Universität Dortmund
Fachbereich Informatik

Endbericht

Peerbasierte Computational Grids

Endbericht der
Projektgruppe 505
August 2006 - September 2007

Teilnehmer:
Stephan Brabender, Robin Fink, Alexander Fölling,
Sebastian Kaps, Andreas Kohne, Christian Kusber,
Dietmar Ogai, Paul Staszkiewicz, Oleg Tolmatcev,
Nurettin Ülger, Thomas Voß und Bjarke Wiggerich

Prof. Dr.-Ing. Uwe Schwiegelshohn,
Institut für Roboterforschung,
Abteilung Informationstechnik

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Dieser Bericht der Projektgruppe 505 „Peerbasierte Computational Grids“ des Instituts für Roboterforschung an der Universität Dortmund im Wintersemester 2006/2007 und Sommersemester 2007 stellt die Arbeit der Projektgruppe sowie die erzielten Ergebnisse vor.

In den vergangenen Jahren sind die Anforderungen an Rechenleistung und Speicherplatz stark gestiegen. Obwohl die Ressourcen und Informationen zur Verfügung stehen, sind sie geographisch weit verteilt bzw. nicht nutzbar. Grid-Computing ist ein Versuch, dieses Problem zu lösen.
Entstanden ist Grid-Computing am Ende der 1990er Jahre durch die Kopplung von Hochleistungsrechnern über schnelle Datenleitungen, um damit speicher- oder rechenintensive Aufgaben lösen zu können (s. Foster u. Kesselman, 1999).
Ein Grid ist eine netzwerkbasierte Infrastruktur, die eine gemeinschaftliche Verwendung der Ressourcen aller teilnehmenden Systeme ermöglicht. Der Begriff des Grids wurde von der Idee des Stromnetzes (engl. Powergrid) und der damit verbundenen einfachen Nutzung von Ressourcen abgeleitet. Beim Stromnetz wird die Energie aus der Steckdose für verschiedenste Zwecke genutzt, ohne Kenntnis seiner Herkunft oder Erzeugung. Welcher stromerzeugende Teilnehmer für die Bereitstellung der Ressource Strom verantwortlich ist, ist für den Verbraucher unerheblich (s. Chetty u. Buyya, 2002).
Übertragen auf ein Computational Grid kann als Ziel eines solchen die gemeinsame Nutzung von Rechenleistung und Speicherplatz, unabhängig von deren geographischen Position, definiert werden. Jede ungenutzte Ressource steht potentiell allen Grid-Teilnehmern zur Verfügung. Dabei wird die Leistung der einzelnen Systeme zu einem „Supercomputer“ zusammengeschaltet. Um dies global zu ermöglichen, werden einheitliche Standards benötigt. Es ist jedoch zu berücksichtigen, dass die einzelnen Ressourcenanbieter unterschiedliche Prioritäten bezüglich der Nutzung ihrer Ressourcen haben können. Für den Betrieb eines Grids sind daher Aspekte der Ressourcenverwaltung und die Verteilung von (Rechen-)Jobs relevant.
Für den Aufbau eines sochen Grid-Systems sind dezentrale Ansätze vorzuziehen, um den Teilnehmern ein möglichst großes Maß an Autonomie zu belassen. Bislang sind für dezentrale Jobaustauschstrategien im Grid-Kontext nur wenige Resultate und Ansätze veröffentlicht worden. Es gibt zwar verschiedene Ansätze und Umsetzungen von Middlewaresystemen zum Betrieb eines Grid, allerdings läuft der Scheduling-Prozess dort oft zentral ab. Gleichzeitig wird davon ausgegangen, dass nahezu alle Informationen über alle Grid-Teilnehmer global abrufbar sind (siehe Kapitel 2.1). Ein weiterer Grund für fehlende Resultate ist die Komplexität eines solchen Grid-Systems. Diese verhindert eine rein theoretische Betrachtung und legt Untersuchungen mit realen Trace-Daten

nahe. Auf Grund dessen und auf Grund des Fehlens eines geeigneten Realsystems wird ein Simulationsframework benötigt, welches eine Laufzeitumgebung für den simulierten Betrieb einer Grid-Umgebung bereithält. Die Simulation ermöglicht es, unterschiedliche Scheduling- und Jobaustausch-Strategien in verschiedenen Grid-Umgebungen und -Konstellationen experimentell zu testen und zu analysieren. Insbesondere die Austausch-Strategien ermöglichen es dabei, das System an die Präferenzen des Betreibers anzupassen und dadurch Vorteile aus dem Jobaustausch zu ziehen.

Das Ziel der Projektgruppe lässt sich in vier Aufgaben unterteilen:

- Entwurf einer einfachen Grid-Umgebung
- Schaffung von Services für den Austausch von Jobs
- Entwicklung eines Simulationsframeworks für Grid-Systeme
- Evaluation des Verhaltens peerbasierter Computational-Grids

Als Grundlage des Projekts war zunächst eine einfache Grid-Umgebung zu entwerfen. Diese Grid-Umgebung sollte aus verschiedenen Sites, die hier vereinfachend als Parallelrechner mit einer spezifizierbaren Anzahl von Prozessoren angenommen werden, bestehen. Die einzelnen Sites bedienen ihre lokalen Benutzer. Dazu ist je Site ein Scheduler für das Ressourcenmanagement vorgesehen. Bei der Implementierung ist eine leichte Austauschbarkeit der lokalen Scheduling-Strategien durch den Einsatz geeigneter Schnittstellen berücksichtigt worden.
Damit die verschiedenen Sites mit den jeweils anderen Teilnehmern der Grid-Umgebung Rechenjobs austauschen können, sind zusätzliche Dienste für die Weitergabe von Jobs erforderlich. Des Weiteren ist je Site statt eines zentralen übergeordneten „Grid-Schedulers“ eine zusätzlichen Entscheidungsstufe notwendig, die über die lokale oder externe Ausführung von lokal eingereichten Jobs entscheidet. Diese Entscheidung wird basierend auf einer Anfrage, ob der Job auf einem anderen System ausgeführt werden kann, an das bekannte Grid-Umfeld getroffen. Das bekannte Grid-Umfeld besteht aus Sites, welche über P2P-Mechanismen gefunden werden. Für externe Jobs muss dieser Mechanismus entscheiden, ob sie angenommen oder ob sie abgelehnt werden. Die Entscheidung zum Austausch eines Rechenjobs wird dabei von jedem System lokal getroffen. Durch diese zusätzliche Entscheidungsstufe muss das lokale Scheduling-System nicht modifiziert werden. Analog zum lokalen Scheduling-System ist auch hier eine Schnittstelle zum Einsatz unterschiedlicher Entscheiderkonzepte, welche Erweiterungsmöglichkeiten für komplexere Modelle bieten, vorgesehen.
Für die entworfene Grid-Architektur ist ein geeignetes Simulationsframework entworfen worden. Dabei sollte eine mögliche Trennung vom eigentlichen Grid und der Simulationskomponente für einen Betrieb der entworfenen Grid-Umgebung in einem realen Kontext berücksichtigt werden.
Im Abschluss wird mit der Hilfe dieses Frameworks das Verhalten peerbasierter Computational Grids evaluiert. Dazu sind verschiedene Scheduling- und insbesondere

Entscheidungsstrategien entworfen und implementiert worden. Die dazu erforderlichen Simulationen sind mit bereits bestehenden Traces von realen Sites[1] durchgeführt worden. Anhand der im Simulationsbetrieb gewonnenen Daten konnten unter Berücksichtigung der verwendeten Strategien Aussagen über die Leistung und die Stabilität des Grid getroffen werden.

Zu Beginn des Berichtes wird in Kapitel 2 die Motivation für diese Projektgruppe anhand des aktuellen Wissensstand herausgearbeitet. Dazu werden bereits existierende Grid-Umgebungen (Kapitel 2.1), Simulationsumgebungen (Kapitel 2.2) und bereits bestehende Resultate (Kapitel 2.3) betrachtet.
Darauf folgt in Kapitel 3 eine Erörterung der theoretischen Grundlagen der Arbeit. Dazu zählt eine kurze Einführung in Peer-to-Peer-Systeme (Kapitel 3.1) und Scheduling-Theorie (Kapitel 3.2).
Im Anschluss an die Grundlagen wird in Kapitel 4 der Entwurf und der Aufbau der Simulationsumgebung anhand ihrer Komponenten vorgestellt, wobei ebenfalls auf einige grundlegende Entscheidungen eingegangen werden soll, die im Rahmen der Modell-Annahmen zu treffen waren.
Hierauf folgt die Vorstellung der verschiedenen Entscheidungsstrategien in Kapitel 5. Es werden die verschiedenen Strategien beschrieben, nach denen die Open Decision Interface Node (ODIN)-Komponente einer einzelne Site über die Annahme oder Ablehnung von Jobs anderer Grid-Teilnehmer, sowie über die lokale oder externe Berechnung lokal eingereichter Jobs entscheidet.
Das Kapitel 6 beschäftigt sich mit der Evaluationsphase des Projektes. Es werden die verwendeten Metriken (Kapitel 6.2) definiert und der beispielhafte Ablauf eines Experiments (Kapitel 6.3) dargestellt.
Es folgt Kapitel 7, in dem die durchgeführten Experimente und die daraus gewonnenen Ergebnisse dokumentiert werden. Abschließend folgt die Schlussfolgerung (Kapitel 8), in der die gewonnenen Erkenntnisse in einem größeren Kontext interpretiert werden und in der ein Ausblick auf weitere mögliche Forschungsansätze gegeben wird.
In den Anhängen werden die Installation des Frameworks, die grafische Oberfläche zur Konfiguration der Simulationen (Kapitel A.2) und die zur Auswertung der Daten verwendeten Werkzeuge (Kapitel A.3) vorgestellt. Im Abschnitt A.4 finden sich auch detaillierte Tabellen und Matrizen mit den ausgewerteten Daten der durchgeführten Experimente.

[1] Dror Feitelson, Parallel Workloads Archive, `http://www.cs.huji.ac.il/labs/parallel/workload/` (08/2007)

2 Hintergrund

Inhalt

Das folgende Kapitel gibt einen Überblick über den Hintergrund der Ziele der Projektgruppe. Es werden verschiedene Grid-Systeme vorgestellt und deren Stärken und Schwächen aufgezeigt. Weiterhin werden existierende Simulationsumgebungen für Grids vorgestellt, sowie Ergebnisse anderer Untersuchungen zum Thema Jobaustausch in Grid-Systemen.

2.1 Grid-Umgebungen

Im Folgenden werden einige bekannte Grid-Umgebungen vorgestellt, welche auch in realen Betrieb erprobt sind und weite Verbreitung finden. Des Weiteren werden ihre Vor- und Nachteile kurz vorgestellt.

Globus

Das Globus Projekt wurde im Jahr 1995 durch die Defense Advanced Research Projects Agency (DARPA) in Amerika gegründet (vgl. Garritano, 2003). Im Rahmen dieses Projektes wurde das Globus Toolkit entwickelt. Das Globus Toolkit hat sich als der quasi Standard für Grid-Middleware herausgestellt und wird von vielen multi-nationalen Projekten zur Erstellung von Grids eingesetzt.
Die aktuellste Version des Globus Toolkit ist die Version 4. Das Globus Toolkit 4 (GT4) wurde im April 2005 veröffentlicht und als Service Oriented Architecture (SOA) implementiert (vgl. Foster, 2005). Dies bedeutet, dass Globus Dienste zur Verfügung stellt, welche von beliebigen Nutzern in Anspruch genommen werden können, die Teil des Grids sind. Da die Kommunikation über das Internet abgewickelt werden sollte, wurde Globus komplett erneuert und arbeitet jetzt als Web-Service.
Das GT4 sammelt an einer zentralen Stelle Informationen über alle Ressourcen, welche im Grid zur Verfügung stehen. Wenn ein Job an einer fremden Site ausgeführt werden soll, wird der Job und seine Ressourcenanforderungen mit einer speziellen Beschreibungssprache beschrieben. Diese Beschreibung wird dann mit den Eigenschaften der zur Verfügung stehenden Sites abgeglichen und an eine Site, welche genug Kapazitäten besitzt, verschickt.

