der Budget-Parameter B in 1%-Schritten von 0 bis 100 Prozent untersucht. Bei der Budget-orientierten Strategie ist es zudem wichtig, dass das innerhalb eines Intervalls ungenutzte Budget verfällt, da eine Aufsummierung von unverbrauchtem Budget auf lange Sicht dazu führen würde, dass die Entscheidungen der Leihstrategie irgendwann keine Relevanz mehr haben, da das freie Budget zu groß wird.

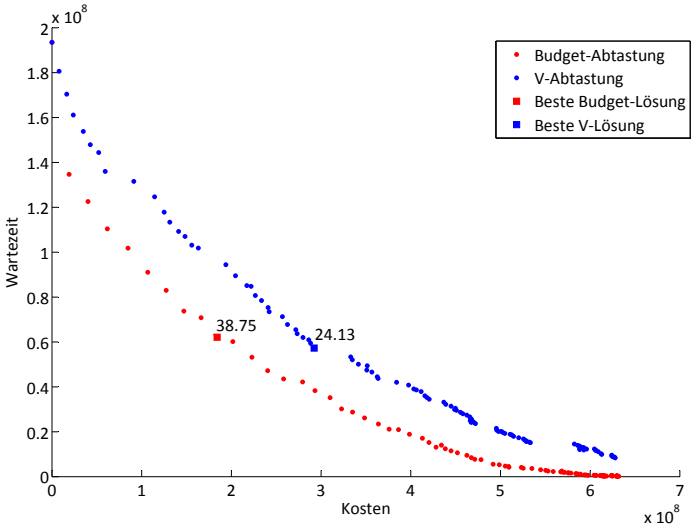


Abbildung 11.6: *Gegenüberstellung der Strategieparameter V und B im Hinblick auf erzielte Kosten und Gesamtwartezeit am Beispiel des KTH-Workloads (*11 *Monate).*

Abbildung 11.6 zeigt am Beispiel des KTH-Workloads welche unterschiedlichen Ergebnisse hinsichtlich entstehender Kosten und Gesamtwartezeit unter Einsatz der beiden unterschiedlichen Strategieparameter erreicht werden.

Auffällig bei dem Ergebnis ist eine Verschiebung der jeweiligen Lösungsfronten in Richtung des utopischen Nullpunktes. Dieser würde einer Bilanz von 100 Prozentpunkten entsprechen, da in dem Fall die Wartezeit um 100% gesenkt würde, wobei gleichzeitig 0% der relativen Kosten entstehen würden.

So weist auch die beste gefundene Lösung mit 9% statischem Budget pro Tag

eine ca. 61% größere Bilanz auf als die beste gefundene Lösung durch den V-Strategieparameter mit $V = 31$ maximalen virtuellen Maschinen.

Ebenfalls ersichtlich bei diesem Ergebnis ist die größere Flexibilität bei Einsatz des Budget-Strategieparameters, die mit einer geringeren Lösungsabdeckung im niedrigeren Kostenbereich einhergeht. Die in Abbildung 11.5 dargestellte höhere Bereitschaft, Jobs auch direkt parallel auszuführen, führt auch bei kleinen Budget-Einstellungen schnell zu Lösungen mit höheren Kosten aber dafür verhältnismäßig größerer Verbesserung in der Wartezeit.

In Abbildung 11.7 zeigt die prozentuale Verbesserung der besten gefundenen statischen Konfigurationen beider Parameter bzgl. des Bilanzwerts für alle fünf Workloads (11 Monate).

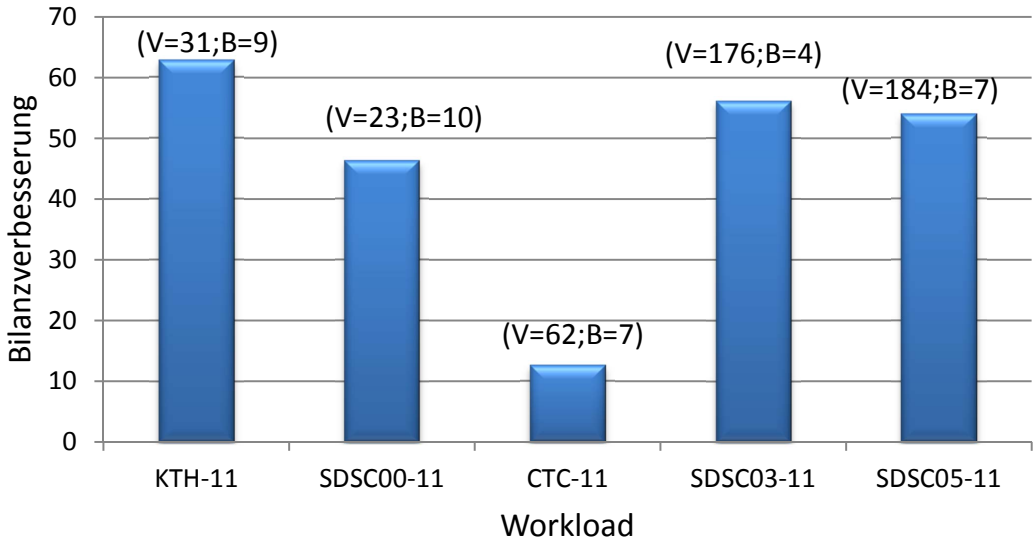


ABBILDUNG 11.7: *Prozentualer Bilanzunterschied der besten gefundenen statischen Budget-Konfiguration relativ zur entsprechenden V-Konfiguration auf allen Aufzeichnungen. Beispiel: $B = 7$ führt bei der Simulation des CTC-Workloads zu einer* 12,6% *höheren Bilanz als $V = 62$.*

Durch die ermittelten Ergebnissen ergeben sich zwei wichtige Erkenntnisse. Zum einen besteht keine direkte Proportionalität zwischen statischen Konfigurationen der Strategieparameter und der lokalen Maschinengröße. Während beispielsweise die SDSC00-Site mit 128 Maschinen größer ist als die KTH-Site mit 100 Maschinen, liegt das beste Kosten-Nutzen-Verhältnis bei einer geringeren Einstellung für den V-Parameter.

Ebenso, wie eine Vervierfachung der Maschinengröße zwischen KTH und CTC scheinbar eine Verdopplung des V-Parameters vorsieht, resultiert eine weitere Vervierfachung der CTC-Site hin zur Größe der SDSC05-Konfiguration in einer Verdreifachung des V-Parameters für den höchsten Bilanzwert. Analog dazu ist auch bei den Resultaten für den Budget-Parameter kein eindeutiges Muster zu erkennen.

Zum anderen zeichnen sich alle Budget-orientierten Simulationen durch höhere Bilanzwerte aus. Hierbei liegen die Verbesserungen zwischen rund 13% und 63%,

was deutlich für den Budget-Strategieparameter spricht. Daher beziehen sich alle weiteren Untersuchungen nur noch auf die Verwendung des Budget-orientierten Strategieparameters.

Durch die statische Konfiguration der Entscheidungen und dem Absuchen aller möglichen Ausprägungen konnten bis hierher bereits gute Strategiepfade ermittelt werden. Streng genommen handelt es sich hierbei nur um eine erste Annäherung an eine mögliche Lösungsfront hinsichtlich der gegensätzlichen Kriterien aus Wartezeit und Kosten. Aus diesem Grund wird im nächsten Abschnitt ein mehrkriterielles Optimierungsverfahren erläutert, welches auch die dynamischen Wechsel des Budget-Parameters zwischen den Entscheidungsintervallen berücksichtigt, da diese auch während des späteren Online-Lernverfahren stattfinden.

11.2 Feststellung des Optimierungspotentials mit mehrkriteriellen Optimierungsverfahren

Eine Optimierung von Strategieentscheidungen hinsichtlich der Bilanzfunktion macht aus Sicht des Betreibers vor allem dann Sinn, wenn er lediglich an dem besten Kosten-Nutzen-Verhältnis interessiert ist und zwar unabhängig von dem bei dieser Lösung erreichtem Grad an Performanzsteigerung und entstehenden Kosten. Im praktischen Einsatz einer Leihstrategie kann es aber durchaus entscheidend sein, ob man eine Bilanz von beispielsweise 30 Prozentpunkten über eine Senkung der Wartezeit um 40% bei 10% der Maximalkosten oder aber über 90% Senkung der Wartezeit mit 60% der Maximalkosten erreicht. So liegen im Falle des elfmonatigen SDSC05-Workloads zwischen 10% und 60% der Maximalkosten (vgl. Tabelle 11.2) in etwa $3\,424 \cdot 10^6$ CPU-Sekunden[10].

Aus diesem Grund und zur Einschätzung der Lösungsqualität von Online-Lernbasierten Leihstrategien ist es daher unerlässlich, zuvor eine Einschätzung über den gesamten mehrkriteriellen Lösungsraum zu erhalten. Hierbei soll der bereits in Abschnitt 5.2 (S. 40) erläuterte Algorithmus NSGA-II zum Einsatz kommen. Er sorgt in unserem Fall dafür, dass eine möglichst diverse Front Pareto-optimaler Lösungen hinsichtlich Kosten- und Wartezeit angenähert wird.

11.2.1 Kodierung des Optimierungsproblems für NSGA-II

Durch die Diskretisierung der Cloud Scheduling Entscheidungen und die Verwendung des Budget-Strategieparameters entsprechen die Individuen des NSGA-II-Algorithmus einzelnen Strategiepfaden bei Evaluation des Austauschverfahrens mit einem bestimmten Workload. So ist die Länge eines Individuums durch die Anzahl der simulierten Tage T begrenzt. Zusätzlich wurde für den Aktionsraum wählbarer Budgets derselbe Aktionsraum genommen wie bei der Gegenüberstellung

[10] Zur Orientierung: Bei einem Stundensatz von 1,3 US-$ pro Stunde für die günstigste Cluster-Instanz (Stand: September 2012) der Amazon EC2 entspricht das mindestens 1,2 Millionen US-$.

unterschiedlicher Strategieparameter (vgl. Abschnitt 11.1.4). Damit ergibt sich abhängig von der Simulationsdauer ein Gesamtlösungsraum von 101^T unterschiedlichen Strategiepfaden. Ein einzelnes Individuum k ist also durch einen Vektor $\vec{o_k} := (o_1, o_2, \cdots, o_T)$ mit $o_i \in [0{,}100]$ bestimmt und liefert bei der Simulation ein Ergebnispaar (C_k, TWT_k) aus Kosten und Wartezeit. Auf diesen Ergebnissen ist NSGA-II — im Sinne einer zweikriteriellen Optimierung — in der Lage, die Selektion von Individuen für die Rekombination durchzuführen (vgl. Abschnitt 5.2).

Für die Erzeugung der Startpopulation wurden die einzelnen Aktionen gleichverteilt zufällig aus dem gesamten Aktionsraum gewählt. Die Populationsgröße war darüber hinaus auf 100 Individuen beschränkt[11]. Neben den zufällig generierten Individuen befanden sich in der Startpopulation auch die Extremlösungen mit $o_i = 0$ und $o_i = 100$ für alle Tage des Pfads, da sich herausgestellt hat, dass der Algorithmus ohne gezielte Einstreuung dieser Lösungen besonders in der Anfangsphase Probleme besitzt, Lösungen in den Randbereichen der Front zu finden.

Für die Rekombination werden jeweils zwei selektierte Eltern kopiert und erzeugen somit zwei Nachkommen. Zusätzlich findet mit Wahrscheinlichkeit $P_{rec} = 0{,}9$ ein Austausch von Pfadabschnitten zwischen den beiden Nachkommen statt. Wie

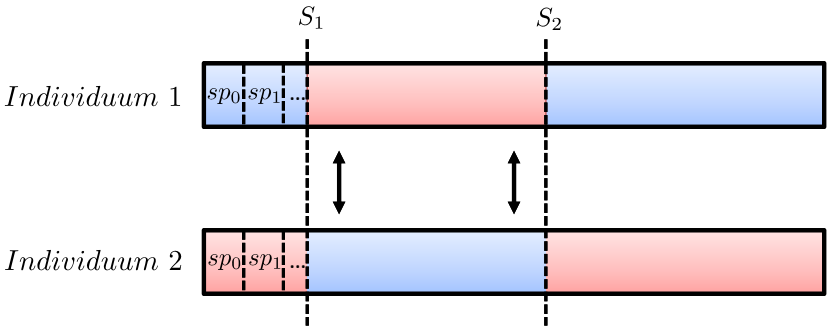


Abbildung 11.8: *Beispiel für ein Zwei-Punkt-Crossover.*

in Abbildung 11.8 dargestellt, findet der Austausch über ein *Zwei-Punkt-Crossover* statt. Dabei werden gleichverteilt zwei verschiedene Schnittpunkte S_1 und S_2 bestimmt. Im Anschluss daran werden die jeweiligen Sequenzen zwischen den beiden Schnittpunkten ausgetauscht[12].

Die Entscheidung für dieses Vorgehen hängt mit der Annahme zusammen, dass zwischen aufeinander folgenden Strategieentscheidungen bestimmte Abhängigkeiten bestehen (z.B. wartende Jobs in der Queue aufgrund restriktiver Budget-Einstellungen in vorangehenden Entscheidungen). Nur durch Austausch von Pfadsequenzen kann diesem Umstand Rechnung getragen werden.

[11] Dies entsprach zur Zeit der Evaluation in etwa der Anzahl permanent verfügbarer Maschinen innerhalb des Evaluations-Clusters.

[12] Werden bei der Wahl der Schnittpunkte z.B. Tag 10 und 15 gewählt, so fällt jedes der Kinder während der Simulation die gleichen Entscheidungen wie eines der Elternteile — allerdings mit getauschten Entscheidungen zwischen Tag 10 und 15.

Die Mutation der einzelnen Nachkommen soll die Entscheidungen auf den Strategiepfaden variieren, also das Budget einzelner Tage auf einem Pfad erhöhen oder senken. Hierbei wird jede Strategieentscheidung (Gen eines Individuums) in 10% der Fälle mit einer konstanten Mutationsschrittweite[13] von 3 und einer Gaußschen Normalverteilung mutiert. Da die Strategieentscheidungen allerdings nur ganzzahlig und innerhalb des Intervalls [0,100] liegen dürfen, erfolgt nach der multiplikativen Mutation mit der Zufallsvariable eine Rundung des Wertes mit anschließendem Setzen auf den entsprechenden Randwert bei Überschreitung einer der Intervallgrenzen.

Durch die Operatoren zur Rekombination und Mutation, werden stets neue Lösungen generiert und somit der evolutionäre Prozess vorangetrieben.

Insgesamt kann bei den Ergebnissen auch nach 500 evaluierten Generationen weder garantiert werden, dass die zu dem Zeitpunkt approximierte Front der tatsächlichen Pareto-Front entspricht noch, dass eine wiederholte Ausführung mit veränderten NSGA-II-Parametern (Wahrscheinlichkeiten für Mutation, Rekombination etc.) nicht zu besseren Ergebnissen geführt hätte. Die Optimierung der NSGA-II-Parameter bzgl. dieses Anwendungsfalls stand nicht im Fokus der Arbeit.

11.2.2 Evaluation

Mit Ausrichtung auf später zu erprobende Übertragbarkeitskonzepte (vgl. Abschnitt 11.3.6) wurde die mehrkriterielle Optimierung mit NSGA-II nicht nur auf den vollen 11 Monaten sondern auch bereits auf Teilsequenzen mit 5 und 6 Monaten Länge sowie für alle fünf Eingabedaten durchgeführt. In diesem Kapitel werden beispielhaft einige der Ergebnisse vorgestellt.

Die Offline-Optimierung mit NSGA-II wurde aufgrund ihres nichtdeterministischen Verhaltens in zehn unabhängigen Läufen mit jeweils 500 Generationen pro Workload durchgeführt[14].

Abbildung 11.9 zeigt beispielhaft die Entwicklung der approximierten Front eines einzelnen Durchlaufs von NSGA-II mit der fünfmonatigen KTH-Aufzeichnung[15].

Dargestellt ist die Startpopulation (1. Generation in rot) und dann jeweils Fronten nach 100 bis 500 Generationen in 100er-Schritten. Aufgetragen sind für jede Lösung jeweils die absoluten Kosten in CPU-Sekunden auf der x-Achse und die absolute Wartezeit in Sekunden auf der y-Achse. Zusätzlich gibt der Rahmen den Lösungsraum zwischen den beiden Extremlösungen an, die sich entweder durch das lokale Scheduling ohne Kosten aber maximaler Wartezeit oder die unbeschränkte Nutzung der Erweiterung ohne Wartezeit aber mit maximalen Kosten ergeben.

[13] Eine Schrittweitenanpassung ähnlich der bei einer Evolutionsstrategie ist bei NSGA-II nicht vorgesehen.

[14] Aus praktischen Gründen wurde nach zehn Durchläufen pro Eingabe abgebrochen, da dies auf dem vorhandenen Rechencluster bereits zwei Monate Simulation benötigt hat.

[15] Die parallelen Durchläufe haben zu ähnlichen Ergebnissen mit leichten Abweichungen geführt, werden aber aufgrund der Übersichtlichkeit nicht in der gleichen Abbildung gezeigt.

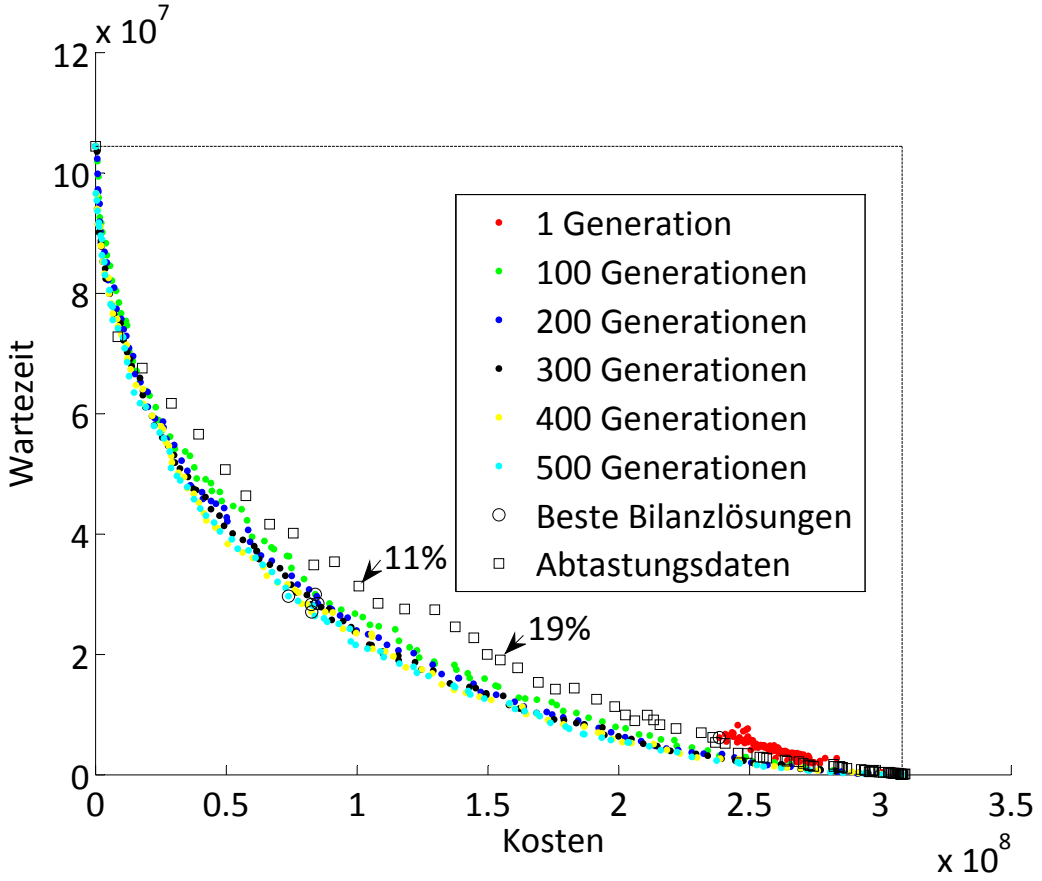


Abbildung 11.9: *Entwicklung der approximierten Front bei der NSGA-II-Optimierung des 5-Monats-KTH-Workloads hinsichtlich Kosten und Wartezeit.*

Ergänzend dazu sind in jeder Front schwarz umkreist die Lösungen hervorgehoben, welche die beste Bilanz erreicht haben, auch wenn dies nicht direkt Ziel der Optimierung gewesen ist. Den Fronten gegenübergestellt sind die Ergebnisse, die wie in Abschnitt 11.1.4 bei einfacher deterministischer Abtastung eines statischen Budgets über den gesamten Simulationszeitraum entstehen.

Auffällig ist hierbei zunächst, dass sich die Lösungen der Startpopulation in einem Bereich mit höheren Kosten sammeln. Das ist dadurch zu erklären, dass die Startindividuen durch gleichverteilte Wahl von Aktionen auf dem gesamten Aktionsraum entstehen und so häufig Aktionen in die Entscheidungssequenzen eingebunden werden, welche zwangsläufig zu hohen Kosten führen. Außerdem deutet dieses Verhalten bereits darauf hin, dass der Einflussgrad größerer Aktionen — also den Aktionen mit einem größeren relativen Tages-Budget — verhältnismäßig früh sinkt. Da es sich nur um eine Erweiterung der lokalen Kapazitäten handelt, macht es für ein Entscheidungsintervall oft keinen Unterschied, ob 20, 21 oder eben 100 Prozent der lokalen Kapazitäten für die Auslagerung zur Verfügung stehen, da

durch all diese Aktionen bereits die maximalen Kosten entstehen. Obwohl sich die zufälligen Individuen eventuell stark in ihren Sequenzen unterscheiden, führen sie zu ähnlichen Ausprägungen in den beiden Kriterien.

Bereits nach wenigen Generationen (100) weist die approximierte Front die gewünschte Divergenz auf. Das ist unter anderem auf das künstliche Einstreuen der Extremlösungen wie die des lokalen Schedulings zurückzuführen. Jene wird von NSGA-II während der frühen Selektionsphasen nämlich häufig zur Rekombination ausgewählt, da sie als Extremlösung nie dominiert werden kann (also immer im niedrigsten Rang einsortiert wird) und aufgrund fehlender Nachbarn zusätzlich auch immer eine unendliche Distanzbewertung besitzt (vgl. Abschnitt 5.2 auf S. 43).

Mit Ausnahme der Startpopulation befinden sich die — hinsichtlich der Bilanz — besten gefundenen Lösungen der Fronten dann ebenfalls in einem verhältnismäßig kleinen Bereich des Lösungsraums. Das hängt damit zusammen, dass die Bilanz durch Gleichgewichtung und prozentuale Normierung der beiden Kriterien die Nähe zum utopischen Nullpunkt bevorzugt.

Die Abtastungslösungen, die im Sinne der Optimierung spezielle Individuen darstellen[16], liefern in der Regel Ergebnisse, die bereits von der Front nach 100 Generationen dominiert werden. An diesen Ergebnissen lässt sich erkennen, dass der Wechsel zwischen Aktionen über die Zeit für die Bilanz von Vorteil ist. Die statischen Ergebnisse jenseits der 19% Budget führen bereits zu Kosten, die größer als die Hälfte der Maximalkosten sind. Dies verdeutlicht einmal mehr den geringen Einflussgrad größerer Aktionen und führt dazu, dass sich ca. 70% der statischen Einstellungen im oberen Kostendrittel befinden.

In Abbildung 11.10 ist zudem beispielhaft die — hinsichtlich der Bilanz — beste optimierte Sequenz dargestellt[17]. Diese entspricht der besonders gekennzeichneten Lösung der 500-Generationen Front. Für jeden der 150 simulierten Tage ist hierbei die Budget-Konfiguration angegeben, die am Ende des Pfades zu der maximalen Bilanz geführt hat. Rund 5/6 aller vorteilhaften Budgets liegen unter dem eingezeichneten Mittelwert von ungefähr 11%, was hinsichtlich der Bilanz eher für eine restriktive Nutzung der Ressourcenerweiterung spricht. Entscheidungen mit einem hohen Budget (beispielsweise an Tag 34 mit 95% Budget) bedeuten darüber hinaus nicht, dass für das gleiche Verhalten nicht auch eine kleinere Budget-Einstellung gereicht hätte. Schließlich ist die unter einer Budget-Einstellung erreichbare Bilanz auch von den einzelnen Jobs über den gegebenen Zeitraum abhängig. Die bei der Offline-Optimierung gezielt stattfindende Anpassung an den unterliegenden Workload führt dazu, dass die beste Lösung, die im Mittel ein Budget von 11% vorsieht, zwar eine vergleichbare Wartezeit erzielt als die entsprechende statische Konfiguration (markiert in Abbildung 11.9) — aber dies bei ca. 25% geringeren Kosten.

Auf Basis der Ergebnisse, könnte man fälschlicherweise davon ausgehen, dass

[16] Hier sind alle Gene eines Individuums identisch.

[17] Auch in diesem Fall handelt es sich nur um eine einzelne Lösung am Ende eines NSGA-II-Durchlaufs.

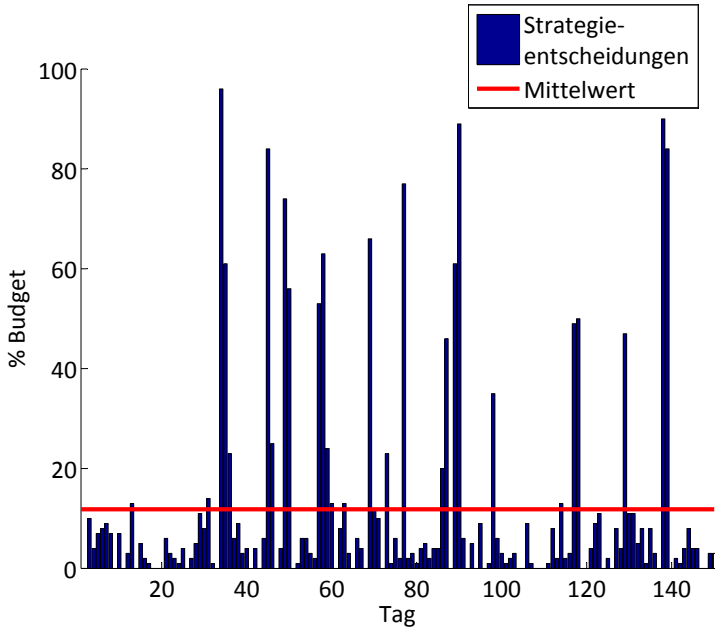


ABBILDUNG 11.10: *Strategiesequenz der hinsichtlich Bilanz besten dominierenden Lösung nach NSGA-II-Optimierung des fünfmonatigen KTH-Workloads.*

eine konstante Konfiguration der Leihstrategie in einem Bereich zwischen 9% und 11% Tagesbudget hinsichtlich des — durch die Bilanz ausgedrückten — Kosten-Nutzen-Verhältnisses stets zu einem vorteilhaften Leihverhalten führt.

Tabelle 11.3 zeigt, dass die besten statischen Konfigurationen, sowohl abgetastet als auch vom Mittelwert der besten NSGA-II optimierten Lösung abgeleitet, stark vom simulierten Workload abhängen. Diese Beobachtung gilt nicht nur für unterschiedliche Sites (KTH, CTC usw.) sondern auch für die unterschiedlichen Workload-Versionen (11 Monate/5 Monate/6 Monate). Die Tabelle enthält einen Mittelwert für die NSGA-II-Ergebnisse, welcher auf Basis aller zehn unabhängigen Durchläufe pro Setup berechnet wurde.

Wenn vor dem Einsatz der Leihstrategie kein explizites Wissen über den zu erwartenden Workload und die damit verbundenen Lastschwankungen vorliegt und es offensichtlich auch keine allgemeingültige statische Konfigurationsempfehlung für die Strategie gibt, so wird ein Verfahren benötigt, welches sich zur Laufzeit dynamisch an die sich ändernden Lastcharakteristika anpassen kann. Ein solches Verfahren wird im folgenden Abschnitt diskutiert.

Abkürzung	Beste statische Konfiguration	Mittelwert der besten Sequenz nach NSGA-II.
KTH-11	9	12
KTH-5	9	12
KTH-6	13	12
SDSC00-11	10	10
SDSC00-5	13	8
SDSC00-6	10	7
CTC-11	7	13
CTC-5	8	14
CTC-6	7	10
SDSC03-11	4	6
SDSC03-5	4	5
SDSC03-6	11	5
SDSC05-11	7	11
SDSC05-5	4	14
SDSC05-6	9	4

TABELLE 11.3: *Veränderungen offline ermittelter Lösungen hinsichtlich der Bilanzfunktion.*

11.3 Dynamisches Online-Lernen von Aktionssequenzen

Im Gegensatz zu dem vorherigen Verfahren geht es nun nicht mehr darum, die Pareto-optimalen Sequenzen hinsichtlich Kosten und Wartezeit zu finden, sondern ein Lernverfahren zu entwickeln, welches bereits innerhalb eines einzelnen Laufs in der Lage ist, ein bzgl. der Bilanz vorteilhaftes Verhalten zu realisieren.

Zu diesem Zweck wird das bereits in Abschnitt 5.3 (S. 44) grundlegend erläuterte Q-Learning genutzt und dessen Anpassung an das Optimierungsproblem im Folgenden erläutert. Die Anpassung teilt sich in zwei verschiedene Aspekte auf. So musste zunächst das Zustands- und Belohnungssignal sowie der Aktionsraum modelliert werden. Zusätzlich wurde eine Analyse möglicher Konfigurationen der Lernparameter des Q-Learnings vorgenommen. Nach erfolgreicher Anpassung des Lernverfahrens an das Problem wurde dessen Effizienz mit den Ergebnissen aus dem Offline-Training und einfachen Heuristiken verglichen.

Der Vergleich wird zeigen, dass eine maximale Performanz nur durch eine starke Anpassung an den Workload (in Form einer Offline-Optimierung) möglich ist, das Q-Learning-Verfahren im Vergleich zu einfachen Heuristiken und ohne vorhandenes Expertenwissen jedoch durchaus zu einem guten Kosten-Nutzen-Gleichgewicht führt. In einer späteren Untersuchung zweier unterschiedlicher Online-Verfahren (Q-Learning und heuristische Vereinfachung) bzgl. des Verhaltens bei unvorhersehbaren Maschinenausfällen wird nachgewiesen, dass sich die Verwendung des

vermeintlich komplexeren Q-Learning-Verfahrens im überwiegenden Teil der Fälle auszahlt.

11.3.1 Grundlegende Modellierung des Problems

Die Wahl des Aktionsraums $\mathbb{A}$ findet analog zu den vorangehenden Untersuchungen der Strategieparameter und der Offline-Optimierung statt. Eine Budget-orientierte Leihstrategie kann daher unter Aktionen in prozentualen Budget-Abstufungen ohne Leihressourcen ($B = 0\%$) bis zu einer Verdopplung der lokal verfügbaren Ressourcen ($B = 100\%$) wählen. Im Gegensatz zu dem ebenfalls diskutierten V-Parameter ist der Budget-orientierte Ansatz in der Anzahl der Aktionen demnach unabhängig von der lokalen Site-Größe.

Die Modellierung eines Zustandssignals s hingegen erweist sich als schwieriger. In einem ersten Ansatz wurde ein verhältnismäßig kleiner Zustandsraum $|\mathbb{S}|$ modelliert, in dem das Zustandssignal ähnlich des Bedingungsteils der Fuzzy-Regeln (vgl. Abschnitt 7.2.4 auf S. 73) über feste Grenzen der normalisierten wartenden Parallelität (NWP) umgesetzt wurde.

Auf diese Art und Weise besaß die Q-Funktion $|\mathbb{S}|^{|\mathbb{A}|}$ viele Q-Werte. Bereits nach wenigen Versuchen stellte sich heraus, dass die Lernrate dieses Verfahrens bei einer begrenzten Eingabelänge (150 bzw. 190 Lernschritte — je nach Workload-Abschnitt) zu gering war.

Das lässt sich dadurch erklären, dass die Wissensbasis repräsentiert durch die Q-Funktion zu Beginn allen Aktionen innerhalb eines Zustandes den gleichen Q-Wert zuordnet. Je öfter ein Zustand dann während des Lernens besucht wird, desto besser werden die Entscheidungen, welche von der Leihstrategie unter diesem Zustand getroffen werden. Das steht aber im Widerspruch zu dem Ziel, durch das Lernverfahren bereits innerhalb weniger Lernschritte zu vorteilhaftem Verhalten zu gelangen. Aus diesem Grund wurde entschieden, den Zustandsraum auf einen einzelnen Zustand zu beschränken und die Vorteilhaftigkeit von Aktionen indirekt in das Belohnungssignal zu kodieren.

Das Belohnungssignal r basiert auf der in Abschnitt 11.1.2 beschriebenen Bilanzfunktion, welche wiederum aus relativen Werten für die durch die Auslagerung von Jobs entstandenen Kosten und relative Verminderung von Wartezeit berechnet wird.

Wie bereits in Abschnitt 11.1.3 beschrieben, lässt sich die Bilanz auf einem Strategiepfad bestehend aus mehreren diskreten Zeitintervallen berechnen, indem man die absoluten Kosten- und Wartezeitergebnisse des gesamten Pfades mit den Ergebnissen zweier Extrempfade (lokales Scheduling und unbegrenzte Auslagerung) in Beziehung setzt. Das gleiche Vorgehen kann auch auf einzelne Entscheidungstage angewandt werden. Dazu wird ausgehend von einem Intervallanfang t das gleiche Scheduling-Verfahren mit unterschiedlichen Budget-Konfigurationen $B \in [0, x, 100]$ aber mit den gleichen Jobeinreichungen zwischen t

und $t + 1$ parallel durchgeführt.

Auf die Art und Weise lässt sich für eine einzelne gewählte Aktion ein Bilanzwert berechnen. Die von Aktionen erreichbaren Bilanzwerte können zwischen den einzelnen Entscheidungsintervallen stark variieren. Daher sollte der Bilanzwert einer Aktion nicht direkt als Belohnungssignal einfließen, sondern zunächst normiert werden. Für diese Normierung ist der Vergleich mit den Bilanzwerten anderer Aktionen nötig. Zu diesem Zweck wird die parallele Simulation eines Intervalls für die beiden Extremlösungen und die gewählte Aktion noch erweitert um die Simulation von Vergleichsaktionen.

Abbildung 11.11 zeigt ein Beispiel für die Ermittlung der Belohnungswerte anhand eines Entscheidungsintervalls $t_i \rightarrow t_{i+1}$. Hierbei werden nach Simulation der zuletzt gewählten Aktion $a_{t_{i-1}}$ alle Datenstrukturen des Scheduling-Systems (Schedule, Queue, Jobs) zum Zeitpunkt t_i dupliziert. Neben den beiden Extremkonfi-

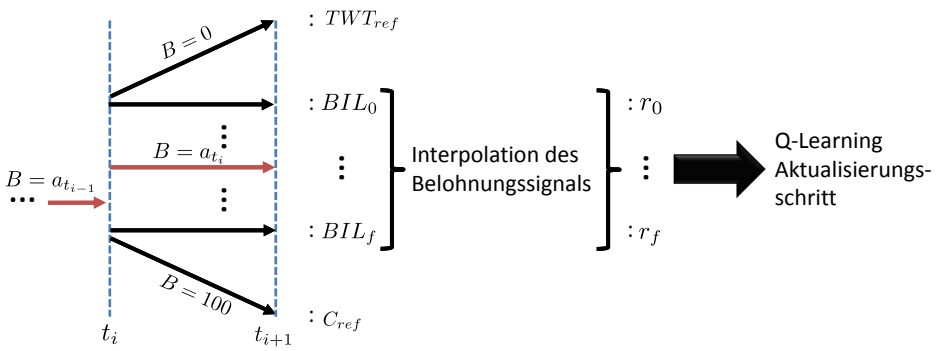


Abbildung 11.11: *Beispiel für Ermittlung der Belohnungssignale parallel simulierter Aktionen für ein Entscheidungsintervall.*

gurationen und der gewählten Aktion a_{t_i} werden f weitere Aktionen berechnet. Jede der simulierten Aktionen bekommt dazu ihr eigenes Duplikat des Scheduling-Systems und kann daher entkoppelt betrachtet werden. Die Wahl der Fensterbreite f hat entscheidenden Einfluss darauf, wie groß der Simulationsaufwand (Laufzeit und Speicher) ausfällt. Gleichzeitig hat die Breite des Fensters ebenfalls Einfluss auf das Lernverhalten, da lediglich Aktionen innerhalb des Fensters ihren Q-Wert und damit die Wissensbasis verändern.

Nach Simulation des Entscheidungsintervalls wird für jede Aktion zunächst der entsprechende Bilanzwert BIL_j ermittelt. Das Belohnungssignal wird dann einfach linear interpoliert zwischen der minimal und maximal erreichbaren Bilanz. Dadurch ist das Verfahren unabhängig von dem sich ändernden Wertebereich der Bilanz und konzentriert sich auf die unterschiedliche Leistung der Aktionen bei gleichen Voraussetzungen (Scheduling-System, eingehende Jobs).

Im Gegensatz zum klassischen Q-Learning kann nun nicht nur die gewählte Aktion belohnt bzw. bestraft werden, sondern alle betrachteten Aktionen. Die parallele Simulation mehrerer Alternativen ist demnach nicht nur ein notwendiges

Mittel, um eine einzelne Aktion zu bewerten, sondern erlaubt es zudem auch, den Lernprozess deutlich zu beschleunigen.

Für die Wahl des Fensters wurde von einer örtlichen Beziehung zwischen den Aktionen ausgegangen, also dass — nach Budget sortiert — benachbarte Aktionen zu ähnlichen Kosten und Wartezeiten führen. Dies bestätigt sich ebenfalls bei den Abtastungsergebnissen aus Abschnitt 11.1.4 auf Seite 142.

Gleichzeitig wurde mit unterschiedlichen Fenstergrößen experimentiert, wobei sich eine Fenstergröße von $f = 10$ als ausreichend groß erwiesen hat[18], um auf Laständerungen zu reagieren. Zu Beginn einer Simulation startet das Fenster in dem restriktiven Budget-Bereich von $B = 0\%$ bis $B = 10\%$. Am Ende jedes Entscheidungsintervalls wird das Fenster um eine Aktion in die Richtung verschoben, in der die Aktion mit dem größten Q-Wert liegt. Befindet sich diese Aktion genau in der Mitte des Fensters, passiert nichts. Aktionen außerhalb des Fensters werden nicht weiter betrachtet. Ihr jeweiliger Q-Wert wird aufgrund fehlender Informationen nicht weiter angepasst und die Aktionen können im Rahmen der Entscheidungsfindung auch nicht mehr gewählt werden.

Kommt eine Aktion allerdings erstmals oder erneut in das betrachtete Aktionsfenster, so muss ihr Q-Wert reinitialisiert werden. Für diese Reinitialisierung wurden drei verschiedene Methoden evaluiert.

In der ersten wurde erneut auf die örtliche Lokalität von Aktionen gesetzt. Dabei wurde eine neu hinzugefügte Aktion mit einem Q-Wert belegt, der minimal kleiner[19] war, als der Wert der benachbarten Fensteraktion. Die neue Aktion sollte demnach nicht gegenüber der Nachbaraktion bevorzugt werden, aber eine ähnliche Güte besitzen.

Der zweite Ansatz hingegen war etwas restriktiver und initialisierte den Q-Wert der neuen Aktion mit dem Mittelwert aller übrigen Aktionen innerhalb des Fensters.

In der dritten (restriktivsten) Alternative nimmt die Aktion mit einem Q-Wert teil, welcher dem Minimum der Q-Werte der übrigen Werte im Fenster entspricht.

Tabelle 11.4 zeigt die drei unterschiedlichen Varianten an einem einfachen Beispiel mit einem Fenster der Breite $f = 10$ ausgehend von dem Fenster $B = 4\%$ bis $B = 14\%$.

Fenster	Aktions-Q-Wert											
	3	4	5	6	7	8	9	10	11	12	13	14
Ausgangslage	0,11	0,73	0,43	0,65	0,69	0,45	0,22	0,56	0,68	0,21	0,35	0,62
(1) Nachbar	0,729	0,73	0,43	0,65	0,69	0,45	0,22	0,56	0,68	0,21	0,35	0,62
(2) Mittelwert	0,46	0,73	0,43	0,65	0,69	0,45	0,22	0,56	0,68	0,21	0,35	0,62
(3) Minimum	0,21	0,73	0,43	0,65	0,69	0,45	0,22	0,56	0,68	0,21	0,35	0,62

TABELLE 11.4: *Beispiel für die Reinitialisierung einer Aktion bei Fensterverschiebung.*

[18] Genau genommen umfasst die Menge betrachteter Aktionen in dem Fall 11 Aktionen, da $f = 10$ zusätzliche Aktionen zu der eigentlich gewählten Aktion simuliert werden.

[19] Mit minimal kleiner ist ein Wert gemeint der in Abhängigkeit des verwendeten Datentyps (z.B. Double) bei der Programmierung minimal kleiner ist.

Bei der gegebenen Ausgangslage ist Aktion 4 diejenige mit dem größten Q-Wert. Daher wird das Fenster eine Aktion in Richtung niedrigerer Budgets verschoben. Demnach wird die Aktion 3 Teil des betrachteten Fensters und muss mit einem Q-Wert belegt werden.

Der alte Q-Wert der Aktion lag bei 0,11 und ist somit für das aktuelle Fenster, dessen Aktionen sich schon über mehrere Entscheidungszyklen hinweg weiterentwickelt haben, nicht mehr passend. Die einzelnen nachfolgenden Zeilen geben beispielhaft an, wie der Q-Wert unter der jeweiligen Reinitialisierungsmethode ausgeprägt wäre.

Eine Bevorzugung neuer Aktionen kann in Einzelfällen zu unerwünschten Effekten führen, da Aktionen im Fensterrandbereich, die zusätzlich eine geringe Güte besitzen, zu einem hohen Q-Wert gelangen können, wenn sie nur kurzzeitig aus dem Fenster entfernt werden.

Analog dazu kann natürlich auch eine Benachteiligung der Aktion stattfinden, wenn eine Aktion vor ihrem Ausstieg einen hohen Q-Wert besessen hat und neben einem schlechten Nachbarn wieder eingesetzt wird. Diese Benachteiligung ist insbesondere in dem Fall gegeben, wenn Alternative (3) eingesetzt wird, da eine zurückkehrende Aktion stets mit dem schlechtesten Q-Wert innerhalb des Fensters reinitialisiert wird und damit auch die schlechtesten Chancen erhält, um gewählt zu werden.

Um festzustellen, welche der drei Alternativen am besten geeignet ist, wurden sie jeweils auf allen elfmonatigen Workloads evaluiert[20]. Hierbei hat sich gezeigt, dass der Mittelwert aller Fensteraktionen (Alternative 2) in allen Fällen zu einer besseren Performanz des Systems führt.

In Kombination von Auslöser zur Fensterverschiebung und Reinitialisierung der Aktionsgüte bewegt sich das Fenster über die Zeit automatisch in den Bereich des Aktionsraums, in dem sich die Aktionen mit den größten Bilanzwerten befinden.

Mit der Modellierung von Zustands- und Aktionsraum sowie der Kodierung des Belohnungssignals ist die grundlegende Anpassung des Lernalgorithmus an das betrachtete Problem abgeschlossen. Nun gilt es, die Lernparameter (α,γ,ϵ), welche das Verhalten des Q-Learning selbst steuern, an das Problem anzupassen, was im folgenden Abschnitt näher behandelt wird.

11.3.2 Konfiguration der Lernparameter

Für die Konfiguration der Lernparameter wurden zunächst umfassende Untersuchungen durchgeführt. Dies gilt insbesondere für die Schrittweite α und die Discount-Rate γ. Für die Wahl der ϵ-Einstellung hingegen waren praktische Gründe ausschlaggebend.

Innerhalb des ϵ-Greedy-Aktionswahlverfahrens (vgl. Abschnitt 5.3 auf S. 44) wurde mit $\epsilon = 0$ ein deterministisches, „ausnutzendes“ Lernverhalten etabliert. Zum

[20] Da über den Einfluss der Lernparameter α und γ zu diesem Zeitpunkt noch keine Erkenntnisse vorlagen, wurden beide mit den in der Literatur [120] empfohlenen Standardeinstellungen (= 0,1) gewählt.

einen würde Nichtdeterminismus zu einer großen Menge unterschiedlicher Lösungen führen, da bei jedem evaluierten Setup unterschiedliche Episoden erzeugt würden. Zum anderen macht ein exploratives Verhalten nur dann Sinn, wenn — wie beim „klassischen“ 1-Step Q-Learning — lediglich die gewählte Aktion evaluiert werden könnte.

In einem solchen Fall könnte Lernfortschritt nur erreicht werden, wenn bisher unbetrachtete Aktionen durch einen Zufallsmechanismus evaluiert würden. In unserem Fall wird stets ein ganzes Set von Aktionen evaluiert und in ihrer erwarteten Belohnung (repräsentiert durch ihren Q-Wert) verändert, so dass eine permanente Ausnutzung des erworbenen Wissens durchaus Sinn macht. Sollte es zu einem Zeitpunkt mehrere Aktionen mit dem höchsten Q-Wert geben, wird die Aktion mit dem kleineren Budget gewählt. Dies entspricht einem eher kostenrestriktivem Verhalten und ist vor allem während der Anfangsphase der Simulation von Bedeutung.

Zwar empfiehlt die Standardliteratur [120] für die Konfiguration von α und γ jeweils 0,1 zu wählen, wenn genaueres Wissen über das Lernverhalten bzgl. des zugrundeliegenden Lernproblems fehlt. Trotzdem wurde im Rahmen der Arbeit untersucht, wie sich unterschiedliche Konfigurationen der beiden Parameter auf die zu erreichenden Bilanzwerte am Ende einer Episode bzw. eines Strategiepfades auswirken.

Am Ende der Untersuchungen hat sich herausgestellt, dass der Parameter *gamma* im vorliegenden Szenario so gut wie keinen Einfluss auf die Leistung des Lernverfahrens besitzt und sich für die Wahl von *alpha* durchaus die Standardkonfiguration mit 0,1 empfiehlt.

Abbildung 11.12 zeigt exemplarisch die Abtastungsergebnisse unterschiedlicher gemessener (α,γ)-Konfigurationen anhand des elfmonatigen KTH-Workloads. Hierbei wurden alle Permutation für die Wertebereiche $0{,}01 \leq \alpha \leq 0{,}99$ und $0{,}00 \leq \gamma \leq 0{,}99$ gemessen[21] und die Ergebnisse anschließend in ein kartesisches Koordinatensystem übertragen mit α auf der x-Achse, γ auf der y-Achse und die durch eine Einstellung erreichte Bilanz am Ende der simulierten Lastaufzeichnung auf der z-Achse.
Hier lässt sich erkennen, dass die Lernentscheidungen hauptsächlich von der Einstellung der Lernrate α abhängig sind. Dieses Ergebnis deckt sich auch mit den Erwartungen, da der Zustandsraum nur einen einzelnen Zustand umfasst und bei der Aktualisierung der Q-Werte aller betrachteter Aktionen das durch γ gewichtete Maximum über alle Aktionen (vgl. Update-Regel 5.2 auf S. 46) identisch ist.