Der ankommende Job wird von einem Gatekeeper entgegengenommen und so umgewandelt, dass der lokale Scheduler den Job in das Schedule mit einbeziehen kann.
Globus stellt selbst keinen Scheduler zur Verfügung, sondern bietet nur ein Rahmenwerk zur Erstellung und Verwaltung von Grids. Somit bleibt das lokale System unangetastet von Globus. Dies ist zwar sehr positiv, aber es lassen sich auch keine Aussagen über das Schedule-Verhalten der einzelnen Sites oder des gesamten Systems machen, da die verschiedenen Sites zwar über Globus miteinander verbunden sind, aber sonst keine weiteren Informationen austauschen oder preisgeben.

Legion

Legion ist ein objektorientiertes Metasystem-Software-Projekt, welches an der Universität Virginia entwickelt wurde (vgl. Grimshaw u. Wulf, 1996). In Legion sind alle Ressourcen wie Nutzer, Daten und Anwendungen Objekte festgelegter Klassen. Diese Objekte können in einer beliebigen objektorientierten Programmiersprache beschrieben sein. Es gibt spezielle Wrapper, die alle Objekte weltweit mit Legion bekannt machen und einheitlich allen Nutzern zur Verfügung stellen können. Legion arbeitet als Middleware, welche auf dem lokalen Betriebssystem aufsetzt und dieses nahtlos in die Gridumgebung einpasst.
Alle im System verfügbaren Ressourcen und Objekte werden zentral in einer so genannten Collection verwaltet. Über das Legion Grid Portal, welches über einen Browser zu erreichen ist, kann ein Benutzer sich einen Überblick über das Grid und alle zur Verfügung stehenden Ressourcen verschaffen und Objekte jeder Art erzeugen oder neue Jobs im Grid starten (vgl. Natrajan et al., 2001).
Wenn ein neuer Job erstellt wird, ermittelt der Ressourcen Manager welche Objekte benötigt werden, um den Job berechnen zu lassen. Danach ermittelt er über die Collection, ob die benötigten Ressourcen zur Zeit zur Verfügung stehen. Dies geschieht über ein simples Pattern-Matching. Dann ermittelt der Scheduler eine Zuordnung der Tasks auf die entsprechenden Objekte. Der Scheduler kann zusätzlich zu diesem Haupt-Schedule noch mehrere Neben-Schedules ermitteln und diese dem Enactor schicken. Der Enactor versucht dann die entsprechenden Ressourcen zu reservieren. War dies erfolgreich, so ruft der Enactor die entsprechenden Methoden auf den Objekten auf, um den Job auszuführen (vgl. Chapin et al., 1999b).
Ein großer Vorteil von Legion ist die zentrale Übersicht über alle Objekte des Systems und deren Eigenschaften. Ein Nachteil ist aber, dass der lokale Scheduler bestimmt, welche Ressourcen genutzt werden, um einen Job auszuführen; auch wenn diese Ressourcen gar nicht in seinen Verwaltungsbereich fallen. Somit versucht der lokale Scheduler fremde Ressourcen zu belegen. Ist dies nicht möglich, ist der Enactor gezwungen ein alternatives Schedule durchzusetzen, oder einen Fehler zu melden.

Condor

Condor ist ein Middlewaresystem, welches speziell für das Management rechenintensiver Jobs an der Universität Wisconsin Madison entwickelt wurde (vgl. Thain et al.,

2004). Es beinhaltet Module für Job-queueing, Schedulingpolitiken, Prioritätsrichtlinien, Ressourcen-Monitoring und Management.
Condor kann grob in zwei Komponenten eingeteilt werden. Da ist auf der einen Seite der Job-Manager, welcher die aktuelle Jobqueue anzeigen kann, und dem auch neue Jobs hinzugefügt werden können. Auf der anderen Seite gibt es den Ressourcen-Manager. Dieser gibt Auskunft über erreichbare Maschinen und deren Kapazitäten. Dies geschieht über das in Condor integrierte ClassAd. ClassAd ist eine Beschreibungssprache, mit deren Hilfe es möglich ist, Maschinen und deren Leistungen genau zu beschreiben. Es handelt sich hierbei um eine Java-API. Mit ihrer Hilfe lassen sich so genannte Property Maps erstellen, in denen der Job und alle Ressourcen durch eine Vielzahl von Parametern beschrieben werden. So kann z.B. ein lokaler PC mit seinem Betriebssystem, seiner Festplattengröße, seiner CPU-Leistung und seinem Hauptspeicher dargestellt werden. Genauso wird nun ein zu bearbeitender Job mit Hilfe von ClassAd beschrieben. Hierbei werden aber die Anforderungen an die entfernte Maschine angeführt. Dann vergleicht Condor die Jobbeschreibung mit allen ihm zur Verfügung stehenden Maschinenbeschreibungen und startet, nachdem eine passende Maschine gefunden wurde, den Job dort.

Condor G

Condor G ist ein Job-Manager für Condor (vgl. Frey et al., 2001). Es stellt so zu sagen ein Fenster zu einem größeren Gridkontext dar. Und zwar ist es mit Condor G möglich, auf Ressourcen, welche über Globus verwaltet werden, zuzugreifen. Somit bietet es eine Oberfläche, aus welcher heraus Jobs nicht nur auf anderen Condor-Maschinen ausgeführt werden können, sondern eröffnet die gesamte Kapazität eines großen Globus-Grids. Condor G kombiniert das Interdomänen-Ressourcenmanagement vom Globus Toolkit mit dem Intradomänen- Jobmanagement von Condor.
Condor G nutzt Globus, um Condor Zugang zu Grid-Ressourcen zu gewähren und Daten an entfernte Maschinen zu übertragen. Weiterhin werden Programme aus der Ferne gestartet und überwacht. Zusätzlich werden noch die Sicherheitsmechanismen zur Authentifikation und Absicherung von Datenübertragungen aus Globus implementiert.

UNICORE

Die Entwicklung von UNICORE (Uniform Interface to Computing Resources, `http://www.unicore.eu/`) begann 1997 am Forschungszentrum Jülich mit finanzieller Unterstützung des Bundesministeriums für Bildung und Forschung (BMBF), mittlerweile liegt es in der Version 6 vor. Bei der Ressourcenverwaltung haben Site-Betreiber die Möglichkeit, in feinen Abstufungen festzulegen, welche Ressourcen sie dem Grid zur Verfügung stellen möchten. Lokale Richtlinien werden dabei automatisch berücksichtigt.
Das System baut auf einer so genannten Three-Tier-Architektur auf und besteht aus einer User- und einer Serverkomponente, sowie einem Batch-Subsystem (Erwin u. Snelling, 2001).
Die Serverkomponente besteht aus zwei Teilen, einem Webserver und dem Network Job Supervisor (NJS). Der Webserver stellt das Graphical User Interface (GUI) und die

Benutzer-Authentifizierung bereit. Der NJS teilt die von einem Benutzer eingereichten Jobs auf die Zielsysteme auf, reicht sie dort ein und kontrolliert ihre Ausführung. Das Batch-Subsystem bildet die dritte zentrale Komponente von UNICORE und umfaßt die eigentlichen Zielsysteme. Jede UNICORE-Site kann dabei noch weiter in so genannte VSites unterteilt werden, wobei jede VSite von einem Anwender als Zielsystem für einen (Teil-)Job ausgewählt werden kann. Scheduling innerhalb von UNICORE erfolgt weitgehend manuell. Neuere Arbeiten (vgl. Streit et al., 2005) beschreiben einen Meta-Scheduler als Aufsatz für UNICORE, der mit Hilfe von Reservierungs-Mechanismen eine globale Scheduling-Instanz realisiert. Besondere Beachtung verdient auch die Sicherheit bei UNICORE. So wird die Kommunikation zwischen verschiedenen Systemen verschlüsselt und jeder Benutzer wird vom System eindeutig über ein digitales Zertifikat identifiziert (vgl. Romberg, 2002).

Die hier vorgestellten Grid-Systeme wurden als Grundlage für den Entwurf des zu erstellenden Systems verwendet. Es mussten aber auch viele Teile neu entwickelt bzw. angepasst werden, da keine zentrale Informationsquelle für die Ressourcen existieren sollte und die gesamte Kommunikation mit P2P-Mechanismen realisiert werden sollte. Zusätzlich sollte der Scheduling-Prozess vollkommen dezentral ablaufen.Das für diese Zwecke neuentwickelte Grid-Modell wird unter Abschnitt 4 ausführlich beschrieben.

Es existieren bereits Simulationsumgebungen für Grids, bzw. allgemeine Simulationsumgebungen für Netzwerksimulation. Einige weit verbreitete Ansätze werden vorgestellt und beurteilt:

2.2 Simulationsumgebungen für Grid-Systeme

Im Bereich der Netzwerksimulatoren kann zwischen Netzwerksimulatoren, die speziell für Gridumgebungen entwickelt wurden, und allgemeinen Netzwerksimulatoren unterschieden werden. Allgemeinen Netzwerksimulatoren wie NS-2 (Breslau et al., 2000) oder OMNeT++ (Varga, 2001) simulieren Paket-basiert und werden z.B. benutzt um Performance-Evaluierungen von speziellen TCP-Implementierungen durchzuführen (Fall u. Floyd, 1996).
Allgemeine Netzwerksimulatoren stellen zur Untersuchung von speziellen verteilten Systemen wie Grids eine zu geringe Abstraktionsstufe dar und können Grid-spezifische Fragestellungen nicht angemessen untersuchen. Zur Simulation von Grids existieren spezialisiertere Ansätze:
Die von Buyya u. Murshed (2002) beschriebene Simulationsumgebung GridSim wurde am Grid Computing and Distributed Systems (GRIDS) Laboratory der Universität Melbourne entwickelt. Ein wesentliches Designziel des Projekts ist es, eine umfangreiche Simulationsumgebung bereitzustellen, die eine Evaluation von unterschiedlichen, auf Markt-orientierten Scheduling-Ansätzen basierenden Grids ermöglicht. GridSim grenzt sich von Ansätzen wie Bricks (vgl. Takefusa et al., 1999) ab, die eine globale Scheduling-Komponente voraussetzen, die Zugriff auf Ressourceninformationen (z.B. Auslastung)

sämtlicher teilnehmender Computer hat. Innerhalb von GridSim kann ein Grid-Nutzer über einen Ressource-Broker einen geeigneten (Geschäfts-)Partner finden lassen, der unter Beachtung von Rahmenbedingungen (z.B. Deadline- / Kostenspezifikation) den Job des Kunden ausführt. Zwar erfolgt das Scheduling dezentral, setzt allerdings voraus, dass mit dem *GridInformationService* ein Dienst bereitsteht, der eine aktuelle Liste aller globalen Grid-Teilnehmer führt. Auf Grund des zentralisierten Informationsdienstes ist es mit GridSim ohne erheblichen Aufwand nicht möglich, einen vollkommen dezentralen, P2P-basierten Ansatz zu untersuchen.
SimGrid ist ein Ansatz, der gemäß Casanova (2001) der Idee von GridSim ähnelt. Er benutzt zur Kommunikation eine globale Datenstruktur und setzt somit eine umfangreiche Kenntnis über die Existenz anderer Grid-Teilnehmer voraus.

2.3 Bereits bestehende Resultate

Eines der wenigen theoretischen Resultate (vgl. Schwiegelshohn, 2004) zeigt, dass die Qualität von zwei Schedules verbessert werden kann, wenn während der Erstellung der Schedules Jobaustausch möglich ist.
In einer ersten Untersuchung von Hamscher et al. (2000) konnte mit trace-getriebener Simulation gezeigt werden, dass Vorteile aus unterschiedlichen Jobaustauschstrategien resultieren. Ernemann et al. (2002b,a) konnten zeigen, dass die Verwendung zentralisierter Scheduling-Strukturen zu einer starken Verbesserung der Average Weighted Response Time (AWRT) im Kontext mehrerer Multi-Processor-Sites (MPS) führt.
Subramani et al. (2002) beschreiben einen verteilten Scheduling-Ansatz, der ohne zentralen Scheduler auskommt. Dieser Ansatz ähnelt strukturell dem eigenen, in Kapitel 4 beschriebenen, Ansatz und soll näher betrachtet werden:
Subramani et al. ordnen ihren Ansatz als eine Erweiterung zum verteilten Meta-Scheduling ein. Beim verteilten Meta-Scheduling besteht eine Multi-Processor-Site (MPS) aus einem Meta-Scheduler und einem lokalen Scheduler. Jobs werden lokal zunächst bei dem Meta-Scheduler eingereicht. Dem Meta-Scheduler sind alle im Grid befindlichen Sites bekannt und er überprüft, ob die lokale Auslastung höher ist als die Auslastung einer anderen Site. Ist dies der Fall, schickt der Meta-Scheduler den eingereichten Job unmittelbar an den lokalen Scheduler der Site mit geringerer Auslastung, auf der der Job dann gerechnet wird. Subramani et al. (2002) erweitern diesen Ansatz dahingehend, dass ein Job an mehrere Sites mit geringerer Auslastung geschickt werden kann. Die Site, die den Job zuerst rechnet, muss die anderen Sites über den Start des Jobs benachrichtigen, so dass diese den Job aus ihrer Queue entfernen können. Da K die Obergrenze von Sites darstellt, an die Jobs zur potentiellen Berechnung geschickt werden können, heißt der Ansatz „K-Distributed“.
Der beschriebene Ansatz weißt einige Probleme auf, die, wie später beschrieben wird, mit der vorliegenden Arbeit vermieden wurden:
Im K-Distributed-Ansatz benötigen alle Grid-Teilnehmer weitreichende Informationen über die anderen Sites. So ist es erforderlich, dass sich alle Sites untereinander kennen und permanent Informationen über ihre Auslastung austauschen. Problematisch ist auch,

dass Betreiber von MPS anderen Grid-Teilnehmern keinen direkten Zugriff auf den lokalen Scheduler erteilen werden. Von genau der Annahme geht der K-Distributed-Ansatz allerdings aus. Kritisch ist auch die Voraussetzung, dass bei K Sites eine Site, die einen Job als erste beginnt zu rechnen, die anderen $K-1$ Sites *atomar* darüber informieren muss. Befinden sich im Grid aktuell einige Sites mit geringer Auslastung, wird ein Job mehrfach verschickt und erzeugt einen großen Datenoverhead, da komplette Jobs und keine Jobbeschreibungen verschickt werden.
Die Plausibilität der Evaluation des K-Distributed-Ansatzes ist auch fraglich: Subramani et al. (2002) versuchen simulativ mit Hilfe von realen Tracedaten die Qualität ihres Ansatzes zu belegen. Dabei wird aus dem Feitelson Workload Archive nur ein Trace (CTC) benutzt. Der Trace wird in Stücke von 5000 Jobs zerteilt, die dann für die unterschiedlichen Sites genutzt werden. Die Vorgehensweise, für alle untersuchten Sites den selben Trace zu Grunde zu legen und sich auf nur 5000 Jobs zu beschränken ist praxisfern. Um eine umfangreiche Evaluation durchzuführen, ist es erforderlich, unterschiedliche Sites mit unterschiedlichen Trace-Daten zu betreiben und realistische Zeiträume von mehreren Monaten zu evaluieren.

3 Theoretische Grundlagen

Inhalt

Im Folgenden werden einige theoretische Grundlagen beschrieben, die das Verständnis von Sachverhalten im Gridkontext besser verdeutlichen sollen. Hierbei wird erst auf P2P-Systeme eingegangen. Dabei werden die Eigenschaften von P2P-Systemen sowie unterschiedliche P2P-Systeme untersucht. Des Weiteren wird das Scheduling-Problem auf parallelen Maschinen beschrieben und anschließend einige Scheduling-Algorithmen vorgestellt.

3.1 Peer-to-Peer

Dieser Abschnitt gibt eine kurze Einführung über Peer-to-Peer-Systeme (P2P-Systeme), ihre grundlegenden Eigenschaften, zentrale Aspekte und erläutert wichtige Schlüsselbegriffe in diesem Zusammenhang. Anschließend werden drei unterschiedliche P2P-Systeme mit ihren spezifischen Charakteristika, sowie Vor- und Nachteilen vorgestellt. Der Abschnitt schließt mit einer kurzen Diskussion der Bedeutung von P2P-Systemen im Kontext der Computational Grids.

P2P-Systeme

Peer-to-Peer steht für eine Klasse von verteilten Anwendungen, die den Austausch von Ressourcen wie Rechenleistung, Speicherplatz, Daten oder Bandbreite der angeschlossenen Knoten (Teilnehmer, Peers) auf Basis einer vorhandenen Netzwerkstruktur realisieren. Die Teilnehmer eines P2P-Netzwerks agieren dabei gleichzeitig sowohl als Server als auch als Client. Eines der prominentesten Beispiele stellen File-Sharing-Applikationen wie z.B. Gnutella, Napster oder KaZaa dar, die sich in unterschiedlichem Maße der P2P-Architektur bedienen.
Nachfolgend werden nun grundlegende Eigenschaften dieser Systeme vorgestellt, und anschließend im Rahmen eines Vergleiches mit klassischen Client-Server-Architekturen qualitativ bewertet:

Dezentral und Selbstorganisierend: P2P-Systeme arbeiten vollständig dezentral, d.h. es gibt keinen zentralen Knoten, der die Interaktionen der Peers untereinander koordiniert. Dementsprechend existiert auch kein zentrales Verzeichnis verfügbarer

Ressourcen. Jeder Peer verfügt über Informationen, die einen Teil der im Gesamtsystem verfügbaren Ressourcen darstellen. Folglich verfügt in solchen Systemen kein Peer über eine globale Sicht des Systems. Weiterhin wird im Zusammenhang mit P2P-Systemen häufig der Begriff der Rollensymmetrie gebraucht, da jeder Peer sowohl als Client als auch als Server interagiert (vgl. Aberer u. Hauswirth, 2002).

P2P-Systeme sind selbstorganisierende Systeme, in denen aus rein lokalen Interaktionen ein komplexes, globales Systemverhalten entsteht. Dies wird vor allem durch das bereits oben erwähnte Fehlen einer zentralen Kontrollinstanz erreicht. Selbstorganisationsprozesse werden durch Zufallsprozesse wie beispielsweise Suchanfragen oder das Einfügen von Daten in das System ausgelöst.

Fehlertoleranz: Bedingt durch das Fehlen einer zentralen Kontrollinstanz führt der Ausfall eines einzelnen oder auch mehrerer Knoten nicht zu einem Versagen des gesamten Systems und Robustheit gegenüber partiellen Systemausfällen wird inhärent durch die zugrundeliegende Struktur erreicht.

Skalierbarkeit: Im Hinblick auf die Skalierbarkeit hinsichtlich der Anzahl der Teilnehmer bietet die dezentrale Organisation der P2P-Architektur Vorteile. Der Organisationsaufwand steigt keinesfalls linear mit der Anzahl der Teilnehmer, da keinerlei globale Information vorgehalten und gepflegt werden muss (vgl. Dornberger u. Fuchs, 2004).

P2P-Systeme bieten im Gegensatz zu Client-Server-Netzwerken eine bessere Skalierbarkeit und niedrige Betriebskosten, da es keinen zentralen Server gibt, der zu warten ist. Im Vergleich zu Client-Server-Systemen werden diese Vorteile aber durch komplexe Such- und Wartungsalgorithmen sowie einen zusätzlichen Sicherheitsaufwand erkauft. Der P2P-Ansatz bietet allerdings für viele Bereiche, in denen Client-Server-Modelle momentan die Standardarchitektur darstellen, eine gute Alternative um Skalierungsprobleme zu lösen. Es ist möglich, die Anzahl der Peers zu erhöhen, ohne dass die Netzwerklast stark zunimmt (vgl. Foster, 2003). P2P-Systeme erlauben eine effiziente Suche nach verteilten Ressourcen, wofür Indexstrukturen von zentraler Bedeutung sind.

P2P-Anwendungen

In der P2P-Welt wird eine Unterscheidung zwischen hybriden, unstrukturierten und strukturierten P2P-Systemen getroffen. Im folgenden wird jeweils ein Beispiel für eines der Systeme gegeben und dabei die Eigenschaften beschrieben.

Napster [1] ist ein Beispiel für eine hybride P2P-Architektur (vgl. Abb. 3.1), d.h. es kommt eine Kombination des Client-Server- und des P2P-Ansatzes zum Einsatz. Der Napster-Server verwaltet eine zentrale Datenbank der verfügbaren Dateien,

[1] `http://www.napster.com` (08/2007)

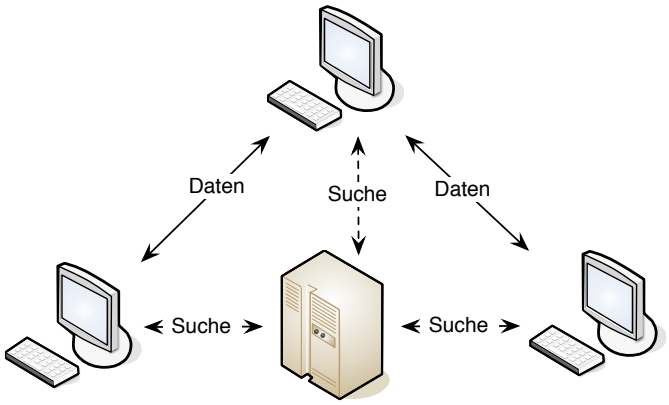


Abbildung 3.1: Schematische Übersicht der Napster Netzwerk-Struktur.

und die jeweilige Internet Protocol (IP)-Adresse der Peers, die die jeweilige Datei aktuell anbieten. Weiterhin wird sowohl der Verbindungsaufbau zwischen den Peers als auch die Suche zentral über einen Server von Napster vermittelt. Lediglich der eigentliche Transfer der Dateien geschieht direkt zwischen den Peers, und führt zu einer signifikanten Entlastung der zentralen Server. Anforderungen an die Dezentralität und die Skalierbarkeit werden von Napster nicht erfüllt.

Gnutella [2] zählt zu den unstrukturierten P2P-Systemen, d.h. die Peers verwalten keine Information über die Ressourcen, die andere Peers zur Verfügung stellen. Die Indexinformationen in Bezug auf eine Ressource werden nur von einem Peers selbst verwaltet (vgl. Abb. 3.2). Somit ist eine gezielte Suche nach Ressourcen nicht möglich, sondern erfordert einen Broadcast der Anfrage an alle bekannten Peers. Um die Überlastung des Netzwerks zu verhindern, wird jede Suchanfrage dabei mit einem Time-To-Live Flag versehen, das dafür sorgt, dass die Anfrage nach einer vorgegebenen Anzahl von Weiterleitungen verworfen wird.