Trotzdem kommt es bei der Aktualisierung der Regeln in Abhängigkeit der γ-Konfiguration durchaus zu unterschiedlichen Q-Werten für dieselbe Aktion. Ist die Belohnung einer Aktion beispielsweise 0 aber mindestens eine Aktion hat einen Q-Wert größer 0, so ist der aktualisierte Q-Wert lediglich abhängig von dem durch γ gewichteten erwarteten Nutzen. So kann die γ-Konfiguration in Einzelfällen dazu

[21] Eine Konfiguration von $\alpha = 0$ macht keinen Sinn, da sich der Q-Wert dann nicht verändert.

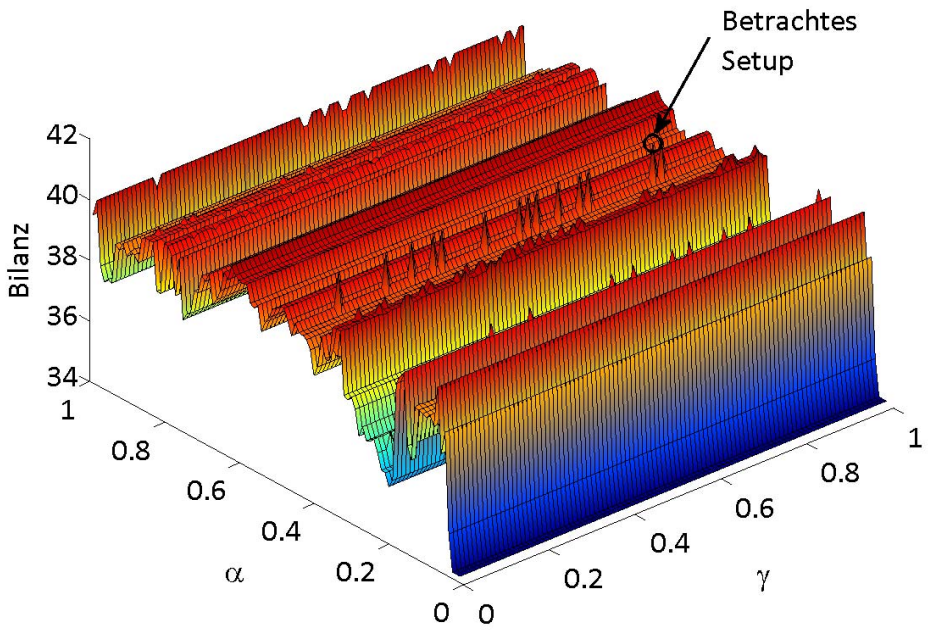


Abbildung 11.12: *Abtastung unterschiedlicher (α,γ)-Konfigurationen mit Q-Learning auf KTH-11-Workload in Auflösung mit* 0,01*-Schritten und Bewertung hinsichtlich der erreichten Bilanzwerte.*

führen, dass sich auch das Q-Wert-Verhältnis zwischen den Aktionen ändert und infolgedessen auch die Aktion mit dem höchsten Q-Wert.

Im Sinne des deterministischen ϵ-Greedy-Aktionswahlverfahrens kommt es dann zu unterschiedlichen Strategieentscheidungen, die auch indirekt Einfluss auf das Fenster betrachteter Aktionen haben können.

Beispielhaft für diesen Effekt wurden die Unterschiede zwischen zwei Setups untersucht, die sich hinsichtlich der γ-Konfiguration leicht unterscheiden. Die typische Konfiguration bei zugrundeliegendem $\alpha = 0{,}45$ ist $\gamma = 0{,}82$. Das abweichende betrachtete Setup (ist ebenfalls in Abbildung 11.12 hervorgehoben) besitzt die Konfiguration $\gamma = 0{,}83$. Abbildung 11.13 zeigt die durch diesen Effekt hervorgehobenen unterschiedlichen Aktionspfade. Abweichende Entscheidungen sind hierbei durch grüne Linien kenntlich gemacht und treten zwischen dem Tag 67 und 160 auf. Sowohl vor als auch nach diesem Zeitraum sind die getroffenen Entscheidungen identisch. Die in dem Zeitraum unterschiedlichen Aktionen führen lediglich zu einer 2,7% höheren Gesamtbilanz am Ende der Simulation.

Bezugnehmend auf Abbildung 11.12 zeigt sich, dass hauptsächlich die Konfiguration von α ausschlaggebend für die Leistung des RL ist. Daher wurde der Raum

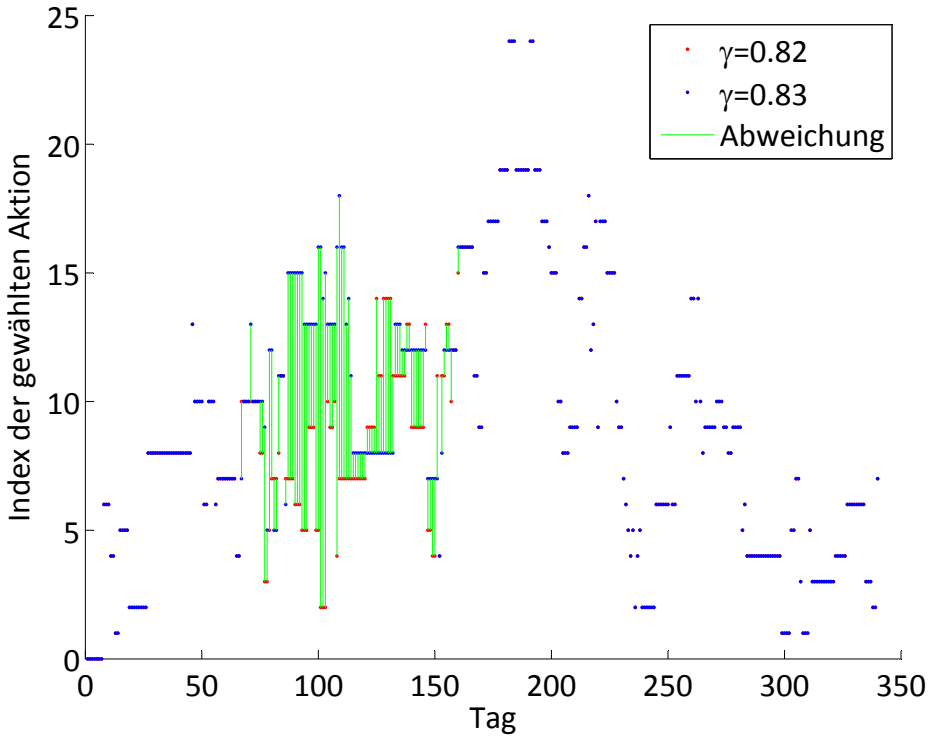


ABBILDUNG 11.13: *Nähere Betrachtung spontaner Abweichung des Verhaltens in Abhängigkeit leicht unterschiedlicher γ-Konfigurationen.*

unterschiedlicher α-Konfigurationen bei einer höheren Auflösung (1000 äquidistante Messpunkte mit $0{,}001 \leq \alpha \leq 0{,}999$) bei konstantem $\gamma = 0$ abgetastet[22].

Diese Abtastung wurde darüber hinaus für alle fünf Lastaufzeichnungen durchgeführt, um zu ermitteln, welche α-Konfiguration sich im Mittel für alle betrachteten Lernsituationen eignet. Da sich die — je nach Lastaufzeichnung — erreichbaren Bilanzwerte in ihrem Wertebereich unterscheiden, wurden sie zusätzlich innerhalb ihres jeweiligen Wertebereichs auf das Intervall [0,1] normiert. Abbildung 11.14 zeigt das arithmetische Mittel der normierten Bilanzwerte aller fünf Lastaufzeichnungen in Abhängigkeit der Einstellungen für α. Gesondert hervorgehoben ist das beste Prozent an Lösungen (entspricht den besten 10 Konfigurationen; rot), sowie die nächstbesten 40 (blau) und darauf folgenden 50 (grün) Lösungen.

Bei dieser Untersuchung ging es weniger darum, genau eine α-Konfiguration zu wählen, welche im Hinblick auf die betrachteten Lastaufzeichnungen am geeignetsten ist. Es ging vielmehr darum, die in der Literatur empfohlene Konfiguration von $\alpha = 0{,}1$ in Abhängigkeit des zugrundeliegenden Problems und der Eingabedaten zu bestätigen oder zu widerlegen. Durch die Messungen in Abbildung 11.14 zeigt sich, dass die Standardkonfiguration von $\alpha = 0{,}1$ durchaus Sinn macht, schließlich liegen

[22] $\gamma = 0$ entspricht einer Eliminierung des durch γ gewichteten Summanden innerhalb der Update-Regel des Q-Learnings.

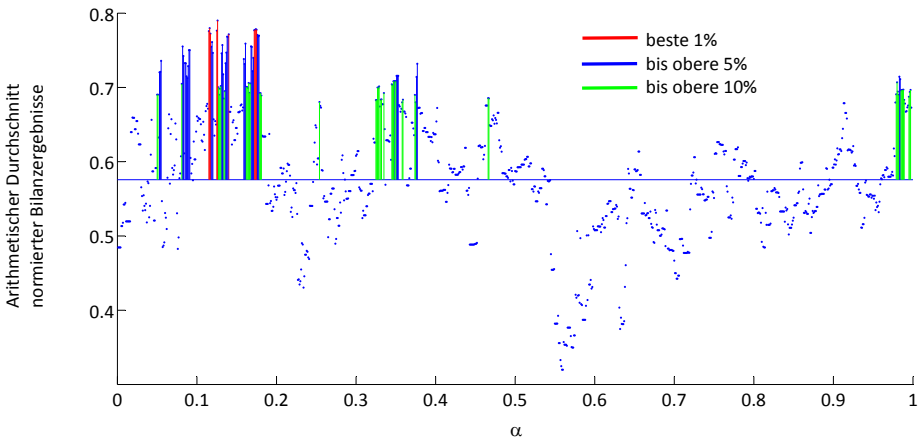


Abbildung 11.14: *Abtastung unterschiedlicher α-Konfigurationen mit Q-Learning auf allen fünf Workloads in Auflösung mit* 0,001*-Schritten und Bewertung hinsichtlich der durchschnittlichen erreichten Bilanzwerte.*

in dem Bereich $0{,}08 \leq \alpha \leq 0{,}18$ die besten Lösungen und — mit einer Ausnahme bei ca. $\alpha = 0{,}095$ — nur überdurchschnittlich gute Lösungen[23].

Daher wurde entschieden, für alle weiteren Untersuchungen zum Q-Learning weiterhin die Standardkonfiguration $\alpha = 0{,}1$ zu verwenden.

11.3.3 Evaluation der Leistung des Q-Learning

Für die Evaluation des Online-Lernverfahrens wurden Simulationen auf allen vollständigen sowie den einzelnen Teil-Workloads durchgeführt. Sowohl bei den Abtastungsexperimenten zur Offline-Bestimmung des besten statischen Budgets als auch bei der Simulation der Q-Learning-gestützten Leihstrategie genügt jeweils ein einzelner Durchlauf[24], da es sich bei beiden um deterministische Algorithmen handelt.

Wie auch zuvor in Abschnitt 11.2.2 beruhen die Ergebnisse für die NSGA-II-Anwendung auf zehn unabhängig durchgeführten Optimierungsläufen mit jeweils 500 Generationen.

Tabelle 11.5 zeigt die — hinsichtlich der Bilanz — erreichten Ergebnisse der drei verschiedenen Verfahren. Dabei wurde der NSGA-II-Bilanzwert arithmetisch gemittelt. Zunächst fällt auf, dass bereits der Unterschied zwischen den beiden Offline-Algorithmen je nach Setup sehr groß ist. Die Abweichung in der Lösungsqualität reicht von 6,26 Bilanzpunkten Unterschied im Falle des elfmonatigen KTH-Workloads bis zu 16,72 Bilanzpunkten Unterschied im Falle der fünfmonatigen Teilsequenz des CTC-Workloads.

[23] Der Durchschnittswert ist mit 0,58 als horizontale Linie eingezeichnet.

[24] Bei verschiedenen abzutastenden Konfigurationen entspricht dies einem deterministischen Durchlauf pro Konfiguration.

Abkürzung	Beste statische Bilanz	Gemittelte beste NSGA-II-Bilanz	QL-Bilanz
KTH-11	38,75	45,01	38,25
KTH-5	40,08	47,67	36,51
KTH-6	39,07	49,83	**41,37**
SDSC00-11	42,85	51,40	40,50
SDSC00-5	39,49	55,75	38,64
SDSC00-6	45,71	53,54	37,44
CTC-11	31,45	41,93	28,34
CTC-5	32,84	49,56	28,44
CTC-6	31,39	43,95	29,19
SDSC03-11	38,02	46,58	37,70
SDSC03-5	40,78	52,82	40,61
SDSC03-6	35,15	46,49	**35,79**
SDSC05-11	32,61	43,56	30,15
SDSC05-5	30,59	46,50	28,96
SDSC05-6	34,87	47,73	28,68

TABELLE 11.5: *Vollständige beste Bilanz-Ergebnisse der drei Ansätze (Statische Konfiguration, NSGA-II, Q-Learning) im Vergleich. Hervorgehobene Werte entsprechen besseren Ergebnissen durch Q-Learning als der besten statischen Konfiguration.*

Dies deutet auf eine hohe benötigte Reaktivität der Leihstrategie auf Änderungen in der Last hin, um ein maximales Kosten-Nutzen-Verhältnis zu erreichen.

Ein so hoher Bilanzwert, wie er teilweise bei einer NSGA-II optimierten Sequenz erreicht wird, kann demnach nur durch eine sehr starke Anpassung an den Workload erzielt werden. Die Anpassung ist ebenfalls in Abbildung 11.15 erkennbar.
Die Abbildung zeigt verschiedene Entscheidungsverläufe, die bei der Evaluation aller drei Ansätze auf der KTH-Site entstehen, im Vergleich. Hierbei sind für jeden Ansatz die Teilsequenzen zusammengefasst und die davon unabhängig evaluierte Gesamtsequenz dargestellt. Da es sich bei dem Q-Learning-Ansatz in unserem Modell um ein deterministisches Verfahren handelt, sind die Entscheidungen der fünfmonatigen Teilsequenz und der Gesamtsequenz im entsprechenden Bereich deckungsgleich.

Dies trifft für die NSGA-II-Ergebnisse nicht zu, da jeder der drei verwendeten Sequenzen (zweimal Teil- und einmal Gesamtsequenz) aus unabhängigen Optimierungen stammen. In allen drei Durchläufen (5,6,11 Monate) sind die Budgeteinstellungen hierbei jeweils starken Schwankungen unterworfen und deuten so auf eine starke Anpassung an den Workload hin, welche im Onlinefall nicht möglich ist.

Gleichzeitig ist erkennbar, dass die QL-Sequenzen sich annähernd um die jeweils beste statische Konfiguration (9% für fünf und elf Monate Simulation gegenüber 13% bei ausschließlicher Simulation der sechsmonatigen Teilsequenz) her-

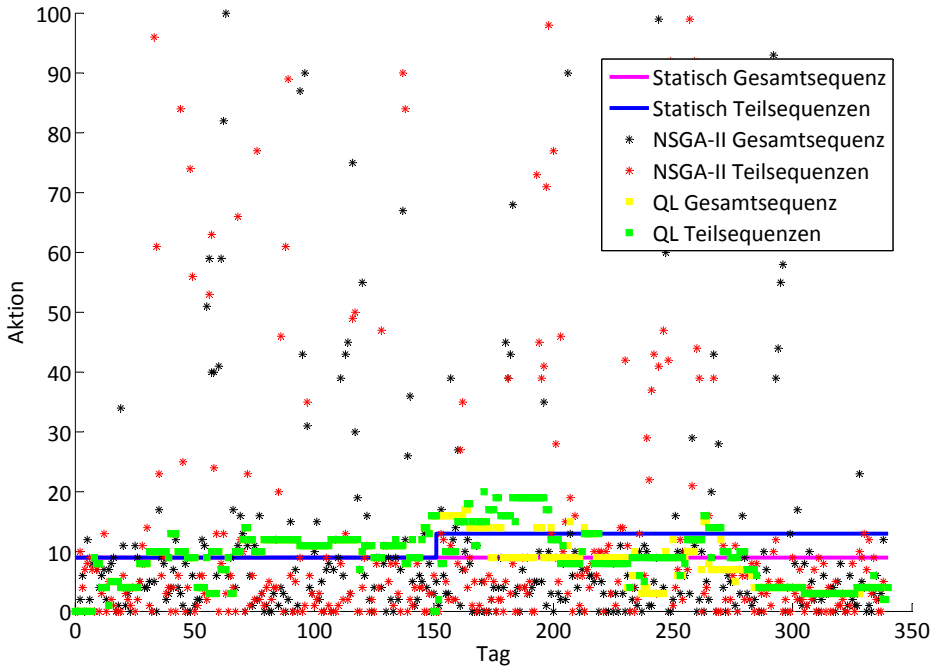


Abbildung 11.15: *Beispiel für Entscheidungsverläufe der drei Ansätze (statische Konfiguration, NSGA-II, Q-Learning) auf Basis des KTH-Workloads.*

umbewegen. Online gelernte Entscheidungsverläufe sind daher eher mit statischen Konfigurationen vergleichbar, da sich der Vorteil einzelner Aktionen (Budgetkonfigurationen) lediglich über kumulierte Erfolge in der längerfristigen Vergangenheit einstellt.

Wie Tabelle 11.5 durch die hervorgehobenen Ergebnisse ebenfalls zeigt, ist das QL-Verfahren in zwei Fällen sogar in der Lage, die offline ermittelte statische Konfiguration zu übertreffen und somit die Nachteile der anfänglichen Einschwingphase beim Lernen sowie das fehlende Wissen über den Workload durch die Vorteile der Flexibilität in der Entscheidungsfindung auszugleichen.

11.3.4 Vereinfachung des Lernverfahrens

Neben den bisherigen Vergleichen der Leistungsfähigkeit des QL-Ansatzes mit Offline-Verfahren zur Optimierung von Budget-Verläufen, wurde versucht, das QL-Verfahren noch weiter zu vereinfachen, um den Einfluss des bestärkenden Lernens besser beurteilen zu können. Schließlich wurden bei der Modellierung des Problems als QL-Verfahren bereits im Vorfeld wichtige Entscheidungen getroffen. Hierzu zählen z.B.:

- Fensterung parallel simulierter Aktionen (Festlegung einer Fenstergröße f, Regeln zur Fensteränderung).

- Normierung der parallel ermittelten Bilanzwerte in Belohnungssignal mit festem Wertebereich zwischen 0 und 1.
- Darstellung der Wissensbasis als Q-Werte.
- Konfiguration von Lernparametern ($\alpha = 0{,}1$ und $\gamma = 0$) und Adaption der Q-Werte mit Hilfe der Lernparameter.

Ausgangsfrage für die folgenden Untersuchungen war, ob sich einige dieser Entscheidungen im Sinne eines einfacheren online adaptiven Ansatzes umsetzen lassen ohne dabei signifikant an Lösungsqualität zu verlieren.

Das Resultat dieser Überlegungen ist ein Ansatz, in dessen Fokus die kumulierte Bilanz der Aktionen steht. Hierbei wird für jede Aktion eine Bilanzsumme mitgeführt, die nach Simulation der betroffenen Aktion immer um den erreichten Bilanzwert erhöht wird. Diese Vorgehensweise ersetzt so zum einen die Speicherung der Wissensbasis in Q-Werten und vereinfacht zum anderen die Adaption der Wissensbasis, da nun keine Lernparameter mehr genutzt werden. Lediglich die parallele Simulation eines (aus Performanzgründen) gefensterten Aktionsraums bleibt dabei erhalten. Die Fenstergröße wird also weiterhin mit $f = 10$ bemessen. Das Fenster beginnt im restriktivsten Budgetbereich und wird in die Richtung verschoben, in der die Aktion mit der größten Bilanzsumme liegt. Für neu zum Fenster hinzugefügte Aktionen wird ebenfalls wieder eine örtliche Lokalität der Aktionen angenommen und gleichzeitig eine Reinitialisierung über den Mittelwert aller Fensteraktionen durchgeführt.

Analog zum QL-Ansatz mit $\epsilon = 0$ und ϵ-Greedy-Aktionswahl wird auch in der Vereinfachung immer die Aktion mit der höchsten Güte gewählt. Tabelle 11.6 zeigt die dabei ermittelten Ergebnisse hinsichtlich der Bilanz für alle (Teil-) Workloads. Zusätzlich wurde auch die Bilanzdifferenz ($\mathrm{DIFF}_{\mathrm{QL}}$) zwischen diesem hier geschilderten vereinfachenden „Summierungsansatz“ (SUM) und dem QL-Ansatz berechnet, um eine bessere Vergleichbarkeit zu ermöglichen. Ein positiver Wert für $\mathrm{DIFF}_{\mathrm{QL}}$ entspricht hierbei einer um den Wert geringeren und damit besseren Bilanz von SUM gegenüber QL.

Die Auswertung zeigt, dass die Vereinfachung durchaus ähnliche Resultate liefert. So schafft der neue Ansatz es ebenfalls in zwei Fällen besser zu sein, als die beste statische Konfiguration. Allerdings schlägt er den QL-Ansatz nur in 7 von 15 Fällen. Beide Ansätze sind daher workload-übergreifend in Bezug auf die Häufigkeit, welcher die bessere Bilanz erzielt, nahezu gleichwertig.

Auch anhand der Abweichungen ist keines der beiden Verfahren signifikant besser als das andere. So erreicht der QL-Ansatz zwar für den fünfmonatigen SDSC03-Workload ein ca. 4,9 Bilanzpunkte besseres Endergebnis als der Summierungsansatz, doch ist im umgekehrten Fall ebenfalls eine auffällige Bilanzdifferenz von ca. 7,3 Bilanzpunkten beim sechsmonatigen SDSC00-Workload gegeben.

Insgesamt sind die Unterschiede zwischen beiden Ansätzen gerade im Hinblick auf die längere Gesamtsequenz marginal, da diese sich für alle Workloads unter 2,1 Bilanzpunkten bewegen. Bei Betrachtung von Entscheidungsverläufen, die bei

Abkürzung	Beste statische Konf.	SUM	DIFF_{QL}
KTH-11	38,75	37,89	−0,35
KTH-5	40,08	36,90	**0,40**
KTH-6	39,07	**39,60**	−1,77
SDSC00-11	42,85	38,44	−2,06
SDSC00-5	39,49	34,27	−4,37
SDSC00-6	45,71	44,73	**7,29**
CTC-11	31,45	29,94	**1,60**
CTC-5	32,84	32,23	**3,79**
CTC-6	31,39	27,45	−1,74
SDSC03-11	38,02	36,12	−1,58
SDSC03-5	40,78	35,69	−4,92
SDSC03-6	35,15	**35,82**	**0,03**
SDSC05-11	32,61	31,48	**1,34**
SDSC05-5	30,59	28,77	−0,19
SDSC05-6	34,87	32,16	**3,49**

TABELLE 11.6: *Vollständige beste Bilanz-Ergebnisse von statischen Konfigurationen und Summierung von Bilanzwerten (SUM) im Vergleich inklusive Bilanzdifferenz ($DIFF_{QL}$) in Bezug auf den QL-Ansatz. Fett gedruckte Werte entsprechen bei SUM erneut besseren Werten als der besten statischen Konfiguration und bei $DIFF_{QL}$ einer besseren Performanz von SUM gegenüber QL.*

beiden Verfahren ermittelt werden, fällt auf, dass der Summierungsansatz stabiler in seinen Entscheidungen ist.
Analog zu Abbildung 11.15 sind in Abbildung 11.16 die Entscheidungsverläufe der SDSC05-Simulationen bei Durchführung des QL- und des Summierungsansatzes dargestellt. Der besseren Übersichtlichkeit halber wurde dieses Mal allerdings auf die gleichzeitige Darstellung der NSGA-II-Sequenzen verzichtet. Die angesprochene Stabilität des Summierungsansatzes zeigt sich für die sechsmonatige Teilsequenz z.B. in dem Zeitraum zwischen dem 170. bis zum 291. Tag. Noch viel stabiler ist der Ansatz, wenn er zuvor auch auf den ersten fünf Monaten gelernt hat. In diesem Fall präferiert er anschließend — mit Ausnahme von $(215 - 216, 294 - 305, 340)$ — immer die Aktion 4.