Die Gnutella-Architektur weist einen hohen Grad an Unabhängigkeit und Fehlertoleranz auf, realisiert diese Eigenschaften aber auf Kosten von Effizienz (Bandbreitenverschwendung) und deterministischem Systemverhalten. Die durch Suchanfragen generierte Netzlast macht in Gnutella-Netzwerken bereits einen signifikanten Anteil der gesamten Netzlast aus.

KaZaa [3] vereint Merkmale hybrider und unstrukturierter P2P-Systeme, allerdings kommen keine zentralen Server zum Einsatz. Alle teilnehmenden Peers werden in zwei Gruppen, so genannte *Ordinary Nodes* und *Super Nodes* eingeteilt. Super Nodes sind Peers, die sich durch zuverlässiges Verhalten und ausgeglichene Up- bzw. Download-Verhältnisse auszeichnen. Ordinary Nodes melden sich bei diesen Super Nodes an, und übermitteln eine Liste der zur Verfügung gestellten Ressourcen.

[2] `http://www.gnutella.com` (08/2007)
[3] `http://wwww.kazaa.com` (08/2007)

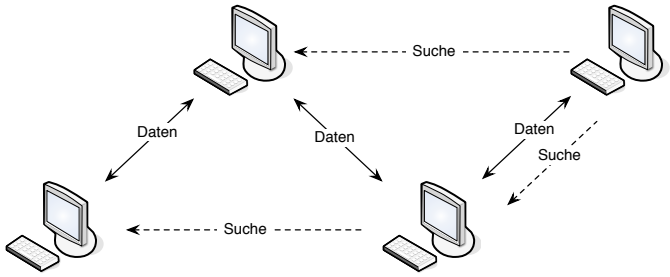


Abbildung 3.2: Schematische Übersicht der Gnutella Netzwerk-Struktur.

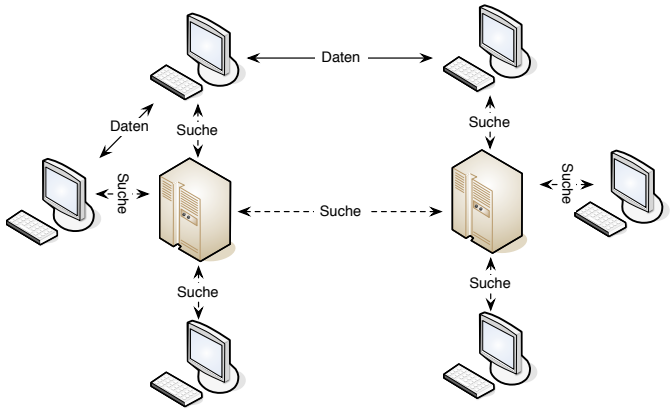


Abbildung 3.3: Schematische Übersicht der KaZaa Netzwerk-Struktur.

Eine Suchanfrage von einem Ordinary Node erfolgt dann gezielt an den bekannten Super Node, der zunächst den von ihm verwalteten Index durchsucht und in einem weiteren Schritt andere Super-Nodes anfragt. Der Datenaustausch findet wiederum direkt zwischen den Ordinary Nodes statt (vgl. Abb. 3.3).

Peer-to-Peer Systeme und Grid-Computing

P2P-Systeme, welche Anwendungen wie Napster und Gnutella bekannt gemacht haben, sind in den vergangen Jahren vor allem durch den Austausch von Daten popularisiert worden. Die Entwicklung und der Einsatz von Grid-Computing dagegen wird von den Erfordernissen der Gemeinschaften motiviert, die auf entfernte Ressourcen wie z.B. auf Rechner für aufwändige Simulationen und Datenanalyse zugreifen. Die beiden Systeme haben ähnliche Ziele (koordinierten Zusammenschluss und Verwendung von verteilten Ressourcen) konzentrieren sich aber auf verschiedene Erfordernisse (vgl. Iamnitchi et al., 2002). Die wesentlichen Unterschiede zwischen den beiden Systemen liegen in der Heterogenität der Ressourcen, Benutzerverhalten sowie Fehleranfälligkeit. Weitere grundlegende Unterschiede zwischen P2P-Systemen und Grid-Computing sind in der Tabelle 3.1 zusammengefasst worden.

Viele Gegenwärtige Grid-Systeme wie Globus[4] oder UNICORE[5] verwalten hunderte von Servern sowie tausende Clients. Aufgrund der wachsenden Anzahl von Jobs sowie größeren Datenmengen skalieren wichtige Bestandteile dieser Systeme, wie das Auffinden von Ressourcen, Scheduling oder Replikationsmanagement, nicht. Die Verwendung von P2P-Technologien vermindert diese Probleme weitgehend. Der Ansatz, um Zuverlässigkeit und Skalierbarkeit für das Grid-Computing zu realisieren, liegt in der Nutzung von Techniken der P2P-Systeme. Die flachen, nicht hierarchischen P2P-Systeme können mit Hilfe ihrer Eigenschaften wie Selbstorganisation eine Skalierbarkeit des Systems gewährleisten oder mit Hilfe von Redundanzmechanismen die Verfügbarkeit von Daten und damit eine bessere Ausfallsicherheit garantieren. Diese genannten Grid-Systeme benutzen den Open Grid Services Infrastructure (OGSI)-Standard für den Zugriff auf Ressourcen. Dabei besteht die Möglichkeit, die Schnittstelle der einzelnen P2P-Komponenten an den OGSI-Standard anzupassen (vgl. Reinefeld et al., 2004).
In den letzten Jahren ist daher eine Konvergenz von Grid- und P2P-Systemen zu erkennen. Dabei werden P2P-Technologien auf Grid-Systeme übertragen, um sie skalierbar, selbstorganisierend und vor allen Dingen zuverlässig zu machen. Iamnitchi und Foster (vgl. Iamnitchi et al., 2002) sehen Grid-Computing und P2P-Systeme zusammenwachsen. Dabei werden die zur Verfügung gestellten Ressourcen sehr unterschiedlich sein. Diese Ressourcen reichen von Computern und Daten über wissenschaftliche Instrumente oder Speicherplatz bis hin zu unterschiedlichen Betriebssystemen, unterschiedlichen Anzahlen von CPUs und Daten unterschiedlicher Größen.
Ein Beispiel, in dem eine Annäherung von beiden Systemen zu erkennen ist, ist ein Ansatz, der auf P2P und dem Komponentenmodell basiert (vgl. Reinefeld et al., 2004). Dabei werden die einzelnen Dienste, die von komplexen Grid-Systemen angeboten werden, durch ein einzelnes, speziell dafür ausgelegtes P2P-System erbracht. Die Komponenten werden durch einzelne P2P-Systeme realisiert, die spezielle Aufgaben erfüllen. Beispielsweise sorgt eine Komponente für die Verfügbarkeit der Daten, ein anderes für die Synchronisation dieser Daten. Ein Vergleich zwischen unterschiedlichen Ansätzen in dem Grid-Systeme mit Hilfe von P2P-Technologien realisiert werden, findet sich in Trunfio et al. (2006).

Einsatz von Peer-to-Peer-Mechanismen

Zum Auffinden von anderen Systemen im Grid wird ein P2P-Mechanismus verwendet. Im Vergleich zu einer Lösung mit einem zentralen Anmeldeserver bietet eine P2P-basierte Lösung wesentliche Vorteile wie höhere Skalierbarkeit durch eine flache, nicht-hierarchische Topologie, Autonomie, Selbstorganisation und hohe Ausfallsicherheit durch Redundanz.

Netzwerkcharakteristik

Die simulierte Grid-Umgebung verzichtet auf die Simulation von Netzwerkeigenschaften wie Verzögerungen, Ausfall von Kommunikationsverbindungen und dergleichen. Dies be-

[4] `http://www.globus.org` (08/2007)
[5] `http://www.unicore.eu` (08/2007)

Merkmale	Grid-Computing	P2P-Systeme
Benutzer	wissenschaftliche Gemeinschaft	Desktop Benutzer
Maschinen	Workstations, Mehrprozessorsysteme, Cluster	Desktop
Netzwerk	Hochgeschwindigkeitsnetz	Internet, TCP-Verbindung
Administration	zentral oder hierarchisch	verteilt, flach
Anwendungen	wissenschaftliche Anwendungen wie Simulationen oder Datenanalyse	Datenverteilung, Echtzeit-Streaming
Größenordnung	verhältnismäßig kleine Anzahl von speziellen Sites	Anschlussmöglichkeit aus allen Desktop-Rechner (große Ausdehnung)
Sicherheit	Sicherheitsdienste zum Einreichen und zur Ausführung der Jobs	Protokolle für Datenverteilung
Teilnahme	statisch oder sich langsam ändernde Anbindung der einzelnen Knoten	sehr dynamisch, zufällige Knotenanbindung
Vertrauen	zuverlässige Benutzer	anonyme Benutzer, kein Vertrauen

Tabelle 3.1: Die wesentlichen Unterschiede zwischen Grid-Computing und P2P-Systeme (vgl. Trunfio et al., 2006)

trifft sowohl die von den Systemen ausgetauschten Nachrichten zur Suche nach Partnern als auch die Übertragung der eigentlichen Rechenjobs zwischen einzelnen Systemen.
Da der Schwerpunkt der Experimente auf der Untersuchung der Auslastung der am Grid beteiligten Rechnersysteme liegt und die Jobs allein anhand der von ihnen benötigten Central Processing Unit (CPU)-Anzahl sowie der Ausführungsdauer charakterisiert werden, haben Verzögerungen durch das Netzwerk nur einen sehr geringen Einfluss auf die Jobausführung.
Die Simulation des Ausfalls von Kommunikationsverbindungen ist ebenfalls wenig sinnvoll, da es sich bei Grid-Umgebungen um Zusammenschlüsse von hochverfügbaren Systemen mit einer extrem geringen Ausfallwahrscheinlichkeit von unter $0,01\%$ handelt. Anders als bei herkömmlichen P2P-Netzen gehören ausfallende Peers bei Grid-Systemen nicht zum normalen Betriebszustand. Die Simulation eines wenige Minuten dauernden Ausfalls einer Kommunikationsverbindung während eines simulierten Zeitraums von mehreren Monaten würde sich vermutlich nicht signifikant auf die Simulationsdaten auswirken.
Weiterhin müssten solche Ausfälle auf Basis von Verteilungsfunktionen simuliert werden, wodurch die Simulation nicht mehr deterministisch arbeiten und die anschließende Auswertung der Daten wesentlich erschwert werden würde. Schlussendlich wäre die voll-

ständige Simulation eines Netzwerkes sehr komplex und würde die Geschwindigkeit der Simulation deutlich verringern.

3.2 Scheduling-Theorie

Dieser Abschnitt gibt eine kurze Einführung in das Scheduling-Problem auf Multi-Prozessor (Multi-Processor-Site, MPS) Systemen und führt grundlegende Notationen bzw. Begrifflichkeiten ein, die im weiteren Verlauf des Berichts häufig verwendet werden.

Das Scheduling-Problem auf Multi-Prozessor Systemen

Für die Definition des Scheduling-Problems ist zunächst die Definition des zugrundeliegenden Maschinen- sowie Job-Modells notwendig. Ein einzelnes MPS wird dabei durch m identische Prozessoren modelliert, von denen beliebige Teilmengen an auszuführende Jobs zugewiesen werden können. Weiterhin verfügt jedes MPS über eine lokale Job-Queue, die von lokalen Benutzern gefüllt wird, sowie einen Scheduling-Algorithmus, der für die Verwaltung des Systems verantwortlich ist. Er greift auf die Warteschlange zu, weist den dort wartenden Jobs die angeforderte Menge von Prozessoren zu und bringt sie somit zur Ausführung. Abbildung 3.4 beschreibt schematisch diese Struktur.
Im Verlauf des Berichtes wird das folgende formale Job-Modell verwendet. Ein einzelner Job j wird dabei durch das Tupel $(r_j, \bar{p}_j, m_j)$ beschrieben, mit:

r_j : Der Zeitpunkt zu dem der Job j dem Scheduling-System übergeben worden ist.