Die Begründung der Stabilität liegt in der Abwandlung des Belohnungssystems. Schließlich stehen die Aktionen beim Q-Learning unter viel härterem Konkurrenzdruck, da sie sich bei der linearen Interpolation des Belohnungswertes nach Berechnung der Bilanz stets in direktem Vergleich mit den anderen Aktionen des Fensters messen müssen. Hierdurch kommt es auch für eine einzelne Aktion zu sehr schwankenden Belohnungssignalen, die oft in eines der Extrema $r = 0$ oder $r = 1$ umschlagen. Das System reagiert dabei also insgesamt flexibler auf Änderungen oder spontane Ereignisse.

Da beim Summierungsansatz lediglich die reine Bilanz einer Aktion Einfluss auf Änderungen in ihrer Güte nimmt ohne direkten Vergleich mit den anderen Aktionen, kann eine Aktion, die über mehrere Tage hinweg zu schlechten Ergebnissen

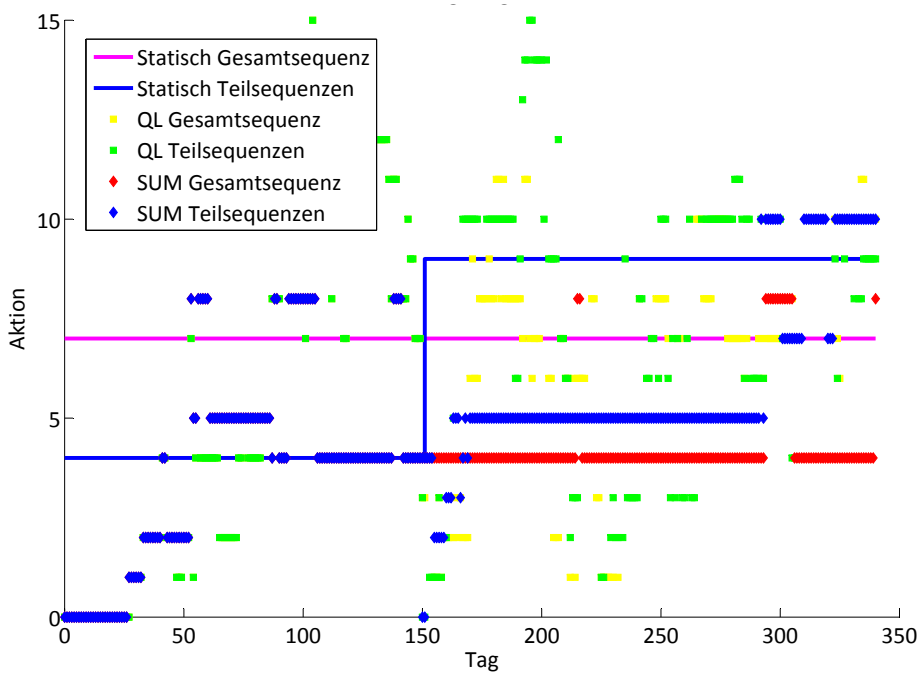


ABBILDUNG 11.16: *Entscheidungsverläufe von Q-Learning und der Bilanzsummierung im Vergleich auf Basis des SDSC05-Workloads.*

führt, trotzdem weiterhin die Aktion mit der höchsten Güte bleiben. Und zwar bleibt sie es genau dann, wenn sie zuvor einen größeren Abstand zwischen sich und die eventuell in dem Fall besseren Aktionen gebracht hat.

Des Weiteren sieht man in der Abbildung auch, dass die Aktion mit der höchsten Bilanzsumme am Ende der Simulation (entspricht der Entscheidung 8% des Summierungsansatzes bei Tag 340.) nicht zwangsläufig der Aktion entsprechen muss, die für die gesamte Sequenz bei statischer Konfiguration am Ende die höchste Bilanz bekommt (statische Aktion 7).

Die Summierungsstrategie ist zwar stabiler und schafft es auch, eine vorteilhafte Bilanz zu erreichen. Die im Gegensatz zum Q-Learning eingebüßte Flexibilität könnte aber wichtig sein, wenn sich am zugrundeliegenden System etwas ändert.

Solche Änderungen könnten z.B. durch eine kurzfristige Überlastung der lokalen Ressourcen durch mehr eingehende Arbeitslast (z.B. durch ungewöhnliches Nutzerverhalten ähnlich der *workload flurries* in Abschnitt 4.1, die nach Empfehlung aus der Literatur im Falle unserer Workloads zuvor entfernt wurden.) oder durch Änderungen im lokalen Ressourcenraum hervorgerufen werden.

Diese Änderungen können zum einen dadurch entstehen, dass die lokale Ressourcenkonfiguration (z.B. durch Zukauf) um weitere Ressourcen erweitert oder durch Ausfall lokaler Ressourcen verkleinert wird. Letzteres soll im folgenden Abschnitt näher untersucht werden.

11.3.5 Untersuchung der Robustheit beider Online-Verfahren im Fehlerfall

Dieser Untersuchung liegt die Annahme zugrunde, dass sich die beiden Ansätze Bilanzsummierung und Q-Learning im Falle von Maschinenausfällen unterschiedlich verhalten. Um dies zu evaluieren, wurde zunächst ein Verfahren entwickelt, um realistische Fehler-Logs zu erzeugen.

Im Rahmen der Untersuchungen hat sich gezeigt, dass der Q-Learning-Ansatz mit zunehmender Fehleranzahl im System weit häufiger zu besseren Scheduling-Ergebnissen führt als die Bilanzsummierung. Dieses Resultat und der Weg dahin wird in den folgenden Ausführungen näher beschrieben.

Dabei wird auf Forschungen von Schroeder & Gibson [107] zurückgegriffen, die sich nicht nur mit der Analyse von Fehlern in HPC-Systemen beschäftigt, sondern mit dem Computer Failure Data Repository (CFDR)[25] zusätzlich auch noch ein Archiv von realen Fehleraufzeichnungen erstellt haben.

Grundsätzlich lassen sich die Fehler innerhalb eines Systems durch drei verschiedene Eckdaten beschreiben. Das sind der Fehlergrund, der Zeitpunkt des Auftretens und die Zeit für die Beseitigung des Fehlers (engl. *Time-To-Repair* (TTR)). In ihren Analysen nutzten die Autoren ca. 20 Fehler-Logs des *Los Alamos National Laboratory (LANL)*, die über eine Gesamtzeit von neun Jahren aufgezeichnet wurden. Im Kern der Analysen stand dabei, die systemweite Zeit zwischen zwei Fehlern (engl. *Time-Between-Failures* (TBF)) und die TTR hinsichtlich ihrer Statistik zu untersuchen. Ein Abgleichen der vorliegenden statistischen Daten mit gängigen Verteilungsfunktionen (Weibull-, Exponential-, LogNormal-Verteilung) sollte es erlauben, künstliche Fehler-Logs für beliebige Site-Größen und Zeiträume zu erzeugen.

Dabei haben sie einige wichtige Erkenntnisse gewonnen, die auch in unserem Fall für die Erzeugung der Fehler-Logs von Bedeutung sind. So hat sich bei Vergleich der unterschiedlichen realen Fehler-Logs gezeigt, dass die Fehlerrate (ausgedrückt durch die TBFs) nahezu linear mit der Systemgröße wächst. Die eigentlichen Instandsetzungszeiten (TTRs) sind jedoch von der Systemgröße unabhängig.

Allerdings besitzen einige reale Aufzeichnungen eine insgesamt höhere Fehlerrate als andere, was die Autoren auf die unterschiedlichen Systemarchitekturen zurückführen, auf denen die Ausführungssysteme beruhten[26].

Für die Simulation von Fehlerfällen im Rahmen der Cloud-Austauschstrategien wurde ebenfalls auf die LANL-Fehler-Logs zurückgegriffen, da diese als einzige

[25] http://cfdr.usenix.org/, Zugriff: 16. Januar 2013.

[26] Bei einer Aufzeichnung waren die Fehlerraten gegenüber den anderen Aufzeichnungen sehr hoch, da es sich bei dem System um eines der ersten NUMA-Systeme (*Non Uniform Memory Access*) handelt, die nach der weiten Verbreitung von SMP (*Shared Memory Processors*) installiert wurden. Die hohen Fehlerraten ergaben sich in diesem Fall durch die noch fehlende Erfahrung von Entwicklern und Systemadministratoren bei Handhabung des neuen Systems.

der Aufzeichnungen aus dem CFDR-Fehlerarchiv alle benötigten Informationen bereitstellen wie den Zeitpunkt eines Fehlers im System, die jeweilige Dauer der Reparatur, sowie die Größe des zugrundeliegenden Systems.

Die Aufzeichnungen wurden zunächst in ein simulationstaugliches Format gebracht, indem beispielsweise die konkreten Datumsangaben in Offsets vom Betriebsstart des Systems übersetzt wurden — ähnlich unserer Simulationszeit, die in Sekunden gemessen bei Zeitpunkt 0 beginnt.

Im Anschluss daran wurden aus den normalisierten Daten (Fehlerereignisse (FE) und Reparaturzeiten (TTR)) die Fehlerzwischenzeiten (TBF) wie in Abbildung 11.17 dargestellt berechnet. Dabei ergibt sich die Zeit zwischen zwei Fehlern a und b als die Zeitspanne nach der Fehlerbehebung von Fehler a und dem Auftreten von Fehler b.

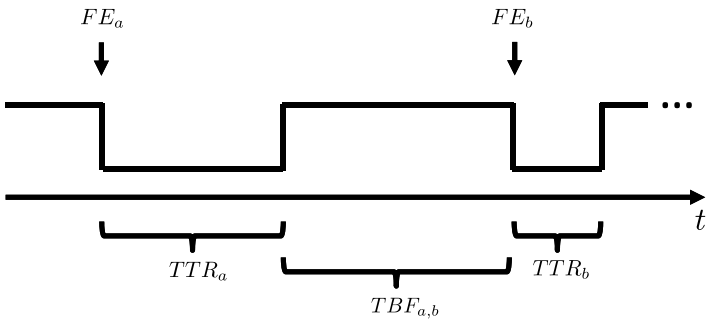


Abbildung 11.17: *Ermittlung der Zeiten zwischen Fehlern aus gegebenen Fehlern eines Fehler-Logs.*

Von den vorhandenen Fehlerdaten wurden als nächstes die 25% ausgewählt, welche die meisten dokumentierten Fehler besaßen, sei es aufgrund einer langen Aufzeichnungsdauer oder aufgrund hoher Systemparallelität. Diese fünf Traces besaßen zwischen 7049 und 2475 Fehlern, wohingegen der nächstgrößere Trace nur 561 dokumentierte Fehler besaß. Da eine so geringe Abtastungs-Rate allerdings statistisch nicht vertretbar gewesen wäre, wurden jene Traces nicht weiter verwendet.

Die aufbereiteten ausgewählten Fehler-Logs wurden anschließend dazu verwendet, systemweite Fehler-Logs von fünf und sechs Monaten für die simulierten Systeme zu generieren. Aufgrund des von Schroeder & Gibson [107] festgestellten linearen Verhältnisses wurden die normalisierten TBFs dabei linear an das Größenverhältnis von Quell- und Zielsystem angepasst.

Angenommen das Quellsystem bestand aus 100 parallelen Maschinen, so konnten die TBFs für die Generierung eines Fehler-Logs des KTH unverändert übernommen werden, während sie für den CTC mit seinen 430 Maschinen zunächst um den Faktor 4,3 verkürzt werden mussten. Bei der Generierung wurden dann gleichverteilt ein entsprechend angepasster TBF auf der Menge der zugrundeliegenden TBFs sowie ebenfalls gleichverteilt eine Reparaturdauer TTR bestimmt.

Zusätzlich wurde noch ein Maschinenoffset berechnet, der sich auf dem Intervall

$[1, M]$ bewegt. Die genaue Bedeutung dieses Offsets wird später erläutert. Dieser Vorgang wurde solange wiederholt, bis die entsprechende Simulationsdauer (fünf oder sechs Monate) erreicht wurde. Ein Fehler-Log für eine Gesamtsequenz ergab sich dann aus dem Zusammenfügen zweier Fehler-Logs für Teilsequenzen.

Auf diese Art und Weise konnten beliebig viele skalierte Fehler-Logs für alle Systeme einer ebenfalls beliebigen Simulationsdauer erzeugt werden, welche die statistischen Eigenschaften der original-Logs nachgebildet haben. Da die beiden Ansätze für das Online-Lernen unter verschiedenen Fehlerhäufigkeiten (ohne Fehler, wenig Fehler, viele Fehler) verglichen werden sollten, wurden die generierten Fehler-Logs hinsichtlich ihrer Fehlerraten noch einmal genauer untersucht. Ein Großteil der generierten Logs wies dabei sehr wenige bis gar keine Fehler auf. Eine Simulation mit diesen Fehler-Logs würde aber zu einem annähernd ähnlichen Ergebnis führen wie eine Simulation ganz ohne Fehler. Daher waren diese Logs ungeeignet. Unabhängig vom Zielsystem haben sich die original-Logs von System 2 der LANL-Aufzeichnungen als am besten geeignet für die Simulation mit wenigen Fehlern erwiesen, da diese für jedes der Zielsysteme linear skaliert und bzgl. einer elfmonatigen Aufzeichnung ein Maschinen-zu-Fehler-Verhältnis von ca. $3 : 1$ erzeugt haben. So wurden für den KTH mit $M = 100$ ca. $30 - 40$ Fehler in einem Log erzeugt[27], für den SDSC05 mit $M = 1664$ ca. $500 - 700$ Fehler[28].

Für die Simulationen mit vielen Fehlern wurde System 16 der LANL-Aufzeichnungsdaten ausgewählt. Hierbei bestand das Verhältnis aus ca. $1 : 2$ Maschinen zu Fehlern bzw. einer Verfünffachung der Fehlerrate gegenüber den generierten Fehler-Logs aus System 2.

Die Begründung für die zuvor erwähnten, zufällig bestimmten Maschinenoffsets liegt darin, dass bei Auftreten eines systemweiten Fehlers während einer Simulation natürlich auch ermittelt werden muss, welche der parallelen Maschinen davon betroffen ist. So wie bei der Aufzeichnung der realen Fehler-Logs auch nur die Fehler notiert wurden, die auch zu Fehlern in der Abarbeitung eines Jobs geführt haben, so sind bei Auslösen eines Fehlers während der Simulation entsprechend auch nur die Maschinen betroffen, die zur Zeit an einer Jobausführung beteiligt sind.

In dem einen Extremfall ist das gar keine in dem anderen sind das alle Maschinen. Der Index der ausfallenden Maschine innerhalb der betrachteten Menge wird nun einfach durch eine Modulo-Rechnung des bei der Erstellung gleichverteilt gewählten Offsets und der Anzahl zur Zeit aktiven Maschinen bestimmt. Auf die Art und Weise ist auch die Wahl auf jeder Teilmenge von Maschinen gleichverteilt und zusätzlich deterministisch bei wiederholter Simulation desselben Fehler-Logs.

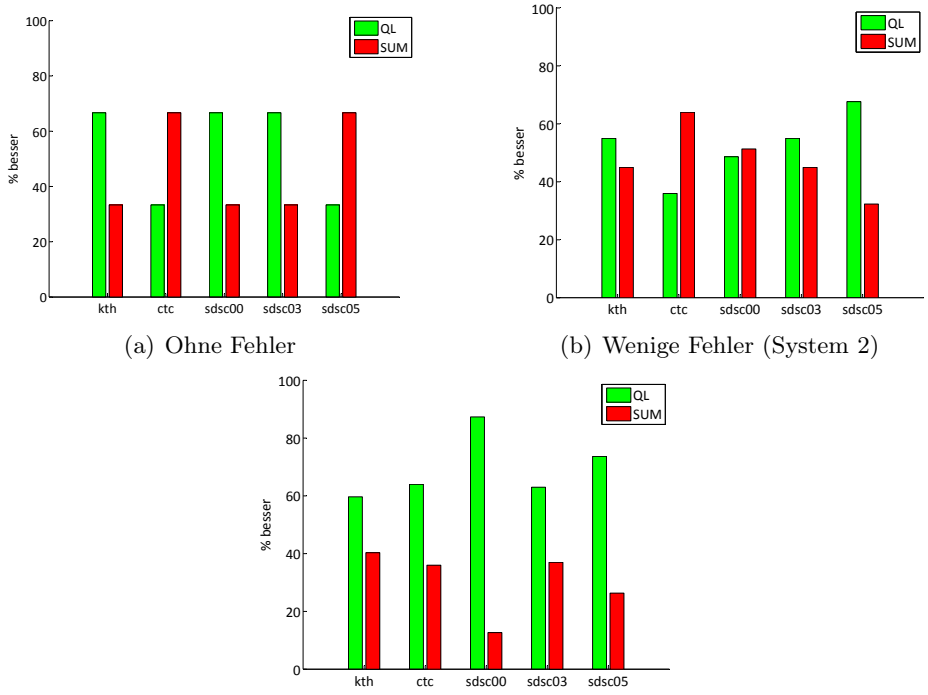
Denn erst letztere Eigenschaft macht es möglich, unterschiedliche Verfahren bei Simulation der gleichen Aufzeichnung miteinander zu vergleichen. Ist eine ausfallende Maschine bestimmt, so wird diese für die Dauer der Instandsetzung (TTR)

[27] Das entspricht einem systemweiten Fehler alle acht bis neun Tage.
[28] Das entspricht einem systemweiten Fehler alle elf Stunden.

für das Scheduling gesperrt und der zur Zeit laufende Job wird abgebrochen und muss neu geplant werden[29].

Da es sich bei dem Generierungsprozess der Fehler-Logs um einen zufallsbasierten Prozess handelt, wurden alle Setups (Paar aus Workload-Sequenz und Quellsystem) 100 mal durchgeführt. D.h. es wurden 100 fünfmonatige und 100 sechsmonatige Fehler-Logs für jeden Workload erstellt und anschließend zu 100 elfmonatigen Logs zusammengeschlossen.

Abbildung 11.18 zeigt die Ergebnisse der Untersuchungen für unterschiedliche Fehlerraten. Dargestellt sind die prozentualen Anteile von Versuchen, die zu einer besseren Bilanz für das Q-Learning bzw. den Summierungsansatz geführt haben. Demnach visualisiert Abbildung 11.18(a) die in Tabelle 11.6 (S.166) aufgeschlüsselten Ergebnisse ohne Fehler. Da für jeden Ursprungs-Workload lediglich einzelne Simulationen für die beiden Teilsequenzen und die Gesamtsequenz durchgeführt wurden, liegen die erreichten Prozentwerte besserer Simulationen also bei Vielfachen von 33%. Anders verhält es sich hingegen bei Abbildung 11.18(b) und Abbildung 11.18(c), da die Vergleichssimulationen hierbei auf den besagten 100 generierten Fehler-Logs beruhen. Im Vergleich wird deutlich, dass sich das Ver-



(a) Ohne Fehler

(b) Wenige Fehler (System 2)

(c) Viele Fehler (System 16)

ABBILDUNG 11.18: *Mehrheitsverhältnisse besserer Versuche für Simulationen mit unterschiedlichen Fehlerraten.*

[29]Eine Wiederaufnahme/Fortsetzung eines Jobs auf einer anderen Maschine ist nicht vorgesehen, da das unterliegende Modell kein Checkpointing (vgl. Kapitel 3) unterstützt.

hältnis von besseren Versuchen positiv in Richtung des Q-Learning entwickelt. Je mehr Fehler im System auftreten, desto vorteilhafter wird die Flexibilität des Q-Learning. Trotzdem gibt es auch bei vielen Fehlern immer noch Situationen, in denen der Summierungsansatz eine bessere Performanz liefert, da die getroffenen Cloud Scheduling-Entscheidungen eines Ansatzes Einfluss darauf haben, welche Jobs lokal laufen und welche nicht. Indirekt wird darüber aber auch die Menge von Jobs bestimmt, die von einem lokalen Maschinenausfall betroffen ist. Hierbei kann es bei beiden Ansätzen zu Situationen kommen, in denen ein aufwändiger Job[30] abgebrochen wird und durch seinen Neustart erheblichen Einfluss auf die Wartezeit hat. Die Untersuchung vieler generierter Fehler-Logs pro Setup zeigt dennoch, welches der beiden Verfahren für eben solche Ausnahmesituationen anfälliger ist.

Der Grund für die vergleichsweise schlechte Performanz des Q-Learning gegenüber dem Aufsummieren bei wenigen Fehlern und den Workloads CTC und SDSC00 liegt begründet in dem Performanzabstand der beiden Ansätze im Falle einer Simulation ganz ohne Fehler. Eine Simulation mit vergleichsweise wenig Fehlern verhält sich auch bei sehr vielen zufallsgenerierten Fehler-Traces sehr ähnlich wie die Simulation ohne Fehler, da nur wenige Jobs abgebrochen werden müssen.

Bezüglich der beiden genannten Workloads war der Summierungsansatz in einer der Teilsequenzen sehr überlegen (vgl. Tabelle 11.6). Im Falle des CTC war dies die fünfmonatige Teilsequenz, die sich in Folge auch auf die Performanz der Gesamtsequenz auswirkt. Im Falle des SDSC00 ist dies vor allem die sechsmonatige Optimierung ohne vorangehende fünfmonatige Teilsequenz. Obwohl es auch in den Sequenzen durchaus einzelne Fehlerversuche gibt, die zu einer besseren Bilanz für das Q-Learning führen, so kann der große Performanzunterschied der entsprechenden Teilsequenzen erst bei Auftreten vieler Fehler überwunden werden. Insgesamt am empfindlichsten für Maschinenausfälle ist die SDSC05-Maschine. Hier zeigt sich bereits bei wenigen Fehlern eine sehr starke Dominanz des Q-Learnings.

Alle Versuche pro Fehlerfall zusammengefasst ergeben die in Tabelle 11.7 aufgeführten Verteilungen von beiden Ansätzen im Vergleich. Darin ist ebenfalls zu

Fehlerexperiment	QL besser in %	SUM besser in %
Ohne Fehler	53,33	46,66
Wenig Fehler	52,46	47,53
Viele Fehler	69,53	34,46

TABELLE 11.7: *Zusammenfassung der Ergebnisse aller Vergleiche zwischen Bilanzsummierung und Q-Learning nach Klasse von Fehlersimulationen.*

erkennen, dass die Wahrscheinlichkeit, mit welcher der QL-Ansatz eine bessere Performanz besitzt als der Summierungsansatz, mit zunehmender Fehlerrate steigt. Wo das Verhältnis zwischen beiden im Fall ohne Fehler noch ausgeglichen ist und bei

[30] Damit sind eine hohe Parallelität und/oder lange Ausführungsdauer gemeint.

wenigen Fehlern noch den üblichen Schwankungen entspricht, verschiebt es sich bei einer hohen Fehlerrate zu mehr als $\frac{2}{3}$ zugunsten des Q-Learnings.

Die Untersuchungen haben gezeigt, dass die vereinfachte Version des Online-Lernverfahrens in einer idealisierten Umgebung ohne Fehlerfälle vergleichbar gut funktioniert wie der aufwändige Q-Learning-Ansatz. Das relativ stabile Verhalten der Strategie zur Steuerung der Budget-Konfiguration macht die Entscheidungen zwar nachvollziehbarer, schränkt die Leistung aber unter Umständen ein, wenn viele Fehler im System auftreten und eine schnelle Reaktion der Strategie und Anpassung des Budgets benötigt wird.

11.3.6 Übertragbarkeitsanalyse der unterschiedlichen Konzepte

In den vorangehenden Untersuchungen lag der Fokus auf einer Optimierung der Leihstrategie im Hinblick auf ein ausgewogenes Kosten-Nutzen-Verhältnis — ausgedrückt durch die in Abschnitt 11.1.2 beschriebene Bilanzfunktion. Die Bilanz selbst aggregiert aber per Definition lediglich das Verhältnis der zwei Kriterien Kosten und Wartezeitersparnis, geht aber nicht direkt auf die konkreten Ausprägungen der Größen ein. Dies zeigt sich beispielsweise in Abbildung 11.19.

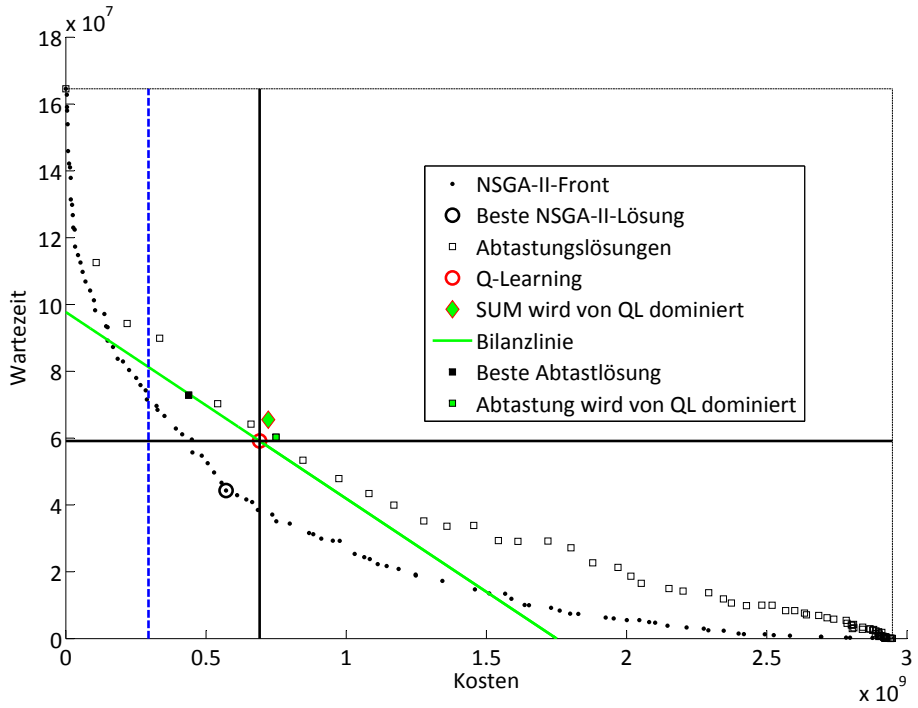


ABBILDUNG 11.19: *Darstellung verschiedener Optimierungslösungen im Vergleich anhand der SDSC03-Teilsequenz über 5 Monate.*

Diese stellt die unterschiedlichen Lösungen in absoluten Kosten (in CPU-Sekunden) und absoluter Wartezeit (in Sekunden) nach dem Lernen auf der fünfmonatigen

Teilsequenz des SDSC03-Workloads dar. Dargestellt sind insbesondere die QL-Lösung, die Bilanzsummierungslösung und die beste Abtastungslösung relativ zu einer durch NSGA-II ermittelten Front.

Zusätzlich wurde auch eine Bilanzlinie durch die QL-Lösung gezogen. Alle Lösungen, die auf dieser Linie liegen, haben den gleichen Bilanzwert. Das gilt in genau diesem Setup beispielsweise für die beste Abtastlösung (Aktion 4 mit 4% dauerhaftem Budget). Obwohl beide in völlig unterschiedlichen Kostenbereichen liegen (23,45% der Maximalkosten bei QL und 14,85% bei statischer Konfiguration) besitzen sie trotzdem eine ähnliche Bilanz von ca. 41 Bilanzpunkten.

Unter Umständen möchte der Betreiber der Site nicht nur eine Optimierung hinsichtlich der Bilanz durchführen, sondern gleichzeitig auch einen bestimmten Lösungsbereich (z.B. prozentuale Kosten) angeben, welcher bei Anwendung einer Leihstrategie erreicht werden soll. Hierbei gehen wir davon aus, dass die Entscheidung des Betreibers auf seinen Erfahrungen der Simulation eines fünfmonatigen Teil-Workloads seiner Site beruht, denn nur auf Basis von Erfahrungswerten kann er abschätzen, welche Lösungsbereiche überhaupt erreichbar sind.

Abhängig von den Ergebnissen entscheidet er sich nun für eine Konfiguration des unterliegenden Lernsystems, welche in der Vergangenheit zu einer Lösung mit geringer Entfernung zu seinem Zielbereich geführt hat. In Abbildung 11.19 ist beispielsweise ein Zielbereich von 10% der maximalen Kosten eingezeichnet. Die dazu am besten passende statische Konfiguration wäre ein Budget von 3%. Im Optimalfall würde diese statische Konfiguration auch in der Übertragung auf die sechsmonatige Teilsequenz zu einer Lösung mit 10% der dort ermittelten maximalen Kosten führen.

In diesem Zusammenhang wurden alle drei Ansätze (statische Konfiguration, Q-Learning und Bilanzsummierung) hinsichtlich ihrer Übertragbarkeit untersucht — also wie gut eine bestimmte vom Betreiber ausgewählte Konfiguration in ihrer Lösung auf dem sechsmonatigen Workload von der Lösung auf dem fünfmonatigen Workload hinsichtlich relativer Kosten und Wartezeit abweicht.

Eine Übertragung NSGA-II optimierter Entscheidungssequenzen ist nicht möglich, da diese immer auf eine bestimmte Länge beschränkt sind (150 Tage bei fünfmonatigem Training) und daher nicht auf Workloads anderer Länge übertragbar sind. Darüber hinaus sind sie sehr stark angepasst auf — in der Trainingssequenz enthaltene — Lastschwankungen (vgl. Abschnitt 11.2.2), so dass eine Übertragung auf die Anwendungssequenz nicht erfolgversprechend ist.

Die nachfolgende Gegenüberstellung der drei Ansätze wird zeigen, dass in der Übertragung von einer fünfmonatigen Abtastung auf einen sechsmonatigen Anwendungszeitraum die Anwendung einer statischen Konfiguration dem Einsatz von Online-Lernverfahren vorzuziehen ist. Diese Empfehlung wird sowohl auf Basis einer Abweichungsmetrik als auch anhand visualisierter Resultate gegeben.

Bei der Abtastung verschiedener statischer Konfigurationen werden unterschiedliche Lösungen erzeugt. Dies ist bei den Online-Lern-Ansätzen allerdings

grundsätzlich nicht der Fall. Zwar besitzt das eigentliche Lernverfahren zwar Parameter wie die Fensterbreite bei paralleler Simulation f oder — im Falle des Q-Learning — Lernparameter wie α und γ, trotzdem resultieren wiederholte Simulationen immer in der gleichen Lösung für Kosten und Wartezeit.

Um auch bei diesen Ansätzen Lösungen in unterschiedlichen Kostenbereichen zu erzeugen, wurde ein Verfahren entwickelt, welches die Zielfunktion (Bilanz) parametrisierbar macht. Hierbei wurde auf die in Abschnitt 5.2 (S. 41) beschriebene A priori-Gewichtungsmethode beider Optimierungskriterien zurückgegriffen.

Ausgehend von der einfachen Bilanzfunktion (vgl. Abschnitt 11.1.2) wurde dazu eine erweiterte Bilanzfunktion in Gleichung (11.2) formuliert, welche über einen einzelnen reellwertigen Parameter w $\in$ [0,1] eine Gewichtung der beiden Faktoren relative Wartezeitverbesserung (TWT_I) und relative Kosten ($\%C$) vornimmt.

$$\mathrm{BIL(w)} = (1 - \mathrm{w}) \cdot \mathrm{TWT}_I - \mathrm{w} \cdot \%C \tag{11.2}$$

Ist das Kostengewicht w auf 0 gesetzt, so entspricht die erweiterte Bilanzfunktion demnach direkt TWT_I und der Fokus liegt auf der Minimierung der Wartezeit ohne Beachtung der dadurch entstehenden Kosten. Im umgekehrten Fall w = 1 spielt die Wartezeit keine Rolle sondern lediglich die Minimierung der prozentualen Kosten.

Wird die Standardbilanzfunktion durch die gewichtete Form BIL(w) ersetzt, so lassen sich sowohl für das Q-Learning als auch für den Summierungsansatz ebenfalls mehrere Lösungen erzeugen, die sich analog zur statischen Abtastung innerhalb des Lösungsraumes verteilen.

Für die in Abbildung 11.20 beispielhaft visualisierte Simulation der unterschiedlichen Ansätze wurden die Kostengewichte w im Fall der Online-Lernverfahren gleichmäßig über den Wertebereich [0,1] verteilt. Ähnlich zur Abtastung statischer Budget-Konfiguration, die von 0% bis 100% vorgenommen wird, wurden ebenfalls jeweils 100 Einstellungen für das Kostengewicht evaluiert.

Hierbei wird deutlich, dass alle Ansätze zu einer Art Inselbildung von Lösungen neigen, jedoch mit unterschiedlicher Charakteristik. Die Abtastung statischer Konfigurationen beispielsweise liefert im vorderen Kostenbereich (bis ca. $0{,}5 \cdot 10^8$ CPU-Sekunden — entspricht etwa $\frac{1}{6}$ des Wertebereichs) lediglich 6 von 101 Lösungen. Mit höheren Kosten wird die Raumabdeckung stetig besser, da es — wie zuvor bereits erwähnt — bei höheren Budget-Einstellungen kaum noch zu unterschiedlichen Entscheidungen bei der Simulation des Eingabe-Workloads kommt. Diese Entwicklung resultiert dann auch in einer Ansammlung von 57 nahezu gleichwertigen Lösungen (im oberen $\frac{1}{6}$ mit über $2{,}7 \cdot 10^8$ CPU-Sekunden). Dennoch ist die Abdeckung des Lösungsraums zwischen diesen beiden Bereichen relativ gleichmäßig, wenn man sie mit den Ergebnissen der beiden anderen Ansätze vergleicht.

Auch diese beiden Verfahren bilden Lösungsinseln, die allerdings nicht wie zuvor nur an einem der Ränder des Lösungsbereichs liegen, sondern vielmehr auch mitten im Lösungsraum.

So liegen im Falle des Q-Learnings beispielsweise 15 Lösungen in dem Kostenintervall ($1{,}5 \cdot 10^8$ bis $1{,}7 \cdot 10^8$) und ebenfalls 15 Lösungen in dem Intervall ($2{,}45 \cdot 10^8$

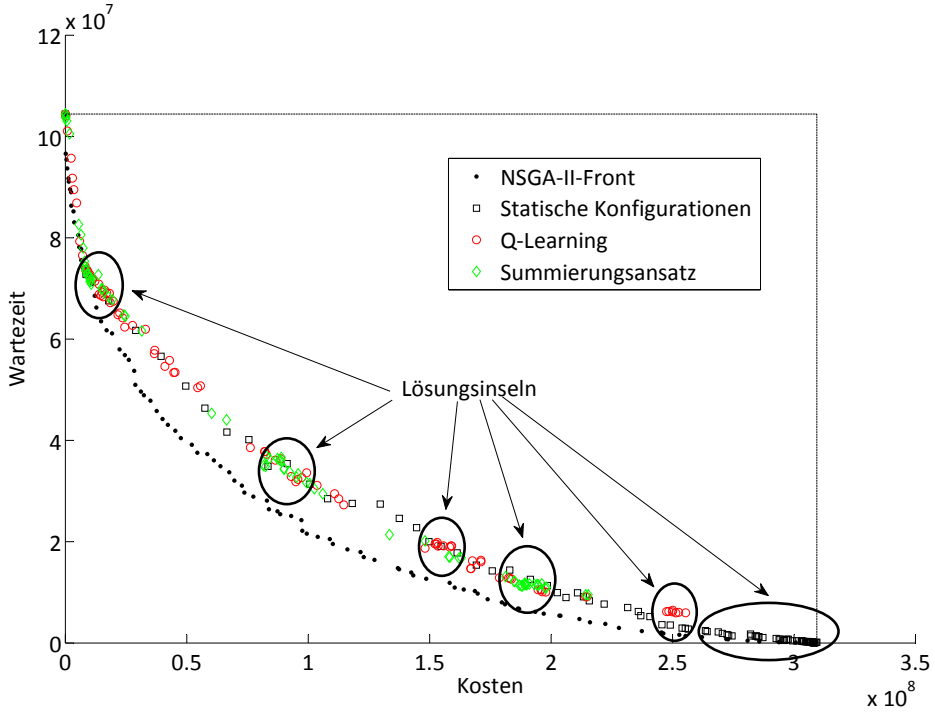


ABBILDUNG 11.20: *Lösungsverteilung statischer Abtastung und der parametrisierbaren Versionen von Q-Learning und Bilanzsummierung bei Verwendung der erweiterten Bilanzfunktion und Anwendung der fünfmonatigen KTH-Lastaufzeichnung.*

bis $2{,}55 \cdot 10^8$) sowie bei der Aufsummierung 15 Lösungen in dem Intervall ($0{,}8 \cdot 10^8$ bis $1{,}0 \cdot 10^8$) und 21 Lösungen in dem Intervall ($1{,}8 \cdot 10^8$ bis $2{,}0 \cdot 10^8$).

Auch diese Inseln sind ähnlichen Entscheidungen bei Simulation des entsprechenden Workloads mit gleichem Ansatz geschuldet. Im Sinne der Übertragung einer bestimmten Konfiguration von einer fünfmonatigen Trainingssequenz auf eine sechsmonatige Anwendungssequenz führt diese Inselbildung allerdings zu dem Problem, dass ein Administrator in der Auswahl möglicher Zielbereiche sehr eingeschränkt ist, da die Lösungsräume zwischen den Inseln relativ spärlich besetzt sind.

Liegt ein Zielbereich in der Nähe einer solchen Lösungsinsel, so wird der Site-Betreiber darüber hinaus mit dem Problem konfrontiert, dass er eine Wahl zwischen vielen — nahezu gleichwertigen — Konfigurationen treffen muss.

Die folgenden Abbildungen (11.21 bis 11.23) visualisieren die Übertragung aller Konfigurationen von den fünf- auf die sechsmonatigen Teilsequenzen des KTH-Workloads für alle drei Ansätze im Vergleich. Die Ergebnisse der Simulationen wurden jeweils mit Hilfe ihrer Maximalwerte C_{ref} und TWT_{ref} prozentual normiert, um eine Vergleichbarkeit zu ermöglichen. Schließlich ist es Ziel der Untersuchungen, festzustellen, wie stabil die Konfigurationen der unterschiedlichen Ansätze bzgl. der prozentualen Ausprägungen hinsichtlich Kosten und Wartezeit sind. Die

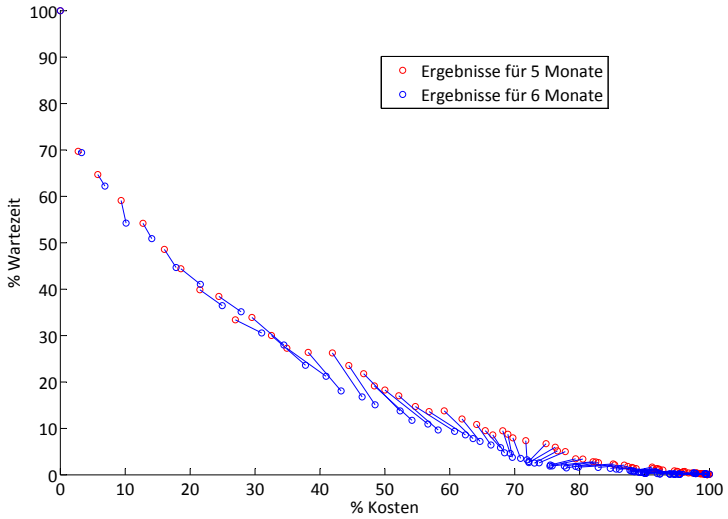


ABBILDUNG 11.21: *Normierte Darstellung der Übertragungsexperimente bei Anwendung des KTH-Workloads für die statische Budget-Abtastung.*

Verbindungen zwischen den jeweiligen Messpunkten soll zeigen, welche Paare von Lösungen zusammengehören.

Im Fall der statischen Budget-Abtastung in Abbildung 11.21 fällt eine gewisse Ordnung der Lösungsverschiebungen bei Übertragung ins Auge. Benachbarte Lösungen verschieben sich dabei in ähnlicher Weise. Bis zu Lösungen um 70% der Maximalkosten findet in der Übertragung eine Verschiebung der Lösungen hin zu höheren Kosten statt, wobei die prozentuale Verschiebung in Kostenrichtung ungefähr der prozentualen Verschiebung in Wartezeitrichtung entspricht. Auf diese Art und Weise erreichen Konfigurationen nach Übertragung eine ähnliche Bilanz wie zuvor. Die Übertragungsexperimente der parametrisierten Versionen des Q-Learning (Abbildung 11.22) und der Bilanzsummierung (Abbildung 11.23) hingegen liefern ein anderes Ergebnis. Lösungen die bei fünfmonatiger Simulation noch Teil einer der zuvor beschriebenen Lösungsinseln waren, schließen sich nun mit anderen Lösungen zusammen oder fächern sich über den Lösungsraum auf. Das Resultat ist ein weniger stabiles Verhalten in der Übertragung. Teilweise liegen zwischen den Lösungen sogar mehr als 20% Abweichung der Kosten gegenüber der Trainingssequenz (z.B. einige der Insellösungen um die 30% Kosten beim Summierungsansatz (Abbildung 11.23), die bei sechsmonatiger Simulation ca. 50% der Gesamtkosten erzeugen.)

Die Unterschiede in der Übertragungsstabilität zeigen sich auch in Tabelle 11.8, welche die durchschnittlichen Übertragungsfehler zwischen den Konfigurationen der Ansätze für alle Workloads zeigt.

Ein Übertragungsfehler (UF) wird dabei betrachtet als die Summe der prozen-

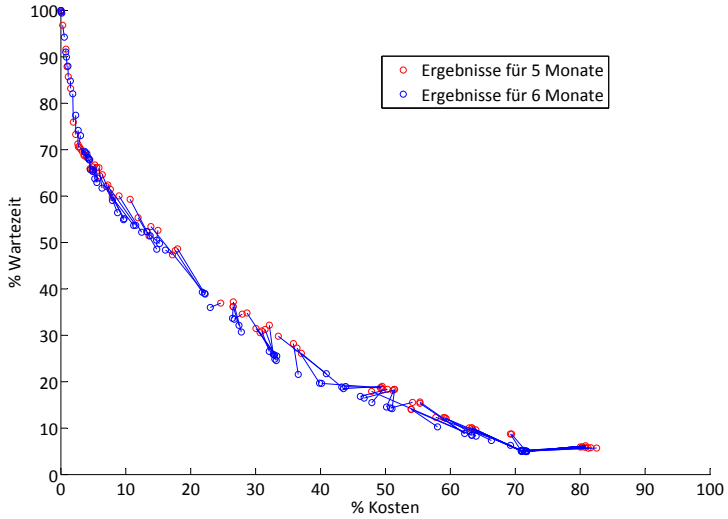


ABBILDUNG 11.22: *Normierte Darstellung der Übertragungsexperimente bei Anwendung des KTH-Workloads für Q-Learning mit Abtastung über* 100 *Kostengewichte.*

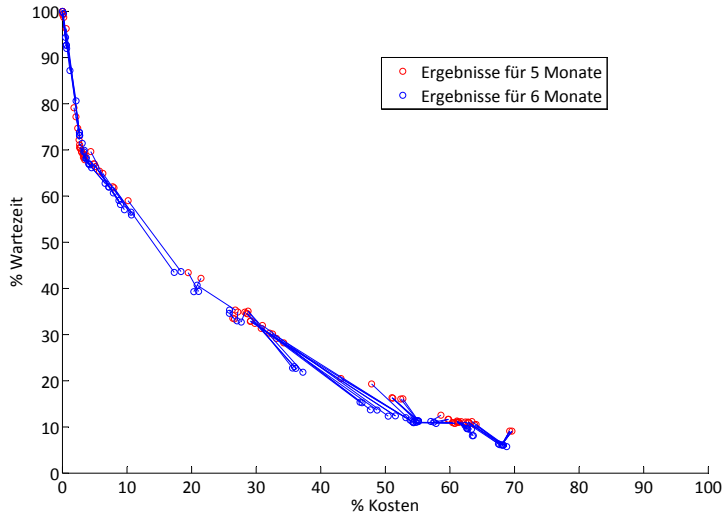


ABBILDUNG 11.23: *Normierte Darstellung der Übertragungsexperimente bei Anwendung des KTH-Workloads für die Bilanzsummierung mit Abtastung über* 100 *Kostengewichte.*

tualen Verschiebung in den höheren Kostenbereich und der Verschiebung in den höheren Wartezeitbereich. Formal entspricht die Verschlechterung demnach

$$UF = \max(0, (\mathrm{TWT}_6 - \mathrm{TWT}_5)) + \max(0, (\mathrm{C}_6 - \mathrm{C}_5)).$$

Durch die Wahl dieser Abweichungsmetrik wird sichergestellt, dass dominierende Lösungen — also solche mit geringeren Kosten und gleichzeitig geringerer Wartezeit — bei der Übertragung nicht bestraft werden. Lösungen, die sich in ei-

nem der Werte auf Kosten des jeweils anderen Wertes verbessern, werden hingegen mit dem erhöhten Wert bestraft.

Übertragungsansatz	KTH	SDSC00	CTC	SDSC03	SDSC05
Statische Abtastung	1,05	0,84	1,79	0,57	0,44
Q-Learning mit Kostengewicht	2,42	2,44	7,53	7,87	2,73
Bilanzsummierung mit Kostengewicht	3,66	3,36	8,98	8,12	1,65

TABELLE 11.8: *Ergebnisse für die durchschnittlichen Übertragungsfehler bei Übertragung der Konfigurationen von den Trainings- auf die Anwendungssequenzen für alle Eingabedaten.*

Wie schon zuvor in den Abbildungen 11.21 bis 11.23, zeigt sich auch in der Tabelle, dass die Übertragung der parametrisierbaren Online-Lernansätze bei Vorhandensein von Wissen schlechter funktioniert als eine direkte Wahl von statischen Konfigurationen. Mit Ausnahme der Ergebnisse für den SDSC05-Workload kommt es bei der Bilanzsummierung im Mittel sogar zu noch größeren Übertragungsfehlern als beim Q-Learning.

Als Ergebnis dieser Untersuchungen ist demnach festzuhalten, dass ein Online-Lernansatz (QL oder SUM) immer dann eingesetzt werden sollte, wenn kein Wissen über die Dynamik des Systems vorhanden ist und lediglich hinsichtlich des besten Kosten-Nutzen-Verhältnisses optimiert werden soll.

Möchte man hingegen für einen Übertragungszeitraum ausgehend von den Erfahrungen bzgl. eines Trainings-Workloads in einen ähnlichen relativen Zielbereich aus prozentualen Kosten und prozentualer Wartezeit kommen, so empfiehlt es sich, direkt eine der zuvor evaluierten statischen Konfigurationen zu wählen, da hier mit den geringsten Abweichungen zu rechnen ist.

Kapitel 12

Abschluss von Teil III

Dieser Teil der Arbeit befasste sich mit dem Jobaustausch von Batch-Jobs ausgehend von einer lokalen privaten Cloud (virtualisierter HPC-Cluster) hin zu einer hochskalierbaren (kommerziellen) öffentlichen Cloud.