$\bar{p}_j$: Die von Job j angeforderte, vom Benutzer geschätzte Rechenzeit.

m_j : Die von Job j benötigte Anzahl von Prozessoren.

Informationen über die tatsächliche Laufzeit der Jobs bzw. deren Endzeitpunkt sind zum Zeitpunkt der Eingabe nicht verfügbar. Es handelt sich hier also um die Online-Variante des Scheduling-Problems, das sich wie folgt definiert.
Sei ein MPS mit m Prozessoren und einer Job-Queue Q gegeben. Lokale Nutzer übergeben dem System Jobs $j = (r_j, \bar{p}_j, m_j)$, die in Q eingefügt werden. Jeder Job $j \in Q$ muss ausgeführt werden, d.h. es müssen ihm m_j Prozessoren für maximal $\bar{p}_j$ Zeiteinheiten zur Verfügung gestellt werden. Ziel ist es, die Auslastung des MPS zu maximieren sowie die durchschnittliche Wartezeit der Jobs bis zu ihrer Ausführung zu minimieren. Die fehlende Information über die tatsächliche Laufzeit der Jobs sowie die Möglichkeit ständig neu hinzukommender Jobs stellt an dieser Stelle das zentrale Problem dar.

Scheduling Algorithmen

Abschließend werden die drei bekanntesten und am häufigsten in der Praxis verwendeten Scheduling-Algorithmen vorgestellt und ihre Funktionsweise anhand eines Beispiels illustriert.

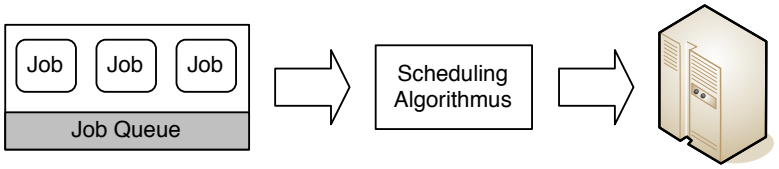


Abbildung 3.4: Schematischer Aufbau eines Scheduling Systems

Job	r_j	m_j	p_j
1	0	10	1
2	0	1	5
3	0	10	1
4	0	1	5
5	0	10	1

Abbildung 3.6: Belegung der Job-Queue für das worst-case Beispiel des FCFS. Ein MPS mit $m = 10$ Prozessoren wird angenommen

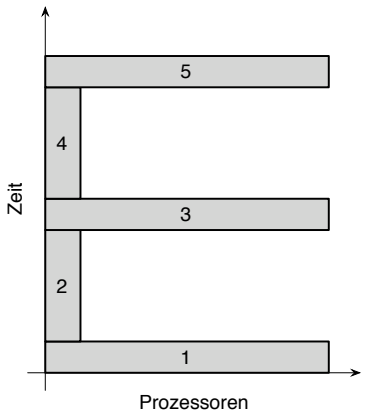


Abbildung 3.8: FCFS-Resultierender Schedule

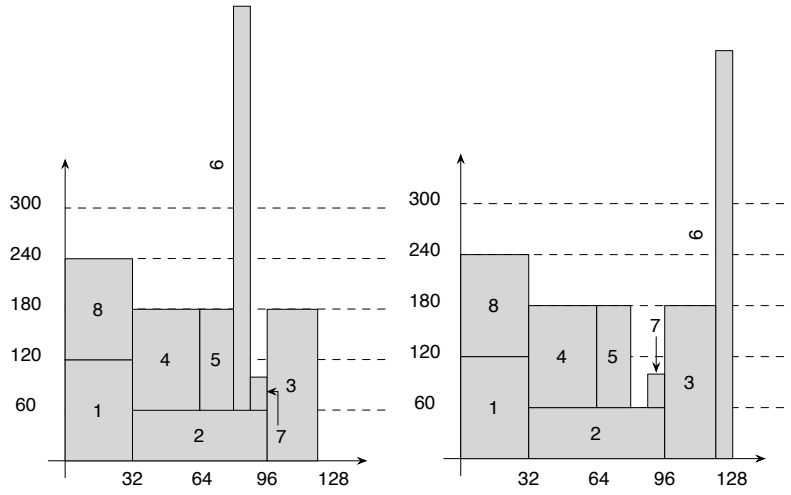
First-Come-First-Serve (First Come First Serve (FCFS)) führt den ersten, d.h. den am längsten wartenden Job in der Warteschlange genau dann aus, sobald genügend freie Prozessoren verfügbar sind. Der Algorithmus überprüft den ersten Job der Warteschlange dann, wenn die Ausführung eines anderen Jobs beendet wird oder sich ein neuer Job am Anfang der Warteschlange befindet. Im worst-case (s. Abb. 3.8) resultiert diese Heuristik in einer sehr geringen Auslastung, führt in der Praxis jedoch häufig zu brauchbaren Ergebnissen (vgl. Graham, 1969).

EASY Backfilling benötigt die erwartete Rechenzeit $\bar{p}_j$ eines Jobs j. Sei t der aktuelle Zeitpunkt, j der aktuell erste Job in der Warteschlange und $T_j \geq t$ der frühestmögliche Zeitpunkt, zu dem j gestartet werden kann. Falls j nicht gestartet werden kann, d.h. $T_j > t$, wird der erste Job k aus der Warteschlange gestartet, für den genügend freie Ressourcen verfügbar sind und dessen Ausführung den Start von j nicht weiter verzögert, d.h. $t + \bar{p}_k \leq T_j$ muss gelten. Die Auswirkungen der Schätzung $\bar{p}_j$ auf die Effizienz des Algorithmus hinsichtlich der Zielsetzung einer maximalen Auslastung bei minimalen Wartezeiten wurden von Schwiegelshohn u. Yahyapour (2000) untersucht. Das Conservative Backfilling stellt eine weitere Variante des allgemeinen Backfilling Ansatzes dar. Sei ein Job j gegeben, für den aktuell ausreichend Prozes-

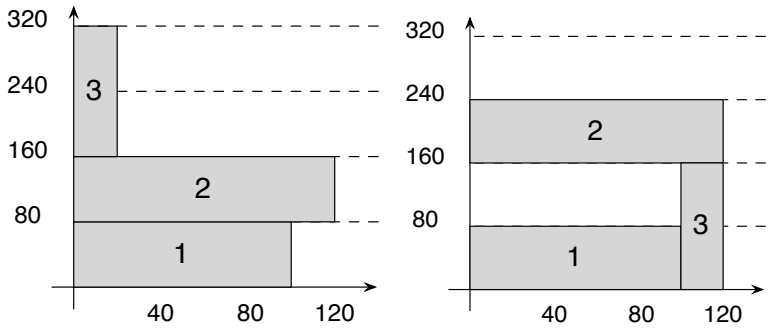
soren zur Verfügung stehen. Die Conservative Backfilling Strategie führt j genau dann aus, falls für alle anderen Jobs j', die sich vor j in der Queue befinden, gilt, dass sie durch die Ausführung von j nicht verzögert werden. Ein detaillierter Vergleich unterschiedlicher Backfilling Strategien findet sich in Weil (1998). Aufgrund des hohen Aufwandes an Rechenzeit, der für das Conservative Backfilling benötigt wird, wird bevorzugt das EASY Backfilling in der Praxis eingesetzt.

LIST stellt eine weitere Backfilling-Strategie dar. Falls der erste Job in der Warteschlange aktuell nicht gestartet werden kann, wird die Warteschlange iteriert, bis ein Job gefunden wurde, der mit den aktuell verfügbaren Ressourcen ausgeführt werden kann oder das Ende der Warteschlange erreicht wird. Zusätzliche Überprüfungen, wie sie beispielsweise das EASY oder Conservative Backfilling ausführen, sind nicht vorgesehen. Ein ähnliches, wenn auch weit komplexeres Vorgehen wurde bereits von Graham (1969) vorgestellt.

Anhand der in Tabelle 3.2 dargestellten Belegung der Job-Queue wird nun noch einmal detailliert auf das Vorgehen der drei Algorithmen eingegangen. Für alle Jobs j in der Job-Queue wird $p_j = \bar{p}_j$ angenommen. Die Job-Queue ist Teil eines MPS mit $m = 128$ Prozessoren.
Alle drei Algorithmen starten mit einem vollständig verfügbaren MPS, und fügen Job j_1 bis j_3 in den Schedule ein. Damit sind 120 Prozessoren belegt. Der FCFS stoppt an dieser Stelle und muss warten, bis wieder genügend Ressourcen für den Job j_4 verfügbar sind. Sowohl der EASY-Backfilling- als auch der LIST-Algorithmus iterieren nun die verbleibenden Jobs in der Queue, auf der Suche nach einem Job, der mit den aktuell verfügbaren acht Prozessoren zur Ausführung gebracht werden kann. Beide Algorithmen werden bei Job j_6 fündig. Der EASY-Backfilling-Algorithmus überprüft nun, ob Job j_4 weiterhin nach maximal 60 Zeiteinheiten gestartet werden kann, falls Job j_6 zur Ausführung gebracht wird. Der LIST-Algorithmus führt diese Überprüfung nicht durch.
Nach 60 Zeiteinheiten endet Job j_2 und 64 CPUs werden wieder verfügbar. Alle drei Algorithmen bringen nun Job j_4 und Job j_5 zur Ausführung. Der FCFS fügt allerdings erst jetzt Job j_6 dem Schedule hinzu. Job j_7 wiederum wird von allen Algorithmen angestartet. Für den Job j_8 stehen zu diesem Zeitpunkt nicht ausreichend Prozessoren zur Verfügung. Dies wird erst der Fall sein, nachdem Job j_1 beendet worden ist, d.h. nach weiteren 60 Zeiteinheiten. Die resultierenden Schedules finden sich in Abbildung 3.9.
Zur Illustration der unterschiedlichen Vorgehensweisen von EASY und LIST werden nun die folgenden Jobs betrachtet. Der Job j_1 mit $m_1 = 100$ und $p_1 = 80$, der Job j_2 mit $m_2 = 120$ und $p_2 = 80$, sowie der Job j_3 mit $m_3 = 20$ und $p_3 = 160$. Für alle Jobs wird angenommen, dass die geschätzte Rechenzeit gleich der tatsächlichen Rechenzeit ist. Sie sollen auf einem MPS mit $m = 128$ Prozessoren ausgeführt werden. Sowohl der EASY- als auch der LIST-Algorithmus starten j_1 an. Damit sind 100 der 120 Prozessoren für 80 Zeiteinheiten belegt. Somit kann j_2 nicht direkt angestartet werden und beide Algorithmen betrachten j_3 als potentiellen Kandidaten für das Backfilling. Der Job j_2 kann frühestmöglich zum Zeitpunkt 81 angestartet werden. EASY betrachtet nun die geschätzte Laufzeit $\bar{p_3} = 160$ und stellt fest, dass sich die Ausführung von j_2 durch die



(a) FCFS - Resultierender Schedule (b) EASY / LIST - Resultierender Schedule



(c) EASY - Resultierender Schedule (d) LIST - Resultierender Schedule

Abbildung 3.9: Die oberen Abbildungen zeigen die resultierende Schedules für die Algorithmen FCFS, EASY und LIST für die in Tabelle 3.2 aufgeführte Belegung der Job-Queue. Die unteren Abbildungen zeigen die Schedules für das in 3.2 vorgestellte Beispiel.

vorzeitige Ausführung von j_3 zusätzlich verzögern würde. EASY bricht ab und wartet auf das Bearbeitungsende von j_1 um zunächst j_2 und anschliessend j_3 zur Ausführung zu bringen. Der LIST-Algorithmus hingegen führt die Überprüfung der zusätzlichen Verzögerung nicht durch, zieht j_3 vor und bringt schliesslich j_2 nach der Beendigung von j_3 zur Ausführung. Die resultierenden Schedules finden sich in Abbildung 3.9.