Neben technischen Aspekten, die zur praktischen Realisierung eines solchen Szenarios beachtet werden müssen, wurden auch bereits veröffentlichte erste algorithmische Ansätze für das Cloud Scheduling diskutiert.

Nach Bereitung der technischen Basis und Einordnung der eigenen Entwicklung in die bestehende Forschung, wurde das eigene algorithmische Konzept Schritt für Schritt beschrieben.

Ausgehend von der Erweiterung des im Grundlagenteil beschriebenen Basismodells für den Austausch von Arbeitslast wurden Ansätze zur Diskretisierung des Entscheidungsraums (1 simulierter Tag), Aktuatoren für die Beeinflussung von leihpolitischen Entscheidungen (V-Parameter vs. Budget) und szenarienspezifische Metriken (Kosten, Bilanz) für die Berechnung der Scheduling-Performanz eingeführt.

Zur Generierung von Referenzergebnissen für die späteren Online-lerngestützten Leihstrategien wurden zwei verschiedene Herangehensweisen diskutiert. Hierbei handelte es sich zum einen um eine statische Abtastung unveränderlicher Budgets für die Leihstrategie und zum anderen um die mehrkriterielle Optimierung der Strategien mit Hilfe von NSGA-II.

Während eine statische Leihstrategie abhängig vom simulierten Workload häufig unterschiedliche Budgeteinstellungen benötigt, um die jeweils beste Bilanz zu erzielen, zeichnet sich die mehrkriterielle Optimierung durch eine starke Anpassung variabler Strategiepfade an den unterliegenden Workload aus, um das Verbesserungspotenzial im Optimalfall und nach vielen Versuchen auszuschöpfen.

Die darauf folgenden Online-lerngestützten Verfahren versuchten — im Gegensatz zu der statischen Abtastung und der mehrkriteriellen Optimierung — eine möglichst vorteilhaften Strategiepfad ohne Mehrfachsimulation des gleichen Workloads, quasi „on the fly“ zu ermitteln. Die hierzu eingeführte parallele Eva-

luation unterschiedlicher Scheduling-Entscheidungen war dabei genauso elementar wie die Adaption der Wissensbasis durch ein angepasstes 1-Step-Q-Learning bzw. Bilanzsummierung von erzielten Bilanzwerten nach Evaluation der Entscheidungen.

Die direkte Gegenüberstellung der beiden Online-Ansätze hat zudem gezeigt, dass diese nahezu gleich gut funktionieren, solange das System nicht aufgrund bestimmter Umstände (z.B. Fehler innerhalb der Maschinenumgebung) einer höheren Flexibilität in der Entscheidungsfindung bedarf.

Die abschließende Untersuchung der Übertragbarkeit aller Konzepte bei Verfügbarkeit von Workload-spezifischen Scheduling-Erfahrungen hat hingegen ergeben, dass in diesem Fall eine einfache statische Konfiguration als am meisten robust anzusehen ist.

Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurden Verfahren zur dynamischen Ressourcenerweiterung bzw. des Lastaustauschs vorgestellt, die dem Betreiber eines Rechenzentrums bzw. einer virtuellen Organisation eine Alternative zur physischen Rekonfiguration seiner/ihrer lokalen Ressourcenumgebung bieten.

Dazu wurde in Teil I der Arbeit eine Einführung in paralleles Rechnen einschließlich einem Basismodell für MPP-Systeme gegeben. Im Zuge dessen wurde auch das unterliegende Online-Scheduling-Problem beschrieben sowie eine Begründung zur Modellierung eines homogenen MPP-Systems gegeben. Zusätzlich wurden mit der Erläuterungen gängiger Notationen, Größen zur Bewertung der Scheduling-Qualität, sowie der Diskussion von Frameworks und realen Lastaufzeichnungsdaten der Grundstein für die späteren Erweiterungsverfahren und deren Evaluation gelegt.

Das unterliegende Modell beschränkte sich demnach auf homogene Ressourcen zur Abarbeitung paralleler rigider Jobs ohne Wissen über die tatsächliche Laufzeit der Jobs. Die lokale Ausführung der Jobs wurde dabei wahlweise mittels der Heuristiken FCFS oder EASY durchgeführt. Die Simulation aller Algorithmen fand mit dem Teikoku Grid-Scheduling-Framework (tGSF) auf Basis realer Aufzeichnungsdaten aus dem Parallel Workloads Archive statt.

Zusätzlich wurde mit NSGA-II neben einkriteriellen Evolutionsstrategien auch ein Optimierungsverfahren zur mehrkriteriellen Offline-Optimierung und mit Q-Learning auch ein Online-Lernverfahren für die spätere Optimierung des Cloud-Scheduling-Verhaltens vorgestellt.

Mit der Aktivitätendelegation in Computational Grids wurde in Teil II die erste von drei möglichen Lastaustauschverfahren besprochen. Durch modulares Hinzufügen eines Grid Ressourcen Management Systems wurde eine Erweiterung des MPP-Basismodells hin zu einer dezentralen Grid-Scheduling-Architektur mit autonomen Grid-Sites (repräsentiert durch einzelne Ressourcenzentren oder auch ganze Grid Communities) vorgenommen.

Der Fokus des Kapitels 7 lag auf der Entwicklung und Analyse vorteilhafter Austauschverfahren für die Aktivitätendelegation. Hierbei mussten die Modellannahmen für das lokale Job- und Maschinenmodell erweitert werden, um den Austausch von Jobs zwischen den Sites adäquat abzubilden. Dabei wurden ausgehend

von dem lokal homogenen Maschinenmodell auch systemweit alle Ressourcen als homogen angenommen, so dass jeder Job mit unveränderten Laufzeiten auf jeder Grid-weiten Teilmenge von Ressourcen ausgeführt werden konnte — allerdings ohne gleichzeitige Ausführung über Ressourcen mehrerer Sites hinweg. Zusätzlich wurden die Einflüsse des Daten-Schedulings (Staging von Ein- und Ausgabedaten) sowie Zeiten für die eigentliche Verhandlung innerhalb des Modells vernachlässigt. Mit den beiden Größen AWRT_I und VRP wurden zudem zwei Bewertungsgrößen eingeführt, mit deren Hilfe sich der Austausch der Last zwischen den Sites hinsichtlich der Steigerung der Scheduling-Qualität und auch quantitativ analysieren lässt.

Daraufhin wurden Strategien zur Aktivitätendelegation in ihre Einzelbestandteile Transfer- und Lokationsstrategie zerlegt, deren Umsetzungen im Anschluss unabhängig voneinander untersucht wurden.

Die Transferstrategie repräsentiert dabei das eigentliche Annahme- und Abgabeverhalten einer Site, da sie für jeden Job auf Basis lokal berechenbarer Eingangsgrößen bestimmt, ob dieser entfernt oder lokal ausgeführten werden soll. Die anschließend betrachtete einfache Heuristik AWF zeigte bereits zu diesem Zeitpunkt auf, dass die Performanz eines Austauschverhaltens stark abhängig von dem lokale verwendeten Scheduling-Algorithmus (FCFS, EASY) ist. So funktionierte AWF in erster Linie am besten, wenn für die lokale Zuteilung mit EASY auf vorhandene Laufzeitschätzungen zurückgegriffen wurde, da ausgetauschte Jobs beim Delegationspartner aufgrund der Trennung zwischen GRMS und LRMS stets an das Ende der Queue angefügt wurden. Dies führte in manchen Setups sogar zu einer Verschlechterung einer der beiden Parteien gegenüber dem Verzicht auf Teilnahme am Grid. Die Heuristik diente im weiteren Verlauf der Untersuchungen als Referenzstrategie für komplexere Austauschstrategien.

Eine solche komplexere Lösung wurde mit einer Austauschstrategie basierend auf Fuzzy-Regelbasen gegeben. Diese kommt mit Hilfe mehrerer Fuzzy-Regeln in Abhängigkeit der normalisierten wartenden Parallelität und der normalisierten Jobparallelität eines betrachteten Jobs über Inferenz zu einer binären Entscheidung, die entweder zur lokalen Ausführung oder im umgekehrten Fall zur Abgabe bzw. Ablehnung eines Jobs führt. Diese Regelbasen wurden im Rahmen paarweiser Trainingsdurchläufe mit mehreren Workloads und unter Verwendung von Evolutionsstrategien hinsichtlich der AWRT ihrer lokalen Nutzergemeinschaft offline optimiert.
Um eine anschließende Robustheitsuntersuchung vornehmen zu können, wurden die realen Lastaufzeichnung in zwei Segmente zu fünf und sechs Monaten aufgeteilt, wobei die ersten fünf Monate stets für das Training von Regelbasen unter einer Strategie und die folgenden sechs Monate für die Übertragung dienten.

Im Vergleich zu AWF weisen die optimierten Fuzzy-Regelbasen ein genau gegenteiliges Leistungsbild auf. So führen sie insbesondere bei FCFS zu weiteren Verbesserungen gegenüber der lokalen Ausführung und dies in vielen Fällen sogar gleichzeitig für beide Delegationspartner. Als wenig robust erweisen sie sich, wenn sie zuvor mit EASY als lokalen Scheduling-Algorithmus trainiert wurden, da es

in diesem Fall zu Fehlanpassungen an die Strategie des Trainingspartners kommt. Daher empfiehlt sich für EASY weiterhin, AWF für den Austausch zu verwenden. Darüber hinaus haben sich die gelernten Regelbasen als recht Site-abhängig erwiesen, so dass sie weder auf fremden Sites für die Steuerung des Austauschs noch bei Wechsel des Partners genutzt werden können, sondern nur bei gleichbleibender Konstellation.

Für die Anwendung trainierter Regelbasen in Grids mit mehr als zwei Teilnehmern, musste eine geeignete Lokationsstrategie gefunden werden, welche für jeden lokalen Job bestimmt, in welcher Reihenfolge Anfragen an potentielle Delegationspartner vorgenommen werden sollen. Im Zuge dessen wurde eine Lokationsstrategie vorgestellt, welche aus der AWRT abgegebener Jobs Rückschlüsse über die Servicegüte der Delegationspartner zieht.

Mit zunehmender Größe eines Grids waren darüber hinaus viele paarweise Trainingsdurchläufe notwendig, was zu einem immensen Rechenaufwand bei der Optimierung geführt hat. Als Alternative wurde daher ein coevolutionäres Optimierungsverfahren eingeführt, welches in der Lage ist, in einem einzigen Optimierungslauf Regelbasen für alle beteiligten Sites zu erstellen. Zusätzlich konnte unter den gefundenen Regelbasis-Kombinationen eines Setups nach bestimmten Kriterien a posteriori eine Kombination selektiert werden, welche entweder die kleineren oder größeren Sites im Grid bevorzugt.

Teil zukünftiger Forschungen könnte die Anwendung von Online-Lernverfahren für die Aktivitätendelegation sein, die sich auch ohne vorangehendes Training ausgehend von einem Standardverhalten z.B. AWF Schritt für Schritt an die Lastsituation und die Konfiguration des Grid anpassen — nach Möglichkeit sogar unabhängig von der Fluktuation der Delegationspartner im betrachteten Grid. Ein solches Verfahren ist auf der einen Seite schwierig umzusetzen, da die ausgeführten Aktionen in ihrer Konsequenz aufgrund des wechselseitigen Austauschs und der Interaktion mit unabhängigen Delegationspartnern schlecht vorauszusehen sind. Auf der anderen Seite, könnte das Verfahren wie AWF auch in unbekannten Grid-Umgebungen eingesetzt werden und bedürfte keiner Trainingsdaten der potentiellen Teilnehmer.

Die Inhalte zur Ressourcendelegation in Kapitel 8 wurden ähnlich strukturiert, wie die der Aktivitätendelegation. So wurde auch hier zunächst auf technische Voraussetzungen sowie bestehende algorithmische Forschungen eingegangen, bevor die Modell-Erweiterung eines einzelnen MPP-Systems vorgenommen wurde.

Auf Basis des Modells wurde eine konfigurierbare Strategie für die Ressourcendelegation untersucht. Hierbei leiht sich eine Site in Abhängigkeit ihrer lokalen Last Ressourcen von einer anderen Site auf unbestimmte Zeit — allerdings mit der Maßgabe, dass sie die Ressourcen zurückgeben muss, falls diese zurückgefordert werden.

Die vorgestellte Strategie konnte über vier binäre Parameter in ihrem Verhalten beeinflusst werden, die beispielsweise das Verhalten bei Rückforderung von Ressourcen oder die Anfrageerstellung in unterschiedlichen Aspekten betrafen.

Nach einer Untersuchung aller möglichen Konfigurationen konnte sich eine einzelne Parametrisierung unabhängig des unterliegenden lokalen Scheduling-Algorithmus durchsetzen. Diese Strategie wurde im Anschluss mit Ansätzen der Aktivitätendelegation verglichen.

Hierbei hat sich gezeigt, dass die AWRT-Performanz der Ressourcendelegationsstrategie bei FCFS vergleichbar mit der Performanz der gelernten Fuzzy-Regelbasen ist — allerdings ohne aufwendigen Trainingsvorgang — und damit auch klar besser als AWF.

Bei EASY hingegen schneidet sie vergleichbar gut wie AWF ab, führt tendenziell aber zu einer Umverteilung der Vorteile des Austauschs von den kleineren Sites hin zu den größeren Sites.

Insbesondere in größeren Grids, die erneut einer Auswahlstrategie zur Sortierung der Delegationspartner bei Ressourcenanfrage und Rückforderung bedurften, hat sich herausgestellt, dass die Rekonfiguration von Ressourcen mitunter extreme Züge annimmt, bei denen eine Site nach längerer Simulation lediglich auf Fremdressourcen rechnet, während sie ihre eigenen Ressourcen komplett verliehen hat.

Für weiterer Untersuchungen im Rahmen der Ressourcendelegation wäre es durchaus interessant, zu untersuchen, in wie weit sich grundlegende Veränderungen im Leihprozess auswirken würden. Dies betrifft beispielsweise den Auslöser des Leihvorgangs. Da die RD-Strategie in ihrer aktuellen Realisierung noch für jeden in das System kommenden Job ausgelöst wird, passt sie sich zwar sehr dynamisch an die Lastsituation an, führt im Gegenzug allerdings zu einer vergleichsweise großen Anzahl von Rekonfigurationen und Nachrichten zwischen den Parteien. An dieser Stelle könnte untersucht werden, wie sich alternative Auslöser (z.B. in festen Zeitintervallen oder erst für eine bestimmte Menge an Jobs) auf die Scheduling-Qualität bei Reduzierung der Rekonfigurationen auswirken.

Eine andere Veränderung des Leihprozesses könnte darin bestehen, dass Ressourcen nicht — wie bisher — für unendliche Zeit mit aktiver Rückforderung, sondern stets mit einer festen Leihdauer und der damit verbundenen Gefahr der Fragmentierung des Ressourcenraums verliehen werden würden.

Die feste Leihdauer könnte beispielsweise auf Basis der geschätzten Laufzeit von Jobs ermittelt werden und würde somit auch die geschätzte Laufzeit mit in die Entscheidungen auf Grid-Ebene einbeziehen.

Ein solcher Einbezug von geschätzten Laufzeiten könnte ebenso Auswirkungen auf die Performanz der optimierten Fuzzy-Regelbasen bei der Aktivitätendelegation haben, was nach entsprechender Anpassung der Regelkodierung ebenfalls weiter untersucht werden müsste.

Mit Teil III wurde in der Arbeit eine Alternative für den wechselseitigen Austausch von Arbeitslast präsentiert. Hier wurden neben grundlegenden Begrifflichkeiten des Cloud Computing Verfahren des Scientific Cloud Computings präsentiert, die belegen, dass wissenschaftliche Anwendungen (auch in Form paralleler Jobs) durchaus in eine öffentliche Cloud ausgelagert werden können.

Kern des Kapitels 11 war die Evaluation eigener Verfahren für die Steuerung der

dynamischen Erweiterung lokaler Ressourcen um Cloud-Ressourcen. Dazu wurde ausgehend vom Kostenmodell eine Bilanzfunktion definiert, welche helfen sollte, das Verhältnis zwischen den Kosten für das Ausleihen von Ressourcen aus der Cloud und dem Nutzen im Sinne einer reduzierten Wartezeit für ausgelagerte Rechenaufgaben bemessen zu können. Während die tatsächliche Anpassung der Ressourcen in Abhängigkeit des Workloads vorgenommen wurde, wurden Strategieentscheidungen zeitlich diskretisiert, so dass ein sogenannter Strategiepfad entstand. Dieser repräsentierte eine Abfolge von Parameterausprägungen, die jeweils für ein Entscheidungsintervall (der Länge eines Tages) gelten. Die Abfolge der Ausprägungen hatte daraufhin Einfluss auf das Leihverhalten einer Site und damit auch darauf, ob sie eher ihre Scheduling-Qualität gesteigert oder sich restriktiv im Hinblick auf entstehende Kosten verhalten hat.

Der Vergleich von zwei verschiedenen Strategieparametern hat ergeben, dass es in Bezug auf das Kosten-Nutzen-Verhältnis sinnvoller ist, der Site für ein Entscheidungsintervall ein kostenorientiertes Budget zur Verfügung zu stellen, als die Maximalzahl gleichzeitig aktiver Leihressourcen zu steuern. Allerdings bedurfte diese Herangehensweise auch der direkten Einbindung von geschätzten Laufzeiten in die Cloud Scheduling-Entscheidungen und ist damit auch nur für Sites geeignet, deren Nutzer die entsprechenden Information bereitstellen.

Gleichzeitig wurde durch die Definition der Strategiepfade erneut ein Optimierungsproblem modelliert, welches im Anschluss zunächst mehrkriteriell mit NSGA-II hinsichtlich erreichbarer Kosten-Wartezeit-Lösungen optimiert wurde. Diese Ergebnisse sollten später zur Abschätzung einer oberen Grenze für die erreichbare Bilanz durch Online-Verfahren dienen.

Als Online-Verfahren wurde Q-Learning gewählt, welches für jede Aktion im System (unterschiedliche Budget-Konfigurationen) den geschätzten Nutzen verwaltet, der wiederum von einer Bilanz-gestützten Belohnungsfunktion abhängig ist. Diese Funktion nutzte es aus, dass beim Cloud-Scheduling im Gegensatz zum Grid-Scheduling nur die Entscheidungen der leihenden Site Einfluss auf die Konsequenzen einer jeden Aktion besitzen. Somit wurde die gleichzeitige Simulation alternativer Entscheidungen möglich, während die Aktion mit dem größten momentanen Nutzen das aktuelle Verhalten bestimmte.

Mit der Q-Learning-Strategie konnte für das Cloud Scheduling so ein Online-Lernverfahren etabliert werden, welches sich dynamisch an Änderungen der Last anpasst und ohne Vorwissen über den zu erwartenden Workload eingesetzt werden kann. Dabei versucht es stets ein vorteilhaftes Gleichgewicht zwischen den Kosten für die Auslagerung und der damit verbundenen Steigerung der Scheduling-Performanz zu erzielen. Eine Vereinfachung des Lernverfahrens durch simples Aufsummieren der erreichten Belohnungen einer Aktion führte trotz minderer Dynamik in der Anpassung zu vergleichbar guten Ergebnissen, solange es im lokalen Ressourcenraum nicht zu Maschinenausfällen kam.

Abschließend wurde noch untersucht, welches der zuvor vorgestellten Verfahren sich ausgehend von einer vorangehenden Untersuchung des Site-Verhaltens auf

Trainingsdaten am besten eignet, um auf späteren Anwendungsdaten eine bestimmte Kosten-Wartezeit-Lösung zu erreichen.

Hier hat sich gezeigt, dass alle lerngestützten Verfahren aufgrund ihrer Bilanz-orientierten Belohnungsfunktion dazu neigen, auf neuen Daten sprunghaft in anderen Lösungsbereichen zu landen. Daher ist es in der Übertragung des Cloud Scheduling-Verhaltens besser, das durch die Untersuchung hinzugewonnene Wissen auszunutzen und auf eine statische Budget-Konfiguration zurückzugreifen, die am besten zum anvisierten Lösungsbereich passt.

Da das grundlegende Modell in Hinblick auf die Realität eine starke Vereinfachung darstellt, könnte eine Annäherung des Modells auch Einfluss auf die Leistungsfähigkeit der propagierten Strategien besitzen. Dies umfasst beispielsweise die Betrachtung komplexerer Kostenmodelle (z.B. mit festen Abrechnungsintervallen) oder die Einführung von Verzögerungen in der Ausführung ausgelagerter Jobs durch die dynamische Bereitstellung von virtuellen Maschinen in der Cloud.

Ähnliches gilt auch für die Einflüsse eventueller Eingabe- und Ausgabedaten von Jobs, da diese sowohl beim Grid-Scheduling (unabhängig der Delegationsart) als auch bei der Auslagerung von Arbeitslast in die Cloud Auswirkungen auf das Scheduling-Verhalten haben können. Im Falle des Cloud-Schedulings wäre mit der Einführung des Daten-Schedulings auch eine komplexere Modellierung des Kostenmodells verbunden — einerseits für den Transfer und andererseits für die Speicherung von Daten.

Von einer solchen Erweiterung des Modells wären in erster Linie die eingeführten Heuristiken (AWF, beste gefundene RD-Strategie) betroffen, da diese nicht in der Lage sind, sich an das geänderte Modell anzupassen. Alle vorgestellten gelernten Verhalten (Fuzzy-AD, NSGA-II, Q-Learning) hingegen würden sich an das veränderte Modell anpassen, müssten allerdings im Hinblick auf eben diese Anpassung hin untersucht werden.

Eine weitere Untersuchungsmöglichkeit bestünde auch in der Betrachtung eines hybriden Modells, welches sowohl das Grid- als auch das Cloud-Scheduling miteinander kombiniert. So könnten Sites innerhalb dieses Modells für nicht ausführbare Jobs zunächst eine Auslagerung per Aktivitätendelegation versuchen, da diese nicht zu zusätzlichen Kosten führen. Erst, wenn die Delegation des Jobs nicht funktioniert hat, würde die Site versuchen, einen Job in die Cloud auszulagern. Auf die Art und Weise könnten die Stärken beider verteilter Architekturen ausgenutzt werden.

Anhang: Inhalt der Daten-CD

Im Zuge guter wissenschaftlicher Praxis wurde der Arbeit ebenfalls eine Daten-CD beigelegt, welche die benutzen Simulationstools sowie Eingabe- und Simulationsdaten enthält. Darüber hinaus enthält sie auch eine Reihe von Matlab-Skripten, welche zur Erstellung der Abbildungen und Tabellen in der Arbeit genutzt wurden.

Übersicht über die Inhalte

Ergebnisse : Hier liegen sowohl Rohdaten als auch Ergebnisse der Simulationen.

Matlab-Skripte : Die Skripte, welche für die Erstellung der Abbildungen genutzt wurden sind in diesem Verzeichnis enthalten.

Quellen : Dieser Ordner enthält zum einen alle eigenen Veröffentlichungen, die in der Arbeit referenziert wurden und zum anderen auch eine Kopie der zitierten Webquellen. Letztere wurden mit Hilfe des Programms (HTTrack) bis zu einer Link-Tiefe von 3 aus dem Web extrahiert, um auch in Zukunft die Nachvollziehbarkeit der Zitate sicherstellen zu können. Die dabei erstellte index.html erleichtert dabei die Navigation.

Simulation : Dieser Ordner enthält die Quelldateien und eine vorkompilierte JAR-Version des Teikoku Grid Scheduling Frameworks. Zusätzlich ist auch eine Anleitung zum Starten von Simulationen gegeben. Die Quelldateien liegen in Form eines Maven2-Projekts vor.

Cluster : Dieser Ordner enthält die Quelldateien zu der selbst erstellten Cluster-Software.

Dokument : Dieser Ordner enthält die LaTeX-Quelldateien zu der Dissertation in der Abgabeversion.

Layout : Dieser Ordner enthält die Quelldateien für das DVD-Design.

Ergebnisdateien und Skripte

- Tabelle 7.1 (S. 68) und Tabelle 7.2 (S. 69):
  Ergebnisse der AWF-Evaluation
  Datenpfad: „/Ergebnisse/AD/AWFResults.txt“

- Tabelle 7.3 (S. 77), Tabelle 7.4 (S. 79) und Tabelle 7.5 (S. 80):
  Ergebnisse der paarweisen Lerndurchgänge für FCFS und EASY
  Datenpfad: „/Ergebnisse/AD/Lernen-5Monate/“
  Unter „/Ergebnisse/AD/Datenaufbau1.txt“ gibt es eine genaue Erklärung zum Aufbau der enthaltenden Daten.

- Tabelle 7.6 (S. 83), Tabelle 7.7 (S. 83) und Tabelle 7.8 (S. 84):
  Ergebnisse der Übertragungsexperimente
  Datenpfad: „/Ergebnisse/AD/Anwenden-6Monate/“
  Unter „/Ergebnisse/AD/Datenaufbau1.txt“ gibt es eine genaue Erklärung zum Aufbau der enthaltenden Daten.

- Abbildung 7.6 (S. 81) und Abbildung 7.7 (S. 84):
  Darstellung von ausgewählten trainierten Regelbasen und Zugriffen auf diese während der Simulation
  Datenpfad: „/Ergebnisse/AD/Lernen-5Monate/“
  Skript: „/Matlab-Skripte/AD/FuzzyRegelbasenPlotten/plotSingleFace.m“
  Das Skript ist bereits für die Berechnung der Abbildung 7.6(a) vorkonfiguriert.

- Tabelle 7.9 (S. 86):
  Anwendung FCFS gelernter Regelbasen mit EASY
  Datenpfad: „/Ergebnisse/AD/AnwendungFCFS-aufEASY/“
  Unter „/Ergebnisse/AD/Datenaufbau1.txt“ gibt es eine genaue Erklärung zum Aufbau der enthaltenden Daten.

- Abbildung 7.8 (S. 88) und Abbildung 7.10 (S. 90):
  Lokationspolitikergebnisse bei Anwendung auf 6 Monate und bei Verwendung von FCFS gelernten Regelbasen unter FCFS und EASY
  Datenpfad: „/Ergebnisse/AD/Lokationsstrategie/“
  In jedem der vier Experimente-Ordner befinden sich:
  - Die Regelbasen einer jeden Site gegen jede andere Site (rb1-2.txt ist beispielsweise die Regelbasis der Site1 gegen Site2 gelernt)
  - Die „grid.properties“-Datei für die Teikoku-Simulation des Setups
  - Alle Metriken der Simulation im „Metrics“-Ordner
  - „ExchangeMatrix.txt“ enthält die prozentualen Verschiebungen des Ressourcenprodukts

  Skripte: „/Matlab-Skripte/AD/Lokationsstrategie/*.m“
  Das Skript „plotGroessereGrids.m“ ist bereits vorkonfiguriert für das Plotten des 3-Site-Setups mit FCFS (Abbildung 7.8(a)).

- Abbildung 7.9 (S. 89):
  Prozentuale Verschiebung von Arbeitslast zwischen den Sites bei größeren Grids. Die Daten stammen aus der zuvor angesprochenen Datei „ExchangeMatrix.txt“ einer jeden Konfiguration.

- Abbildung 7.12 (S. 96):
  Vergleich von COEA-Läufen mit klassischem EA und AWF
  Datenpfade:
  - „/Ergebnisse/AD/Lokationsstrategie/FuzzyLaeufe/“ für die klassischen EA-Durchläufe
  - „/Ergebnisse/AD/COES/“ für die coevolutionären Durchläufe
  - „/Ergebnisse/AD/Lokationsstrategie/AWF“ für die AWF-Resultate

  Skript: „/Matlab-Skripte/AD/Lokationsstrategie/plot_COES_Vergleich.m“ ist für den Plot des 3-Site-Setups der Abbildung vorkonfiguriert.

- Abbildung 8.3 (S. 109), Tabelle 8.1 (S. 110) und Tabelle 8.2 (S. 111):
  Analyse der Performanz der Strategieparameter zur Ressourcendelegation
  Datenpfad: „/Ergebnisse/RD/RD-Abtastung.txt“
  Skript: „/Matlab-Skripte/RD/analysiereRD_Strategien.m“
  In der vorliegenden Version ist das Skript so eingestellt, dass es auf der vorhandenen Datengrundlage einen separaten Vergleich für alle Algo-/Monats-Paarungen ausgibt.

- Abbildung 8.4 (S. 113):
  Ressourcenaustauschverlauf der besten RD-Strategie bei gegebenem Setup
  Datenpfad: „/Ergebnisse/RD/KonkreteErgebnisse/“
  Die Metrik „rdtrac“ einer Site gibt den zeitlichen Verlauf der verliehenen Ressourcen nach Ziel an und ermöglicht so die Erstellung der Abbildung über die Zeit.

- Abbildung 8.5 (S. 116):
  Analyse der Auswahlstrategien bei mehreren Partnern in der Ressourcendelegation
  Datenpfad: „/Ergebnisse/RD/Austauschstrategie/“
  Skript: „/Matlab-Skripte/RD/RD_Lok_Vergleich.m“ ist für die Analyse in Abbildung 8.5 vorkonfiguriert.

- Abbildung 8.6 (S. 118):
  Vergleich der besten RD-Strategien mit AWF und EA der Aktivitätendelegation
  Datenpfad: „/Ergebnisse/RD/Austauschstrategie/“
  Skript: „/Matlab-Skripte/RD/plotRD_Lok_Strategie.m“ ist für den Plot in Abbildung 8.6 vorkonfiguriert.

- Abbildung 8.7 (S. 119) und Abbildung 8.8 (S. 120):
  Relative Leihverläufe zwischen den Sites bei Anwendung der besten Austauschstrategie und der besten RD-Strategie
  Datenpfad:
  „/Ergebnisse/RD/Austauschstrategie/5Sites/FCFS/6Monate/SavedMetrics/4/“
  Der Pfad führt zu den Ergebnissen des konkreten Experiments.
  Skript: „/Matlab-Skripte/RD/percentualRDTrac.m“ produziert in der vorgegebenen Konfiguration direkt beide Abbildungen.

- Tabelle 11.1 (S. 141):
  Diskretisierung der Entscheidungen zum Cloud-Scheduling
  Datenpfad: „/Simulation/traces/“
  Unter „/Ergebnisse/HybridCloud/perDaySubmission.m“ sammelt alle betrachteten Workflows ein und analysiert diese bzgl. der präsentierten Größen.

- Abbildung 11.6 (S. 146) und Abbildung 11.7 (S. 147):
  Vergleich der beiden Aktuatoren V- und Budget-Parameter
  Datenpfade: „/Ergebnisse/HybridCloud/Abtastungsexperimente-Budget/“ und „/Ergebnisse/HybridCloud/Abtastungsexperimente-V/“ enthalten die Daten für die jeweiligen Abtastungsexperimente.
  Skript: „/Matlab-Skripte/HybridCloud/plotVundBudgetAbtastungImVergleich.m“ stellt die Ergebnisse beider Aktuatoren auf allen elfmonatigen Workloads einander gegenüber (vgl. Abbildung 11.7).

- Abbildung 11.9 (S. 151):
  Darstellung der Frontentwicklung bei einer NSGA-II-Optimierung
  Datenpfade: „/Ergebnisse/HybridCloud/NSGA2/“ und „/Ergebnisse/HybridCloud/Abtastungsexperimente-Budget/“
  Skript: „/Matlab-Skripte/HybridCloud/frontDarstellungFuerArbeit.m“ ist für die fünfmonatige Frontdarstellung des KTH wie in der Abbildung vorkonfiguriert.

- Abbildung 11.10 (S. 153):
  Beispielhafte Darstellung eines einzelnen Entscheidungsverlaufs des besten Individuums nach einer 500-Generationen NSGA-II-Optimierung
  Datenpfade: „/Ergebnisse/HybridCloud/NSGA2/“ und „/Ergebnisse/HybridCloud/Entscheidungen/“
  Skript: „/Matlab-Skripte/HybridCloud/Entscheidungen.m“ ist mit der fünfmonatigen KTH-Optimierung vorkonfiguriert, die zu dem dargestellten Bild führt.

- Abbildung 11.12 (S. 160):
  Analyse der RL-Lernparameter α und γ
  Datenpfad:
  „/Ergebnisse/HybridCloud/AlphaGammaAnalyse/FitnessLandscape-KTH.txt“
  Skript: „/Matlab-Skripte/HybridCloud/plotFitnessLandscape.m“ erzeugt das besagte Bild der Analyse.

- Abbildung 11.13 (S. 161):
  Unterschiedliches Entscheidungsverhalten zweier Strategien bei leicht verändertem γ
  Datenpfad:
  „/Ergebnisse/HybridCloud/AlphaGammaAnalyse/GammaReaktivitaet“
  Die Abbildung entspricht den Plots der einzelnen Entscheidungen der beiden am Datenpfad abgelegten Simulationsergebnissen. Die Entscheidungen selbst sind in der Variable „newActionIndex“ in der Metrik „../Metrics/LearningStrategyMetric“ hinterlegt.

- Abbildung 11.14 (S. 162):
  Normierter Vergleich unterschiedlicher α-Einstellungen hinsichtlich ihrer Bilanz auf allen 5 Workloads
  Datenpfad: „/Ergebnisse/HybridCloud/AlphaGammaAnalyse/“
  Skript:
  „/Matlab-Skripte/HybridCloud/plotFitnessLandscapeOnlyAlpha5Site.m“
  erzeugt das besagte Bild der Analyse.

- Tabelle 11.5 (S. 163) und Tabelle 11.6 (S. 166):
  Gegenüberstellung der durch QL erreichten Bilanz und den Bilanzen der durch Abtastung und NSGA-II-Optimierung ermittelten Bilanzwerte
  Datenpfad: „/Ergebnisse/HybridCloud/CloudScheduling.xls“
  Hier befinden sich alle zugrundeliegenden Daten für die Budget-Abtastung, die gemittelten NSGA-II-Läufe, alle QL-Läufe und auch die Untersuchungen des Summierungsansatzes.

- Abbildung 11.15 (S. 164):
  Entscheidungsverläufe von NSGA, QL und Abtastung im Vergleich anhand des elfmonatigen KTH-Workloads
  Datenpfad: „/Ergebnisse/HybridCloud/“
  Da es sich um eine vergleichende Studie mehrerer Experimente handelt (NSGA-II, QL, Abtastung) befinden sich die Daten in entsprechenden Unterordnern und der dort abgelegten Excel-Tabelle.
  Skript: „/Matlab-Skripte/HybridCloud/aktionenVergleichen.m“
  Achtung: Das Skript ist in der gegebenen Version für das Plotten der nachfolgenden Abbildung konfiguriert. Mit Hilfe der Kommentare im Skript kann es jedoch so konfiguriert werden, dass es die Abbildung 11.15 erstellt.

- Abbildung 11.16 (S. 167):
  Vergleich der Entscheidungsverläufe bei statischer Konfiguration, QL und dem Summierungsansatz im Vergleich anhand des Beispiels SDSC05
  Datenpfad:„/Ergebnisse/HybridCloud/Entscheidungen/“
  Da es sich um eine vergleichende Studie mehrerer Experimente handelt (NSGA-II, QL, Abtastung) befinden sich die Daten in entsprechenden Unterordnern und der dort abgelegten Excel-Tabelle.
  Skript: „/Matlab-Skripte/HybridCloud/aktionenVergleichen.m“ erzeugt das besagte Bild der Analyse.

- Abbildung 11.18 (S. 171) und Tabelle 11.7 (S. 172):
  Vergleich der beiden Online-Lernansätze QL und Aufsummierung im Fehlerfall
  Datenpfad: „Ergebnisse/HybridCloud/Fehleruntersuchungen/“
  - „../OriginalLogs-LANL/LA-URL-05· · ·-2005.csv“ ist die Originaldatei des Fehlerarchivs; Quelle: Computer Failure Data Repository (CFDR)
  - „../OriginalLogs-LANL/PFunktionen/“
    Hier liegen für jedes System die Time-Between-Failure- und Repair-Time-Werte, die in den Original-Logs auftauchen und für die konkrete wahrscheinlichkeitsbasierte Fehler-Trace-Erstellung genutzt wurden.

  Skripte: „/Matlab-Skripte/HybridCloud/Fehleruntersuchung/*.m“
  - „../vergleicheRLundSumImFehlerfall.m“
    Das Skript erstellt auf Wunsch neue Fehler-Traces nach dem in der Arbeit beschriebenen Prinzip und simuliert diese auch direkt mit Teikoku.
  - „../errorTraceGenerator.m“
    Hilfsskript zur Erzeugung eines einzelnen Fehler-Traces
  - „../errorTraceGeneratorKit5and6.m“
    Hilfsskript für das Zusammenlegen von Traces von 5 und 6 Monaten Länge zu einem Verbund-Trace zu 11 Monaten

- Abbildung 11.19 (S. 173):
  Beispieldarstellung verschiedener Strategien im Bezug zur NSGA-II-Front und Abtastung zzgl. Angabe eines anvisierten Kostenbereichs in einem SDSC03-5-Monats-Setup
  Datenpfad: „/Ergebnisse/HybridCloud/“
  Umfasst verschiedene Unterordner mit Ergebnissen, die in die Abbildung eingezeichnet sind. Diese sind ebenfalls detailliert innerhalb des nachfolgenden Matlab-Skripts zu finden.
  Skript: „/Matlab-Skripte/HybridCloud/loesungenUndZiel.m“ ist vorkonfiguriert, um die vorliegende Abbildung zu erstellen.

- Abbildung 11.20 (S. 176):
  Lösungsinseln vor der Übertragung (QL und SUM im Vergleich)
  Datenpfad: „/Ergebnisse/HybridCloud/“
  Umfasst verschiedene Unterordner mit Ergebnissen, die in die Abbildung eingezeichnet sind. Diese sind ebenfalls detailliert innerhalb des nachfolgenden Matlab-Skripts zu finden.
  Skript:
  „/Matlab-Skripte/HybridCloud/frontDarstellungFuerArbeitUebertragung.m“
  ist vorkonfiguriert, um die vorliegende Abbildung zu erstellen.

- Abbildung 11.21 bis Abbildung 11.23 (S. 177 bis 178):
  Darstellung der Abweichungen in der Übertragung von 5 Monaten auf 6 Monate Simulationszeit je nach Übertragungsansatz
  Datenpfad: „/Ergebnisse/HybridCloud/“
  Umfasst verschiedene Unterordner mit Ergebnissen, die in die Abbildung eingezeichnet sind. Diese sind ebenfalls detailliert innerhalb des nachfolgenden Matlab-Skripts zu finden.
  Skript: „/Matlab-Skripte/HybridCloud/abweichungs_analyse.m“ ist für die Erstellung der drei Abbildungen vorkonfiguriert.

Anhang: Architektur einer Cluster-Simulation

Bei nahezu allen Evaluationen dieser Arbeit handelt es sich um (embarrassingly parallel) Parameterstudien auf Basis des Teikoku Grid Scheduling Frameworks (tGSF). Zur Erzeugung, Koordinierung, Submission und Überwachung der parallelen Simulationen wurden mehrere modulare Softwarepakete geschaffen. Abbildung 1 gibt eine Übersicht über die verwendete Architektur und den für

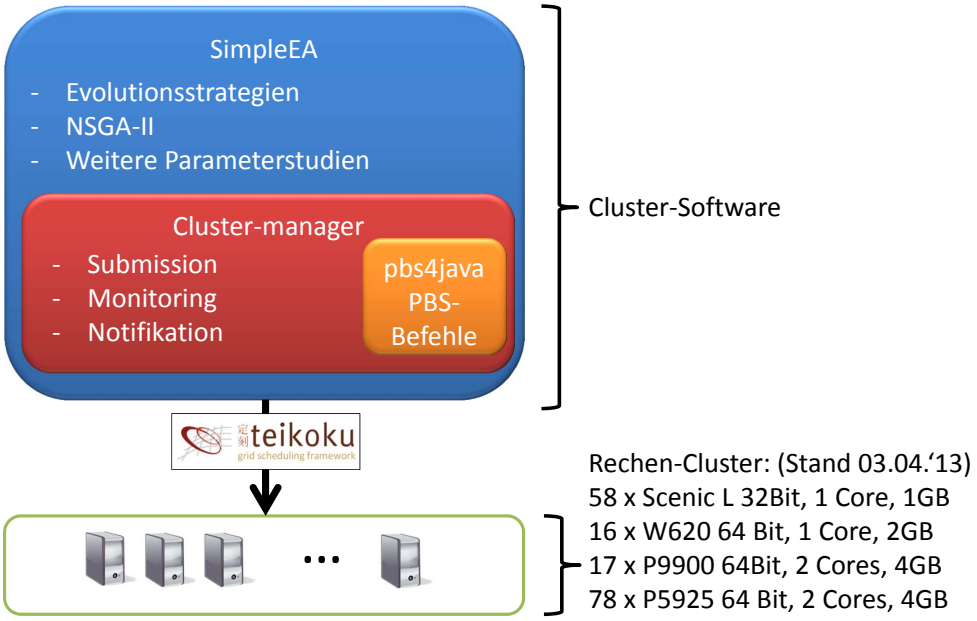


Abbildung 1: *Übersicht einer Cluster-Simulation.*

die Evaluationen genutzten Rechencluster des Lehrstuhls. Auf oberster Ebene existiert das Projekt „simpleEA“, welches die Cluster-unabhängige Implementierung unterschiedlicher Parameterstudien umfasst wie die in der Arbeit verwendeten Evolutionsstrategien (vgl. Abschnitt 7.2), NSGA-II oder Abtastungsexperimente des hybriden Cloud Schedulings (vgl. Kapitel 11).
Um die eigentlichen Simulationen zur parallelen Ausführung auf den Cluster zu

bringen, greift das Projekt auf das ebenfalls selbst entwickelte Framework „Clustermanager“ zurück. Dieses führt eine Generation von Individuen, bzw. eine Menge verschiedener Parametersätze transparent auf einem Rechencluster aus, nimmt — sofern gewünscht — eine Vorverarbeitung von Ergebnisdaten vor und liefert jene an das aufrufende Projekt zurück.
Für die Basisfunktionalität zum Einreichen und Überwachen einzelner Jobs greift das Framework wiederum auf eine modifizierte Version der „pbs4java“-Bibliothek von Mohamed M. El-Kalioby[1] zurück.
Die einzelnen parallelen Jobs entsprechen den Evaluationen der Grid- und Cloud-Scheduling-Strategien mit Hilfe des tGSF.
Diese wurden auf dem Cluster des Lehrstuhls für Datenverarbeitungssysteme (Fakultät ET/IT) der TU Dortmund ausgeführt, welcher aus verschiedenen Rechnersystemen mit $1-2$ Cores sowie $1-4$ GB RAM ausgestattet sind.

[1] http://code.google.com/p/pbs4java/

Literaturverzeichnis

[1] Advanced Micro Devices Inc. (2005). *AMD64 Virtualization Codenamed „Pacifica“ Technology - Secure Virtual Machine Architecture Reference Manual.* AMD. http://www.mimuw.edu.pl/~vincent/lecture6/sources/amd-pacifica-specification.pdf, Zugriff: 16. Januar 2012.

[2] Al-Azzoni, I. & Down, D. (2009). Decentralized Load Balancing for Heterogeneous Grids. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, (S. 545–550).

[3] Badger, L., Grance, T., Patt-Corner, R., & Voas, J. (2012). Cloud Computing Synopsis and Recommendations. http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf, Zugriff: 16. Januar 2013.

[4] Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V., & Weeratunga, S. K. (1991). The NAS Parallel Benchmarks-Summary and Preliminary Results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, (S. 158–165)., New York, NY, USA. ACM.

[5] Bailey Lee, C., Schwartzman, Y., Hardy, J., & Snavely, A. (2005). Are user runtime estimates inherently inaccurate? In *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing*, JSSPP'04, (S. 253–263)., Berlin, Heidelberg. Springer-Verlag.

[6] Beyer, H. G. (2001). *The Theory of Evolution Strategies.* Berlin: Springer.

[7] Bianco, P., Lewis, G. A., & Merson, P. F. (2008). Service Level Agreements in Service-Oriented Architecture Environments. Technischer Bericht Carnegie Mellon University.

[8] Birkenheuer, G., Carlson, A., Fölling, A., Högqvist, M., Hoheisel, A., Papaspyrou, A., Rieger, K., Schott, B., & Ziegler, W. (2009). Connecting Communities on the Meta-Scheduling Level: The DGSI Approach! In *Proceedings of the Cracow Grid Workshop (CGW).*

[9] Buyya, R. & Murshed, M. (2002). GridSim: a Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *Concurrency and Computation: Practice and Experience*, *14*, 1175–1220.

[10] Casanova, H. (2001). SimGrid: A Toolkit for the Simulation of Application Scheduling. In *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, (S. 430–437). IEEE Computer Society Press.

[11] Casanova, H., Legrand, A., & Quinson, M. (2008). SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In *Proceedings of the 10th International Conference on Computer Modeling and Simulation*, (S. 126–131)., Washington, DC, USA. IEEE Computer Society.

[12] Chapin, S. J., Cirne, W., Feitelson, D. G., Jones, J. P., Leutenegger, S. T., Schwiegelshohn, U., Smith, W., & Talby, D. (1999). Benchmarks and Standards for the Evaluation of Parallel Job Schedulers. In *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 1659 von *Lecture Notes in Computer Science*, (S. 67–90). Springer.

[13] Chen, P.-Z. & Chen, S.-N. (2000). A New Program-Driven Parallel Machine Simulation Environment. *Journal of Information Science and Engineering*, *16*, 201–224.

[14] Cirne, W. & Berman, F. (2001). A Comprehensive Model of the Supercomputer Workload. In *IEEE International Workshop on Workload Characterization*, (S. 140 – 148).

[15] Cloer, T. (2011). Amazon bohrt sein Cloud-HPC auf. http://www.computerwoche.de/a/amazon-bohrt-sein-cloud-hpc-auf,2500220, Zugriff: 16. Januar 2013.

[16] Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems* (2. Aufl.). Springer.

[17] Cordón, O., Herrera, F., Hoffmann, F., & Magdalena, L. (2001). *Evolutionary Tuning and Learning of Fuzzy Knowledge Bases* (1. Aufl.)., Band 19 von *Advances in Fuzzy Systems – Applications and Theory*. World Scientific.

[18] De, P., Gupta, M., Soni, M., & Thatte, A. (2012). Caching VM Instances for Fast VM Provisioning: A Comparative Evaluation. In Kaklamanis, C., Papatheodorou, T., & Spirakis, P. (Hrsg.), *Euro-Par 2012 Parallel Processing*, Band 7484 von *Lecture Notes in Computer Science*, (S. 325–336)., Berlin, Heidelberg. Springer.

[19] Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Schoenauer, M. et al. (Hrsg.), *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN 2000)*, Band 1917 von *Lecture Notes in Computer Science*, (S. 849–858). Springer.

[20] Deelman, E., Singh, G., Livny, M., Berriman, B., & Good, J. (2008). The Cost of Doing Science on the Cloud: The Montage Example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, (S. 1–12)., Piscataway, NJ, USA. IEEE Press.

[21] Dias de Assuncao, M., di Costanzo, A., & Buyya, R. (2009). Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *Proceedings of the 18th ACM international symposium on High performance distributed computing (HPDC09)*, (S. 141–150)., New York, NY, USA. ACM.

[22] Downey, A. (1997). A parallel workload model and its implications for processor allocation. In *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, (S. 112 –123).