Verzicht auf Preemption im Scheduler

Das Scheduling-System der simulierten MPS-Systeme wurde so ausgelegt, das Preemption, also das Einfrieren und zeitversetzte Weiterbearbeiten eines Jobs, um einen anderen Job an dessen Stelle zu berechnen, nicht unterstützt wird. Dieses Merkmal ist zwar bei

Job	r_j	m_j	p_j
j_1	0	32	120
j_2	0	64	60
j_3	0	24	180
j_4	0	32	120
j_5	0	16	120
j_6	0	8	480
j_7	0	8	40
j_8	0	32	120

Tabelle 3.2: Belegung der Job-Queue

modernen Betriebssystemen für Server und Workstations durchweg vorhanden, jedoch nicht im Bereich der MPS-Systeme. Sofern das Betriebssystem eines MPS-Systems Preemption überhaupt unterstützt, ist es dort ein optionales Feature. Die Verbreitung von Preemption bei MPS-Systemen ist in der Praxis allerdings sehr gering. Dies liegt vor allem daran, dass Sitebetreiber und Nutzer zusätzlichen Aufwand betreiben müssten, um Preemption durch Verfahren wie z.B. Checkpointing zu realisieren. Die meisten Nutzer sind indessen eher interessiert daran, dass ihre Jobs eben nicht unterbrochen werden. Um möglichst realistische und praxisnahe Ergebnisse zu erzielen und die Komplexität der Simulation nicht unnötig zu steigern, ist deshalb auf die Implementierung von Preemption im Scheduling-System verzichtet worden.

Das Standard-Workload-Format

Um unterschiedliche Scheduling-Algorithmen miteinander vergleichen zu können, müssen die resultierenden Schedules in sogenannten Traces gespeichert und eben diese in einem weiteren Verarbeitungsschritt analysiert werden. Im Rahmen dieser Projektgruppe (PG) wurde das von Chapin et al. (1999a) vorgestellte Standard Workload Format (SWF) verwendet, dessen Job-Modell kompatibel zu dem hier verwendeten Job-Modell ist. Für einen Job j ist weiterhin die tatsächliche Laufzeit p_j und somit auch der entsprechenden Endzeitpunkt $C_j(S)$ im Schedule S verfügbar. Einzelne Jobs werden zeilenweise organisiert, und jeder Spalte ist ein entsprechender Parameter eines Jobs zugewiesen. Weiterhin werden Wertebereiche und Standardwerte definiert.
Zusätzlich zu der eigentlichen Format-Definition stellt Feitelson unter `http://www.cs.huji.ac.il/labs/parallel/workload/index.html` (08/2007) einige Traces realer Systeme sowie die Konfigurationen der entsprechenden MPS-Systeme zur Verfügung. Diese dienen, wie bereits oben erwähnt, als Eingabe bzw. Grundlage für die in Kapitel 7 durchgeführten Simulationen. Für eine genauere Beschreibung der in den Simulationen verwendeten Teildaten aus den Traces und Bedeutung der in diesem Bericht verwendeten Bezeichnungen siehe Kapitel 7.1.

4 Aufbau des Systems

Inhalt

Dieses Kapitel dient zur Beschreibung des Aufbaus und der Funktionsweise der wesentlichen Komponenten des Multi-Processor-Site (MPS)-Systems. Zunächst wird ein grober Überblick über die Aufgaben der einzelnen Komponenten des Systems gegeben und beschrieben, wie diese miteinander interagieren. Anschließend wird ihre Funktionsweise detaillierter beschrieben.

4.1 Überblick

Ein (MPS)-System besteht im allgemeinem aus einem Hochleistungsrechner, welcher mit vielen leistungsstarken CPUs in Form von so genannten Rechenknoten (englisch: nodes) ausgestattet ist und einem speziellen Betriebssystem, das die zu berechnenden Aufgaben möglichst gleichmäßig auf die vorhandenen CPUs verteilt. Des Weiteren ist ein großer Massenspeicher und eine schnelle Netzwerkanbindung, sowohl zwischen den Rechenknoten als auch nach außen hin, vorhanden.
Um ein solches MPS-System simulieren zu können, muss zunächst ein Modell entwickelt werden, welches von der Realität abstrahiert. Es wird angenommen, dass das zu simulierende System über einen unbegrenzten Haupt- und Massenspeicher verfügt. Da bei der Simulation nur der Austausch von Jobs sowie die Parameter im Zusammenhang mit ihrer Berechnung (z.B. Berechnungsdauer, Antwortzeit und dergleichen) von Interesse sind, sind diese Annahmen legitim.
Unter Abschnitt 2 wurden reale Grid-Systeme und Systeme zur Simulation von Grid-Umgebungen beschrieben. Der Aufbau dieser Systeme galt als Grundlage für das zu entwickelnde System. Abweichend von den existierenden Systemen sollte aber keine zentrale Informationsquelle zur Verwaltung der Ressourcen zur Verfügung stehen und jegliche Kommunikation sollte mit Hilfe von P2P-Mechanismen realisiert werden. Des Weiteren sollte auch das Scheduling vollkommen dezentral ablaufen.

Hierzu wurden in einem nächsten Schritt wichtige Komponenten des Systems identifiziert, welche für die Ausführung von Jobs und die Teilnahme an einem Grid-System benötigt werden.
Eine zentrale Komponente bildet das Scheduling-System (siehe Abschnitt 4.4), welches mit Hilfe eines Scheduling-Algorithmus (siehe Abschnitt 4.5) die auszuführenden Jobs auf die Ressourcen des MPS-Systems verteilt.
Als zweites ist eine Komponente zur Kommunikation mit anderen Grid-Teilnehmern erforderlich, die auch einen Dienst zum Transfer von Jobs von bzw. zu anderen Systemen bereitstellt (siehe Abschnitt 4.6).
Als drittes Systemelement wird eine Entscheidungskomponente benötigt, welche anhand von verschieden Kriterien entscheiden kann, ob ein lokal eingereichter Job (im Folgenden auch als „lokaler Job“ bezeichnet) auch lokal ausgeführt wird, oder ob versucht werden soll, ihn auf einer entfernten Site zur Ausführung zu bringen. Eine weitere Aufgabe dieser Komponente ist die Entscheidung, ob Jobs von externen Sites (im Folgenden auch als „externer Job“ bezeichnet) für die lokale Berechnung akzeptiert werden sollen (siehe Abschnitt 5.2).
Die vierte Komponente bildet der Informationsdienst (siehe Abschnitt 4.2). Er sammelt an einer zentralen Stelle zur Laufzeit Informationen über Systemparameter und berechnet daraus verschiedene Metriken, die den anderen lokalen Systemkomponenten zur Verfügung gestellt werden.
Des Weiteren hält er „öffentliche“ Informationen über das System für die anderen Grid-Teilnehmer bereit, die von diesen abgefragt werden können. Dies sind zum Beispiel der Name der Site und die Anzahl der zur Verfügung stehenden CPUs.
Einen Überblick über die Systemkomponenten und wie sie miteinander interagieren zeigt Abbildung 4.1. Die mit Ziffern gekennzeichneten Pfeile stellen den Informationsfluss dar, also die Kommunikation zwischen den einzelnen Komponenten. Die mit Buchstaben gekennzeichneten Pfeile symbolisieren die Wege, über die Jobs in dem gegebenen System laufen.
Im Folgenden wird erklärt, welche Informationen die Komponenten aus Abbildung 4.1 austauschen und wie die Informationen ausgetauscht werden. Der mit 1 bezeichnete Pfeil zeigt den Informationsfluss vom Scheduling-System zum Informationsdienst. Der Informationsdienst erhält darüber aktuelle Informationen über die Auslastung des Systems, sowie Informationen über Statuswechsel der sich im System befindenden Jobs sowie den Zustand der Komponenten des Scheduling-Systems. Dazu gehören zum Beispiel Angaben über die CPU-Auslastung, die Belegung im Schedule und in der lokalen Jobqueue.
Die mit 2 bezeichnete Kante zeigt den Informationfluss vom Informationsdienst zur Entscheidungskomponente ODIN (siehe Kapitel 4.3). Der Informationsdienst stellt der Entscheidungskomponente sowohl lokale, als auch öffentliche Informationen zur Verfügung. Informationen zum aktuellen Systemzustand können dabei die Entscheidung, ob ein Job lokal oder remote ausgeführt werden soll, beeinflussen. Weiterhin können diese Informationen für die Entscheidung, ob ein Job von einer entfernten MPS angenommen wird, herangezogen werden.
Die Kanten 3 und 4 zeigen den Informationsfluss zwischen der Entscheidungskomponente

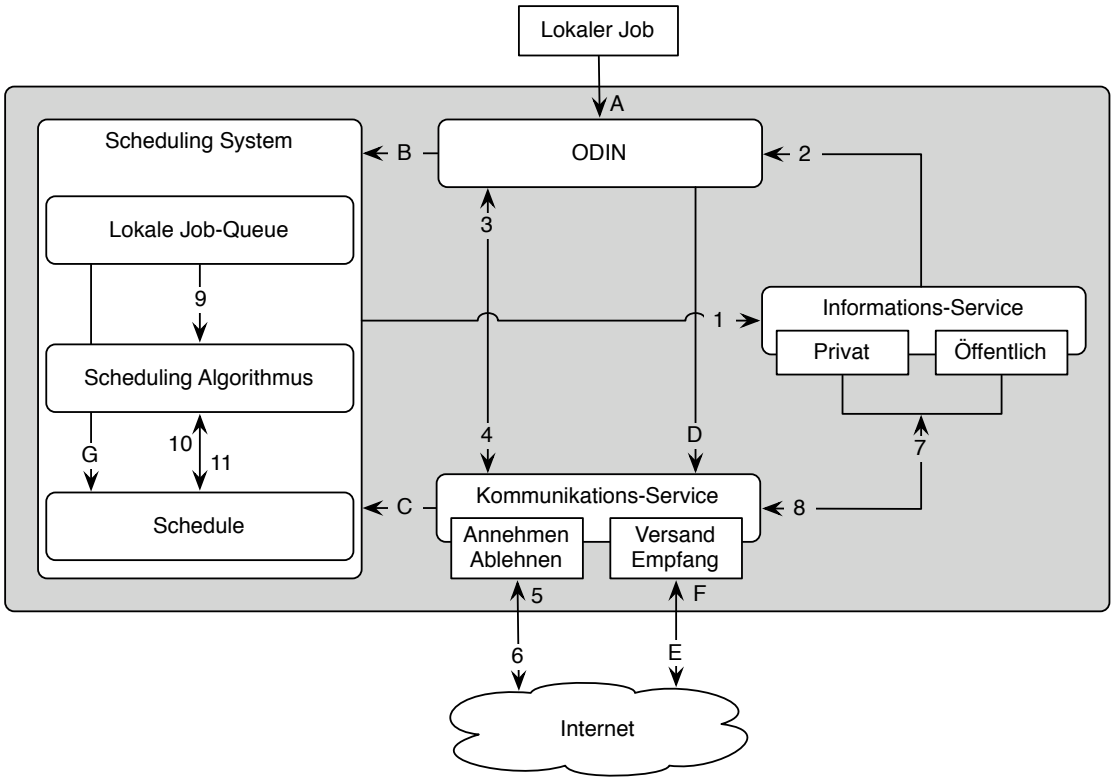


Abbildung 4.1: Übersicht über eine Multi-Processor-Site

(ODIN) und dem Kommunikationsdienst. Der Kommunikationsdienst liefert Informationen, die er aus eingehenden Anfragen anderer Grid-Teilnehmer extrahiert, an ODIN. Anhand dieser Informationen kann entschiedenen werden, ob ein externer Job angenommen oder abgelehnt wird. Des Weiteren informiert der Kommunikationsdienst die Entscheidungskomponente über die Entscheidung externer MPS bzgl. der eigenen Anfragen. Der Kommunikationsdienst erhält von ODIN Jobbeschreibungen, die jeweils Informationen über einen Job beinhalten (wie z.B. die zur Berechnung erforderliche CPU-Anzahl oder die vom Benutzer geschätzte Ausführungszeit), und Namen anderer Grid-Teilnehmer, die er um die Ausführung eines lokal eingereichten Jobs bitten soll. Zusätzlich erhält er von ODIN positive oder negative Antworten über die Ausführungsanfragen für externe Jobs.

Die mit 5 und 6 bezeichnete Kante stellt den Informationsfluss zwischen dem Kommunikationsdienst und anderen Grid-Teilnehmern dar. Der Kommunikationsdienst erhält Anfragen und Jobbeschreibungen von anderen Sites, sowie Antworten auf eigene Anfragen. Er sendet Anfragen und Jobbeschreibungen an andere Sites, sowie Antworten auf erhaltene Anfragen. Des Weiteren werden über diese Schnittstelle öffentliche Informationen über das lokale und bekannte externe Systeme ausgetauscht.

Die Kanten 7 und 8 zeigen den Informationsfluss zwischen dem Informationsdienst und dem Kommunikationsdienst. Der Informationsdienst übergibt öffentliche Informationen

über das lokale System an den Kommunikationsdienst, der diese Informationen für das An- und Abmelden am Grid verwendet.
Über die mit 9 bezeichnete Kante bezieht der Scheduling-Algorithmus Informationen über Jobs in der lokalen Queue. Über die Kanten 10 und 11 werden Informationen zwischen dem Schedule und dem Scheduling-Algorithmus ausgetauscht.
Nachfolgend werden die Wege beschrieben, auf denen sich ein Job in einer Site bewegen kann. Die mit A bezeichnete Kante beschreibt den Weg eines Jobs, der von einem lokalen Benutzer auf der Site eingereicht wird. Über die Kante B werden eingereichte Jobs zur lokalen Ausführung an die lokale Queue weitergeleitet. Die Kante C zeigt den Weg über den externe Jobs zur lokalen Ausführung an die Entscheidungskomponente übermittelt werden. Über die Kante D werden lokale Jobs zur externen Ausführung an den Kommunikationsdienst weitergeleitet. Lokale Jobs werden an eine entfernte Site über die Kante E übermittelt. Kante F beschreibt den Weg über den externe Jobs zur lokalen Ausführung entgegengenommen werden. Schließlich beschreibt Kante G den Weg über den ein Job aus der lokalen Queue im Schedule platziert wird.
Im Folgenden werden die oben erwähnten Komponenten, ihr Aufbau und ihre Funktion näher beschrieben.

4.2 Der Informationsdienst

Der Informationsdienst ist die zentrale Komponente, die Informationen über das lokale System sammelt, aufbereitet und den anderen Komponenten zur Verfügung stellt. Der Dienst teilt sich in zwei Unterkomponenten, welche im Folgenden genauer vorgestellt werden.

Der lokale Teil

Der lokale Teil des Informationsdienstes sammelt permanent Informationen über die lokale Auslastung des Systems und stellt diese ODIN zur Verfügung. Diese Informationen sind dynamisch und werden zur Laufzeit aus dem Schedule und der lokalen Queue ausgelesen. Dabei handelt es sich um Informationen wie z.B. die Anzahl der belegten CPUs, die Anzahl der Jobs im Schedule oder die letzte Zeit, an der ein Job dem Schedule hinzugefügt wurde. Diese Informationen werden ausschließlich den lokalen Systemkomponenten zur Verfügung gestellt. Aus diesen Informationen berechnet der Informationsdienst laufend Werte wie z.B. die durchschnittliche Anzahl angefragter CPUs, die durchschnittliche Wartezeit eines Jobs, die durchschnittliche geschätzte Rechenzeit eines Jobs und die durchschnittliche tatsächliche Rechenzeit eines Jobs, welche ebenfalls den lokalen Komponenten zur Verfügung stehen.

Der öffentliche Teil

Der öffentliche Teil des Informationsdienstes stellt größtenteils statische Informationen bereit. Diese Informationen beschreiben die lokale MPS. Hier werden zum Beispiel der Name und die IP-Adresse der Site gespeichert. Zusätzlich ist hier noch die Anzahl der zur

Verfügung stehenden CPUs hinterlegt. Des Weiteren verwaltet der Informationsdienst im öffentlichen Teil eine Liste aller durch P2P-Suche gefundenen Grid-Teilnehmer mit ihrem Namen und der dazugehörigen IP-Adresse.
Wird die lokale MPS bei einer P2P-Suche durch eine andere Site gefunden, überträgt sie ihre Liste der „bekannten Welt“ an das neue System.
Eine zweite Funktion des öffentlichen Teils ist es, ODIN eine Liste mit Systemen, die in der Lage sind einen lokal eingereichten Job auszuführen, zur Verfügung zu stellen. Dazu sendet ODIN eine Anfrage an den Informationsdienst, die die Minimalanforderungen des Jobs enthält. Daraufhin filtert der Informationsdienst aus den bekannten Systemen diejenigen heraus, die diese Anforderungen nicht erfüllen können. Die resultierende Liste – die durchaus auch leer sein kann, sofern keine passende Site bekannt ist – wird an den ODIN übermittelt.

Informationsdienst als eigenständige Komponente

Bei der Entscheidung, in welcher Komponente das Sammeln der Informationen über den Schedule und die lokale Jobqueue stattfindet, wurde beschlossen, den `InformationService` als eigenständigen Dienst zu modellieren, anstatt diese Aufgabe der ODIN -Komponente zu übertragen.
Der wesentliche Grund hierfür war, dass dadurch die Komplexität der ODIN -Komponente abnimmt, und diese nicht mit zu vielen Funktion „belastet“ wird. Da der `InformationService` auch Informationen speichert, die nur vom `CommunicationService` benötigt werden, ist, durch die Bereitstellung einer dedizierten Komponente nur für Informationen, nicht nur eine saubere Trennung der Komponenten möglich, sondern auch ein zentraler Anlaufpunkt zur Abfrage von Informationen gegeben. Bei einer möglichen Weiterentwicklung weiß ein Programmierer dadurch sofort, von welcher Komponente Informationen bezogen werden können.

4.3 ODIN – Open Decision Interface Node

ODIN stellt die zentrale Entscheidungskomponente einer MPS dar, die darüber entscheidet, ob ein lokal eingereichter Job lokal ausgeführt werden soll oder zur Ausführung an einen anderen Grid-Teilnehmer geschickt werden soll. Des Weiteren entscheidet ODIN darüber, ob Anfragen von anderen Systemen, einen ihrer Jobs lokal zu berechnen, akzeptiert oder abgelehnt werden. Dazu kann ODIN unterschiedliche Entscheidungsstrategien verfolgen, auf die in Kapitel 5 näher eingegangen wird.
Es kann zwischen zwei Szenarien unterschieden werden: ein Job wird entweder lokal von einem Benutzer eingereicht oder von einem entfernten System.

Ein Job wird lokal eingereicht

Ein Nutzer möchte einen Job ausführen lassen und reicht dazu eine Jobbeschreibung, welche die Anforderungen des Jobs an das System beschreibt, auf dem lokalen System bei

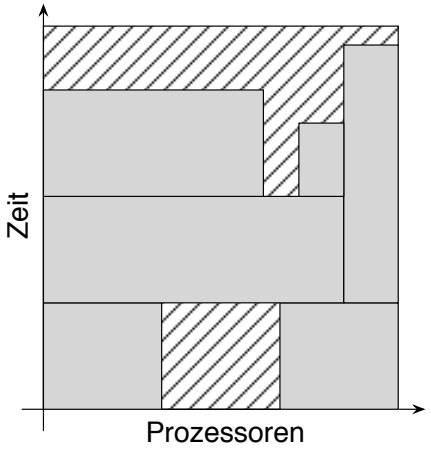


Abbildung 4.2: Beispiels eines Schedules, der durch eine Datenstruktur modelliert werden soll.

ODIN ein. ODIN bildet also den lokalen Zugangspunkt des jeweiligen Systems. Anschließend entscheidet ODIN anhand seiner Entscheidungsstrategie, ob der Job lokal oder auf einer anderen Site ausgeführt werden soll. Im Falle einer Entscheidung zu Gunsten der lokalen Ausführung wird der Job direkt aus dem Eingangspuffer in die lokale Jobqueue verschoben, wo er anschließend vom Scheduling-System weiter verarbeitet wird. Ansonsten wird versucht, den Job an eine andere Site zu verschicken, um ihn dort ausführen zu lassen.

Ein externer Job wird eingereicht

Wenn eine entfernte Site einen Job nicht lokal ausführen möchte, schickt sie Anfragen an andere ihr bekannte Teilnehmer des Grids. Diese Anfragen enthalten eine Jobbeschreibung mit den Kenndaten des betreffenden Jobs, wie z.B. die Anzahl der benötigten CPUs und die geschätzte Ausführungszeit. Diese Anfragen gelangen über den Kommunikationsdienst in das jeweilige System und werden sofort an ODIN weitergeleitet, wo eine Entscheidung getroffen wird, ob der Job akzeptiert oder abgelehnt werden soll. Entscheidet ODIN, den Job zu akzeptieren, so wird eine Bestätigung an den Kommunikationsdienst gesandt, der daraufhin das anfragende System informiert und die Übertragung des Jobs veranlasst. Lehnt ODIN die Anfrage ab, so wird eine Ablehnungsnachricht an den Kommunikationsdienst gesandt, die von diesem ebenfalls an das anfragende System weitergeleitet wird.

4.4 Schedule

Für den Schedule wurde eine Datenstruktur entworfen, die die zur Verwaltung notwendigen Operationen effizient unterstützt. Neben der Unterstützung von schnellen Operationen zum Einfügen, Entfernen und Verschieben von Jobs stand dabei besonders die

Möglichkeit zur effizienten Abfrage verschiedener Informationen zum aktuellen Status des Schedules im Vordergrund, die zum Beispiel vom eingesetzten Scheduling-Algorithmus oder dem Informationsdienst (siehe Abschnitt 4.2) benötigt werden. Die Abbildung 4.2 zeigt ein Beispiel für einen Schedule. Grau gefüllte Flächen repräsentieren Jobs, wobei die Zuordnung zu Prozessoren auf der horizontalen Achse und ihre (geschätzte) Verarbeitungszeit auf der vertikalen Achse aufgetragen ist. Schraffierte Flächen stehen für ungenutzte Kapazitäten im Schedule. Um eine effiziente Abfragen von Status-Informationen zu ermöglichen, werden neben den belegten Blöcken auch die aktuell und zukünftig freien Blöcke von der Schedule-Datenstruktur verwaltet.