[23] Downey, A. B. & Feitelson, D. G. (1999). The Elusive Goal of Workload Characterization. *Performance Evaluation Review*, *26*(4), 14–29.

[24] Du, J. & Leung, J. (1989). Complexity of Scheduling Parallel Task Systems. *SIAM Journal on Discrete Mathematics*, *2*(2), 473–487.

[25] Eager, D., Lazowska, E., & Zahorjan, J. (1986). A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing. *Performance Evaluation*, *6*(1), 53–68.

[26] England, D. & Weissman, J. B. (2005). Costs and Benefits of Load Sharing in the Computational Grid. In *Proceedings of the 10th International Conference on Job Scheduling Strategies for Parallel Processing*, JSSPP'04, (S. 160–175)., Berlin, Heidelberg. Springer.

[27] Ernemann, C., Hamscher, V., Schwiegelshohn, U., Streit, A., & Yahyapour, R. (2002). On Advantages of Grid Computing for Parallel Job Scheduling. In *Proccedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, (S. 39–46). IEEE Computer Society Press.

[28] Ernemann, C., Hamscher, V., & Yahyapour, R. (2002). Economic Scheduling in Grid Computing. In *Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 2537 von *Lecture Notes in Computer Science*, (S. 128–152). Springer.

[29] Ernemann, C., Hamscher, V., & Yahyapour, R. (2004). Benefits of Global Grid Computing for Job Scheduling. In *Proceedings of the 5th IEEE/ACM*

International Workshop on Grid Computing, (S. 374–379). IEEE Computer Society.

[30] Erwin, D. W. & Snelling, D. F. (2001). UNICORE: A Grid Computing Environment. In Goos, G., Hartmanis, J., & van Leeuwen, J. (Hrsg.), *Proceedings of the 7th International Euro-Par Conference (Euro-Par 2001)*, Band 2150 von *Lecture Notes in Computer Science*, (S. 825–834)., Manchester, UK. Springer.

[31] Fallenbeck, N., Picht, H.-J., Smith, M., & Freisleben, B. (2006). Xen and the Art of Cluster Scheduling. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*.

[32] Feitelson, D. (2008). Looking at Data. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, (S. 1–9).

[33] Feitelson, D., Rudolph, L., & Schwiegelshohn, U. (2005). Parallel Job Scheduling — A Status Report. In D. Feitelson, L. Rudolph, & U. Schwiegelshohn (Hrsg.), *Job Scheduling Strategies for Parallel Processing*, Band 3277 von *Lecture Notes in Computer Science* (S. 1–16). Berlin, Heidelberg: Springer.

[34] Feitelson, D. G. (1996). Packing Schemes for Gang Scheduling. In Feitelson, D. G. & Rudolph, L. (Hrsg.), *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, Band 1162 von *Lecture Notes in Computer Science*, (S. 89–110). Springer.

[35] Feitelson, D. G. (2001). Metrics for Parallel Job Scheduling and their Convergence. In Feitelson, D. G. & Rudolph, L. (Hrsg.), *Proceedings of the 7th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 2221, (S. 188–206). Springer.

[36] Feitelson, D. G. & Nitzberg, B. (1995). Job Characteristics of a Production Parallel Scientific Workload on the NASA Ames iPSC/860. In Feitelson, D. G. & Rudolph, L. (Hrsg.), *Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing*, Band 949 von *Lecture Notes in Computer Science*, (S. 337–360). Springer.

[37] Feitelson, D. G. & Rudolph, L. (1998). Metrics and Benchmarking for Parallel Job Scheduling. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, (S. 1–24)., Berlin, Heidelberg. Springer.

[38] Feitelson, D. G. & Tsafrir, D. (2006a). Instability in Parallel Job Scheduling Simulation: The Role of Workload Flurries. In *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Lecture Notes in Computer Science. IEEE Press.

[39] Feitelson, D. G. & Tsafrir, D. (2006b). Workload Sanitation for Performance Evaluation. In *IEEE International Symposium on Performance Analysis of Systems& Software*, (S. 221–230). IEEE Computer Society.

[40] Fenn, M., Holmes, J., & Nucciarone, J. (2011). A Performance and Cost Analysis of the Amazon Elastic Compute Cloud Cluster Compute Instance. Technischer Bericht Research Computing and Cyberinfrastructure Group, Penn State University.

[41] Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2009a). Co-evolving Fuzzy Rule Sets for Job Exchange in Computational Grids. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, (S. 1683–1688). IEEE Computer Society Press.

[42] Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2009b). Decentralized Grid Scheduling with Evolutionary Fuzzy Systems. In Frachtenberg, E. & Schwiegelshohn, U. (Hrsg.), *Proceedings of the 14th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 5798 von *Lecture Notes in Computer Science*, (S. 16–36). Springer.

[43] Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2010a). Robust Load Delegation in Service Grid Environments. *IEEE Transactions on Parallel and Distributed Systems*, *21*(9), 1304–1316.

[44] Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2010b). The Gain of Resource Delegation in Distributed Computing Environments. In Schwiegelshohn, U. & Frachtenberg, E. (Hrsg.), *Proceedings of the 15th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 6253 von *Lecture Notes in Computer Science*, (S. 77–92). Springer.

[45] Fölling, A., Grimme, C., Lepping, J., & Papaspyrou, A. (2011). Connecting Community-Grids by supporting job negotiation with coevolutionary Fuzzy-Systems. *Soft Computing*, *15*(12), 2375–2387.

[46] Fölling, A., Grimme, C., Lepping, J., Papaspyrou, A., & Schwiegelshohn, U. (2009). Competitive Co-evolutionary Learning of Fuzzy Systems for Job Exchange in Computational Grids. *Evolutionary Computation*, *17*(4), 545–560.

[47] Fölling, A. & Hofmann, M. (2012). Improving Scheduling Performance using a Q-Learning-based Leasing Policy for Clouds. In Kaklamanis, C., Papatheodorou, T., & Spirakis, P. (Hrsg.), *Euro-Par 2012 Parallel Processing*, Band 7484 von *Lecture Notes in Computer Science*, (S. 337–349)., Berlin, Heidelberg. Springer.

[48] Fölling, A. & Lepping, J. (2012). Knowledge discovery for scheduling in computational grids. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*, 287–297.

[49] Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley.

[50] Foster, I. (2005). Globus Toolkit Version 4: Software for Service-Oriented Systems. In *Proceedings of the 2005 IFIP international conference on Network and Parallel Computing*, NPC'05, (S. 2–13)., Berlin, Heidelberg. Springer.

[51] Foster, I. & Kesselman, C. (2003). *The Grid: Blueprint for a New Computing Infrastructure* (2. Auflage. Aufl.). Morgan Kaufmann Publishers.

[52] Foster, I. & Kesselmann, C. (1997). Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High performance Computing Applications*, *11*(2), 115–128.

[53] Franke, C., Lepping, J., & Schwiegelshohn, U. (2007). Greedy Scheduling with Complex Objectives. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling*, (S. 113–120). IEEE Press. CD-ROM.

[54] Fuechsle, M., Miwa, J. A., Mahapatra, S., Ryu, H., Lee, S., Warschkow, O., Hollenberg, L. C. L., Klimeck, G., & Simmons, M. Y. (2012). A single-atom transistor. *Nature Nanotechnology*, *7*(4), 242–246.

[55] Garey, M. & Graham, R. (1975). Bounds for Multiprocessor Scheduling with Resource Constraints. *SIAM Journal on Computing*, *4*(2), 187–200.

[56] Garfinkel, T. & Rosenblum, M. (2003). A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proceedings of the Network and Distributed Systems Security Symposium*, (S. 191–206).

[57] Genaud, S. & Gossa, J. (2011). Cost-Wait Trade-Offs in Client-Side Resource Provisioning with Elastic Clouds. In *4th IEEE International Conference on Cloud Computing (CLOUD)*, (S. 1–8). IEEE.

[58] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman.

[59] Grimme, C., Lepping, J., Papaspyrou, A., Wieder, P., Yahyapour, R., Oleksiak, A., Wäldrich, O., & Ziegler, W. (2007). Towards a Standards-Based Grid Scheduling Architecture. CoreGRID Technical Report TR-0123, Institute on Resource Management and Scheduling.

[60] Grimme, C. & Papaspyrou, A. (2009). Cooperative Negotiation and Scheduling of Scientific Workflows in the Collaborative Climate Community Data and Processing Grid. *Future Generation Computer Systems*, *25*(3), 301–307.

[61] Henderson, R. (1995). Job Scheduling Under the Portable Batch System. In D. Feitelson & L. Rudolph (Hrsg.), *Proceedings of the 1st Workshop on Job*

Scheduling Strategies for Parallel Processing, Band 949 von *Lecture Notes in Computer Science* (S. 279–294). Berlin,Heidelberg: Springer.

[62] Hillenbrand, M., Mauch, V., Stoess, J., Miller, K., & Bellosa, F. (2012). Virtual InfiniBand Clusters for HPC Clouds. In *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*.

[63] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

[64] Hotovy, S. (1996). Workload Evolution on the Cornell Theory Center IBM SP2. In *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, Band 1162 von *Lecture Notes in Computer Science*, (S. 27–40). Springer.

[65] Iosup, A., Epema, D. H. J., Tannenbaum, T., Farrellee, M., & Livny, M. (2007). Inter-Operating Grids through Delegated MatchMaking. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, SC '07, New York, NY, USA. ACM.

[66] Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D. (2011). Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems*, *22*, 931–945.

[67] Kandel, A. & Langholz, G. (1993). *Fuzzy Control Systems* (1. Auflage. Aufl.). CRC Press.

[68] Kettimuthu, R., Subramani, V., Srinivasan, S., Gopalsamy, T., Panda, D. K., & Sadayappan, P. (2005). Selective Preemption Strategies for Parallel Job Scheduling. *International Journal on High Performance Computing and Networking*, *3*(2/3), 122–152.

[69] Kost, B. (2003). *Optimierung mit Evolutionsstrategien*. Harri Deutsch Verlag.

[70] Krefting, D., Bart, J., Beronov, K., Dzhimova, O., Falkner, J., Hartung, M., Hoheisel, A., Knoch, T. A., Lingner, T., Mohammed, Y., Peter, K., Rahm, E., Sax, U., Sommerfeld, D., Steinke, T., Tolxdorff, T., Vossberg, M., Viezens, F., & Weisbecker, A. (2009). MediGRID: Towards a user friendly secured grid infrastructure. *Future Generation Computer Systems*, *25*(3), 326–336.

[71] Lagar-Cavilla, H. A., Whitney, J. A., Scannell, A., Patchin, P., Rumble, S. M., de Lara, E., Brudno, M., & Satyanarayanan, M. (2009). SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing. In *Proceedings of the 4th ACM European conference on Computer systems*, (S. 1–12). ACM.

[72] Lakshmi, J. & Nandy, S. (2009). I/O Device Virtualization in the multi-core era, a QoS perspective. In *Proceedings of the 2009 Workshops at the Grid and Pervasive Computing Conference*.

[73] Li, Y., Gujrati, P., Lan, Z., & Sun, X.-h. (2007). Fault-Driven Re-Scheduling For Improving System-level Fault Resilience. In *Proceedings of the 2007 International Conference on Parallel Processing*, (S. 39–46)., Washington, DC, USA. IEEE Computer Society.

[74] Lifka, D. A. (1995). The ANL/IBM SP Scheduling System. In *Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing*, Band 949 von *Lecture Notes in Computer Science*, (S. 295–303). Springer.

[75] Litzkow, M., Livny, M., & Mutka, M. (1988). Condor-A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, (S. 104 –111).

[76] Lo, V., Mache, J., & Windisch, K. (1998). A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling. In Feitelson, D. G. & Rudolph, L. (Hrsg.), *Proceedings of the 4th Workshop on Job Scheduling Strategies for Parallel Processing*, Band 1459 von *Lecture Notes in Computer Science)*, (S. 25–46). Springer.

[77] Lu, K., Subrata, R., & Zomaya, A. (2006). Towards Decentralized Load Balancing in a Computational Grid Environment. In Y.-C. Chung & J. E. Moreira (Hrsg.), *Advances in Grid and Pervasive Computing*, Band 3947 von *Lecture Notes in Computer Science* (S. 466–477). Berlin, Heidelberg: Springer.

[78] Lu, W., Jackson, J., & Barga, R. (2010). AzureBlast: A Case Study of Developing Science Applications on the Cloud. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, (S. 413–420)., New York, NY, USA. ACM.

[79] Lublin, U. & Feitelson, D. G. (2003). The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. *Journal of Parallel and Distributed Computing*, *63*(11), 1105–1122.

[80] Mamdani, E. H. & Assilian, S. (1974). Application of fuzzy algorithms for control of simple dynamic plant. *Proceedings of the IEEE*, *121*(12), 1585–1588.

[81] Mao, M., Li, J., & Humphrey, M. (2010). Cloud Auto-scaling with Deadline and Budget Constraints. In *11th IEEE/ACM International Conference on Grid Computing (GRID)*, (S. 41–48). IEEE.

[82] Marshall, P., Keahey, K., & Freeman, T. (2010). Elastic Site: Using Clouds to Elastically Extend Site Resources. In *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, (S. 43–52)., Washington, DC, USA. IEEE Computer Society.

[83] Marshall, P., Tufo, H., & Keahey, K. (2012). Provisioning Policies for Elastic Computing Environments. In *Proceedings of the 9th High-Performance Grid and Cloud Computing Workshop and the 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Washington, DC, USA. IEEE Computer Society.

[84] Marshall, P., Tufo, H., Keahey, K., LaBissoniere, D., & Woitaszek, M. (2012). Architecting a Large-Scale Elastic Environment: Recontextualization and Adaptive Cloud Services for Scientific Computing. In *Proceedings of the 7th International Conferenc on Software Paradigm Trends (ICSOFT)*, Communications in Computer and Information Science, (S. 409–418)., Berlin, Heidelberg. Springer.

[85] Mell, P. & Grance, T. (2011). The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, Zugriff: Juli 2012.

[86] Miettinen, K. (1999). *Nonlinear Multiobjective Optimization.* Dissertation University of Jyväskylä.

[87] Morgan, T. P. (2011). Amazon gooses HPC cloud with Xeon E5s. http://www.theregister.co.uk/2011/11/16/amazon_aws_hpc_xeon_e5/, Zugriff: 16. Januar 2013.

[88] Motwani, R., Phillips, S., & Torng, E. (1994). Non-Clairvoyant Scheduling. *Theoretical Computer Science*, *130*(1), 17–47.

[89] Nandagopal, M., Gokulnath, K., & Uthariaraj, V. R. (2010). Sender initiated decentralized dynamic load balancing for multi cluster computational grid environment. In *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*, A2CWiC'10, New York, NY, USA. ACM.

[90] Naroska, E. & Schwiegelshohn, U. (2002). On an On-line Scheduling Problem with Parallel Jobs. *Information Processing Letters*, *81*, 297–304.

[91] Neiger, G., Santoni, A., Leung, F., Rodgers, D., & Uhlig, R. (2006). Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. *Intel Technology Journal*, *10*(3), 167–178.

[92] Nguyen, T. D., Vaswani, R., & Zahorjan, J. (1996). Parallel Application Characterization for Multiprocessor Scheduling Policy Design. In *Job Scheduling Strategies for Parallel Processing*, Band 1162 von *Lecture Notes in Computer Science*, (S. 175–199). Springer.

[93] Nishimura, H., Maruyama, N., & Matsuoka, S. (2007). Virtual Clusters on the Fly—Fast, Scalable, and Flexible Installation. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid07)*, (S. 549–556)., Washington, DC, USA. IEEE Computer Society.

[94] Nussbaum, L., Anhalt, F., Mornard, O., & Gelas, J.-P. (2009). Linux-based virtualization for HPC clusters. In *Proceedings of the Montreal Linux Symposium*. Online. http://www.linuxsymposium.org/2009/ls-2009-proceedings.pdf, Zugriff: 7. August 2012.

[95] Omer, K., Maljevic, I., Anthony, R., Petridis, M., Parrott, K., & Schulz, M. (2009). Dynamic Scheduling of Virtual Machines Running HPC Workloads in Scientific Grids. In *IEEE International Conference of New Technologies, Mobility and Security*.

[96] Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D. (2010). A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing. In *Cloud Computing*, Band 34 von *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* Kapitel 9, (S. 115–131). Berlin, Heidelberg: Springer.

[97] Ostermann, S., Prodan, R., & Fahringer, T. (2009). Extending Grids with Cloud Resource Management for Scientific Computing. In *10th IEEE/ACM International Conference on Grid Computing*, (S. 42–49).

[98] Paredis, J. (2000). Coevolutionary algorithms. In T. Bäck, D. B. Fogel, & Z. Michalewicz (Hrsg.), *Evolutionary Computation 2: Advanced Algorithms and Operators* (S. 224–238). Institute of Physics Publishing, Bristol.

[99] Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems* (4. Aufl.). Springer.

[100] Potter, M. A. & De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, *8*(1), 1–29.

[101] Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzbook.

[102] Rechenberg, I. (1994). *Evolutionsstrategie '94*. Stuttgart: Frommann Holzboog.

[103] Rehr, J. J., Vila, F. D., Gardner, J. P., Svec, L., & Prange, M. (2010). Scientific Computing in the Cloud. *Computing in Science and Engineering*, *12*, 34–43.

[104] Ries, C. B. (2012). *BOINC: Hochleistungsrechnen mit Berkeley Open Infrastructure for Network Computing*. Springer.

[105] Saravanakumar, E. & Prathima, G. (2010). A Novel Load Balancing Algorithm for Computational Grid. In *2010 International Conference on Innovative Computing Technologies (ICICT)*, (S. 1–6).

[106] Schley, L. (2008). Prospects of Co-Allocation Strategies for a Lightweight Middleware in Grid Computing. In Gonzalez, T. F. (Hrsg.), *Proceedings of the 20th International Conference on Parallel and Distributed Computing and Systems*, (S. 198–205). ACTA Press.

[107] Schroeder, B. & Gibson, G. (2010). A Large-Scale Study of Failures in High-Performance Computing Systems. *IEEE Transactions on Dependable and Secure Computing*, *7*(4), 337 –350.

[108] Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung.* Dissertation Technische Universität Berlin, Fachbereich Verfahrenstechnik.

[109] Schwefel, H.-P. (1995). *Evolution and Optimum Seeking.* John Wiley & Sons, New York.

[110] Schwiegelshohn, U. (2009). An Owner-centric Metric for the Evaluation of Online Job Schedules. In Blazewicz, J., Drozdowski, M., Kendall, G., & McCollum, B. (Hrsg.), *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009)*, (S. 557–569)., Dublin, Ireland.

[111] Schwiegelshohn, U. & Yahyapour, R. (1998). Improving First-Come-First-Serve Job Scheduling by Gang Scheduling. In *IPPS'98 Workshop: Job Scheduling Strategies for Parallel Processing*, Band 1459 von *Lecture Notes in Computer Science (LNCS)*, (S. 180–198). Springer.

[112] Schwiegelshohn, U. & Yahyapour, R. (2000). Fairness in Parallel Job Scheduling. *Journal of Scheduling*, *3*(5), 297–320.

[113] Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., & Hanrahan, P. (2008). Larrabee: A Many-Core x86 Architecture for Visual Computing. In *Proceedings of ACM SIGGRAPH 2008*, New York, NY, USA. ACM.

[114] Senkul, P. & Toroslu, I. H. (2005). An architecture for workflow scheduling under resource allocation constraints. *Information Systems*, *30*(5), 399–422.

[115] Skovira, J., Chan, W., Zhou, H., & Lifka, D. A. (1996). The EASY - LoadLeveler API Project. In *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, Band 1162 von *Lecture Notes in Computer Science*, (S. 41–47)., London, UK. Springer.

[116] Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*. Dissertation Department of Computer Science, University of Pittsburgh.

[117] Song, H. J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K., & Chien, A. (2000). The MicroGrid: A Scientific Tool for Modeling Computational Grids. *Scientific Programming*, *8*(3), 127–141.

[118] Streit, A., Bala, P., Beck-Ratzka, A., Benedyczak, K., Bergmann, S., Breu, R., et al. (2010). UNICORE 6 - Recent and Future Advancements. *annals of telecommunications*, *65*, 757–762.

[119] Suri, P. & Singh, M. (2010). An Efficient Decentralized Load Balancing Algorithm For Grid. In *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, (S. 10–13).

[120] Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning - An Introduction* (4. Aufl.). The MIT Press.

[121] Suzaki, K. & Walsh, D. (1998). Implementing the Combination of Time Sharing and Space Sharing on AP/Linux. In Feitelson, D. G. & Rudolph, L. (Hrsg.), *Job Scheduling Strategies for Parallel Processing*, Band 1459 von *Lecture Notes in Computer Science*, (S. 83–97)., Berlin, Heidelberg. Springer.

[122] Takagi, T. & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on System, Man, and Cybernetics*, *SMC-15*(1), 116–132.

[123] Talby, D. & Feitelson, D. G. (2005). Improving and Stabilizing Parallel Computer Performance Using Adaptive Backfilling. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*.

[124] Toosi, A., Calheiros, R., Thulasiram, R., & Buyya, R. (2011). Resource Provisioning Policies to Increase IaaS Provider's Profit in a Federated Cloud Environment. In *13th International Conference on High Performance Computing and Communications (HPCC)*, (S. 279 –287).

[125] Tsafrir, D., Etsion, Y., & Feitelson, D. (2007). Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE Transactions on Parallel and Distributed Systems*, *18*(6), 789 –803.

[126] Watkins, C. J. C. H. W. (1989). *Learning from Delayed Rewards*. Dissertation King's College.

[127] Xhafa, F., Carretero, J., & Abraham, A. (2007). Genetic algorithm based schedulers for grid computing systems. *International Journal of Innovative Computing, Information and Control*, *3*(6), 1053–1071.

[128] Yahyapour, R. (2002). *Design and Evaluation of Job Scheduling Strategies for Grid Computing*. Dissertation Universität Dortmund.

[129] Youseff, L., Wolski, R., Gorda, B., & Krintz, C. (2006). Paravirtualization for HPC systems. In *Proceedings of the 2006 international conference on Frontiers of High Performance Computing and Networking*, ISPA'06, (S. 474–486)., Berlin, Heidelberg. Springer.

[130] Yu, J., Buyya, R., & Ramamohanarao, K. (2008). Workflow Scheduling Algorithms for Grid Computing. In *Metaheuristics for Scheduling in Distributed Computing Environments*, Band 146 von *Studies in Computational Intelligence* (S. 173–214). Berlin, Heidelberg: Springer.

[131] Yu, J., Buyya, R., & Tham, C. K. (2005). Cost-based Scheduling of Scientific Workflow Applications on Utility Grids. In *Proceedings of the First International Conference on e-Science and Grid Computing*, (S. 140–147).

[132] Yu, Z., Wang, C., & Shi, W. (2010). Failure-aware workflow scheduling in cluster environments. *Cluster Computing*, *13*(4), 421–434.

[133] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.

[134] Zeng, X. & Sodan, A. (2009). Job Scheduling with Lookahead Group Matchmaking for Time/Space Sharing on Multi-core Parallel Machines. In E. Frachtenberg & U. Schwiegelshohn (Hrsg.), *Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science (S. 232–258). Berlin, Heidelberg: Springer.

[135] Zhang, Y., Squillante, M. S., Sivasubramaniam, A., & Sahoo, R. K. (2005). Performance Implications of Failures in Large-Scale Cluster Scheduling. In *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, (S. 233–252)., Berlin, Heidelberg. Springer.

[136] Zheng, g., Kalkulapati, G., & Kalé, L. V. (2004). BigSim: A Parallel Simulator for Performance Prediction of Extremely Large Parallel Machines. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*.

[137] Zhou, S., Zheng, X., Wang, J., & Delisle, P. (1993). Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems. *Software: Practice and Experience*, *23*(12), 1305–1336.