4.5 Scheduling-Algorithmus

Scheduling-Algorithmen dienen der Aufteilung der Systemressourcen unter den Jobs, die diese Ressourcen anfordern (siehe Abschnitt 3.2). Die Art der Aufteilung ist dabei stark vom eingesetzten Algorithmus abhängig. Beispiele für Scheduling-Algorithmen sind FCFS und Extensible Argonne Scheduling System (EASY) (vorgestellt in Lifka, 1995). Scheduling-Algorithmen können präemptiv arbeiten, wodurch sie in der Lage sind, laufende Prozesse anzuhalten und ihren Zustand einzufrieren, um sie später weiter auszuführen. Im Rahmen dieser Projektgruppe werden jedoch ausnahmslos nicht präemptive Algorithmen verwendet, d.h. ein einmal gestarteter Prozess kann nicht in seiner Ausführung unterbrochen werden.

4.6 Der Kommunikationsdienst

Der Kommunikationsdienst ist die einzige Verbindung einer Site zur Außenwelt. Er verwaltet die komplette Kommunikation einer Site mit den anderen Grid-Teilnehmern. Beispiele für Kommunikationsaufgaben sind das An- und Abmelden einer Site bei einem bestehenden Grid-Verbund, der Versand und Empfang von Nachrichten jeder Art, sowie die zur Annahme bzw. Ablehnung von Ausführungsanfragen anderer Sites notwendige Kommunikation. Im Folgenden werden die einzelnen Teilaufgaben des Kommunikationsdienstes genauer vorgestellt.

An- und Abmelden einer Site

Um an einem bestehenden Verbund von Sites teilzunehmen, muss sich jede neue Site bei einer anderen Site im Verbund registrieren. Hierbei spielt keine Rolle, um welche der Sites es sich dabei handelt, da keine hierarchische Ordnung existiert. Sobald sich die neue Site registriert hat, werden alle anderen Sites in diesem Verbund über die Existenz der neuen Site informiert. Diese senden daraufhin ebenfalls eine Nachricht an die neue Site und machen sich so mit ihr bekannt. Verlässt eine Site den Verbund oder ist sie für einen längeren Zeitraum nicht mehr erreichbar, so wird diese Site aus der „bekannten Welt“ der anderen Systeme entfernt.

Versand und Empfang von Nachrichten und Jobs

Wenn ODIN (siehe Kapitel 5) für einen lokal eingereichten Job entscheidet, diesen auf einem anderen System ausführen zu lassen, übergibt er den Job zusammen mit einer von in Frage kommenden Systemen an den Kommunikationsdienst. An jede Site aus dieser Liste wird eine Anfrage gestellt, ob der Job dort ausgeführt werden darf. Sollte eine der angefragten Sites den Job akzeptieren wollen, so erhält diese eine Nachricht mit einer Zusage. Anschließend wird der Job an diese Site verschickt. Sollten mehrere positive Rückmeldungen von verschiedenen Sites eintreffen, so wird eine der Sites ausgewählt, wobei hier jeweils die Site mit der ersten eintreffenden positiven Rückmeldung den Zuschlag erhält. An die übrigen Sites wird eine Absage-Nachricht verschickt. Sollte sich keine andere Site bereit erklären, den Job auszuführen, wird dieser an die lokale Jobqueue übergeben und lokal berechnet.

Annehmen und Ablehnen von externen Ausführungsanfragen bzw. Jobs

Die Anfragen zur Jobausführung der anderen Grid-Teilnehmer werden vom Kommunikationsdienst empfangen. Dieser leitet diese Anfragen an ODIN (siehe Kapitel 5) zur Entscheidung weiter. Nachdem ODIN eine Entscheidung getroffen hat, wird diese der anfragenden Site mitgeteilt. Bei einer positiven Entscheidung wird auf eine endgültige Zusage für die Jobausführung, sowie auf den eigentlichen Job gewartet. Trifft der Job ein, wird dieser direkt in die lokale Jobqueue weitergeleitet und anschließend vom Scheduling-System weiter verarbeitet.

5 Entscheidungsstrategien

Inhalt

Eine wichtige Aufgabe während der Entwicklung des Systems war es, Strategien für die Entscheidungskomponente ODIN (Open Decision Interface Node) zu entwerfen, die anhand verschiedener Informationen über die Ausführung lokal bzw. remote eingereichter Jobs entscheiden sollen. Die meisten hierzu genutzten Informationen werden vom Informationsservice zur Verfügung gestellt. Dies sind z.B. Informationen über die Anzahl freier CPUs der lokalen MPS, Anzahl Jobs, die sich in der Queue befinden und noch ausgeführt werden müssen, oder Durchschnittswerte, wie die lokale Wartezeit der letzten Stunden oder die durchschnittliche Ausführungszeit der Jobs. Einige Informationen die zur Entscheidungsfindung benutzt werden, sind in den Jobbeschreibung zu finden, die bei Anfragen an die entfernte MPS verschickt werden. Dies sind z.B. Informationen über die benötigte CPU-Anzahl eines Jobs oder der Name der MPS, von der die Anfrage kommt. Es gibt sowohl konfigurierbare wie nicht konfigurierbare Strategien. Die konfigurierbaren Strategien müssen von Seiten des Benutzers bzgl. bestimmter Parameter eingestellt werden. Auf diese Weise kann versucht werden, die Strategien an die lokalen Gegebenheiten der Site anzupassen. Dies erfordert jedoch auch einen gewissen Erfahrungswert, falls Simulationen mit sinnlosen Einstellungen vermieden werden sollen. Die nicht konfigurierbaren Strategien können ohne Einstellungen von Seiten des Benutzers eingesetzt werden. Diese entscheiden immer nach dem selben Prinzip und verfolgen mit ihrer Strategie ein bestimmtes Ziel.

Vor der Entwicklung der verschiedenen Strategien wurde versucht einige brauchbare Ziele zu formulieren, welche von den Strategien umgesetzt werden sollten. Einige Ziele, die sich als sinnvoll erwiesen haben, sind:

- Bevorzugung von lokalen Benutzern
- Bevorzugung von entfernten Benutzern
- Bevorzugung von Jobs einer bestimmten MPS
- Gute Auslastung des Systems
- Verringerung der Wartezeit für Jobs oder
- Verringerung der Anzahl wartender Jobs

Einzelne oder ein Teil der erwähnten Ziele werden von den im folgenden vorgestellten Strategien verfolgt. Bevor jedoch näher auf die verschiedenen Strategien mit ihren Zielen und Arbeitsweisen eingegangen wird, werden einige der zur Entscheidungsfindung benutzten Werte aus dem Informationsservice vorgestellt.

5.1 Verwendete Werte aus dem Informationsservice

Einige der Strategien arbeiten mit Werten aus dem Informationsservice. Um die Arbeitsweise der Strategien besser zu verstehen, werden drei der Werte kurz vorgestellt.
Die verwendeten Werte stellen einen Durchschnitt der letzten n Minuten dar. Wie weit diese Werte in die Vergangenheit zurück reichen, kann vom Benutzer vorgegeben werden. Durch Anhängen von einem Index $i \in$[1,15] an den abgefragten Wert, wird der Durchschnitt der letzten $i \cdot n$ Minuten abgefragt. Die Größe des Intervalls [1,15] wurde bei der Implementierung des Informationsservices willkürlich gewählt. Es handelt sich um die folgenden Werte:

- Durchschnittlich benötigte CPU-Anzahl der letzten $i \cdot n$ Minuten.
- Durchschnittliche Wartezeit der letzten $i \cdot n$ Minuten.
- Durchschnittliche Ausführungszeit der letzten $i \cdot n$ Minuten.

5.2 ODIN

Hierbei handelt sich um die einfachste Implementierung. Zu Anfang wurde eine einfache Strategie benötigt, die willkürlich Entscheidungen fällt und hauptsächlich zu Testzwecken und zur Evaluierung der Systemfunktionen benutzt wurde. Bei dieser Strategie werden alle Anfragen externer Sites bzgl. der Ausführung eines Jobs positiv beantwortet. Ein lokal eingereichter Job, wird mit einer Wahrscheinlichkeit von $4/5$ ebenfalls lokal ausgeführt und mit einer Wahrscheinlichkeit von $1/5$ wird versucht, diesen auf einer entfernten MPS ausführen zu lassen.

5.3 ODIN_OnlyLocal ($ODIN_{OL}$)

Zum Erzeugen von Referenzwerten für den Betrieb einzelner, nicht vernetzter Sites, wurde eine Klasse namens `ODIN_OnlyLocal` implementiert. Einziges Ziel dieser Strategie ist es, zu verhindern, dass Kommunikationsversuche mit nicht vorhandenen Peers-Sites ungewollte Nebeneffekte erzeugen.
Lokale Jobs werden bei dieser Strategie ausnahmslos lokal berechnet, ohne dass der Versuch unternommen wird, eine andere Site für die Ausführung zu finden.

5.4 ODIN_AcceptWhenFit ($ODIN_{AWF}$)

Diese Strategie liegt in drei Varianten vor. `ODIN_AcceptWhenFit` ($ODIN_{AWF}$), `ODIN_AcceptWhenFitBack` ($ODIN_{AWFB}$) und `ODIN_AcceptWhenFitEgo` ($ODIN_{AWFE}$).
Die beiden Strategien $ODIN_{AWF}$ und $ODIN_{AWFB}$ nehmen sowohl lokale Jobs als auch Jobs von entfernten Sites zur lokalen Berechnung genau dann an, wenn dafür aktuell genügend Ressourcen zur Verfügung stehen. Ist das nicht der Fall, wird bei lokalen Jobs zunächst versucht, diese an andere Sites zu senden (wie üblich wird der Job jedoch trotzdem lokal gerechnet, falls dieses erfolglos ist). Jobs von anderen Sites werden einfach abgelehnt, falls nicht genügend Ressourcen zur Verfügung stehen. Der Unterschied der beiden Strategien, liegt in dem Einfügen der zur lokalen Ausführung akzeptierter Jobs in die Jobqueue. Bei $ODIN_{AWF}$ werden die Jobs ganz vorne in die Jobqueue eingefügt, was zur sofortigen Ausführung und Nutzung vorhandener Ressourcen führt. Da dies einem Schedulingverhalten ähnelt, und die Strategien eigentlich nur zur Entscheidungsfindung und nicht zum Scheduling verwendet werden sollen, wurden parallel zur $ODIN_{AWF}$ Strategie die Strategien $ODIN_{AWFB}$ und $ODIN_{AWFE}$ implementiert. Beim $ODIN_{AWFB}$ wird der Job hinten in die Jobqueue eingefügt und besitzt somit keinen Schedulingcharakter. Das Einfügen der Jobs in die Queue auf diese Art führt dazu, dass der Ausführungszeitpunkt eines angenommenen Jobs nur abhängig vom eingesetzten Schedulingverfahren ist und nicht von der Verfügbarkeit der Ressourcen.
Diese Strategien basieren auf der Annahme, dass der Schedule nicht allzu stark fragmentiert. Wenn zum Zeitpunkt der ODIN-Anfrage ausreichend Ressourcen für einen Job j zur Verfügung standen, ist die Wahrscheinlichkeit hoch, dass sie zum Zeitpunkt, zu dem j gerechnet werden soll, immer noch zur Verfügung stehen. Auf diese Weise könnten sich ansonsten vorhandene Lücken im Schedule verringern bzw. sogar komplett schließen lassen.
Die Strategie $ODIN_{AWFE}$ soll ein egoistisches Verhalten simulieren, bei der alle Anfragen entfernter MPS abgelehnt werden. Lokal eingereichte Jobs werden jedoch genauso behandelt wie bei der Strategie $ODIN_{AWFB}$. Der Einsatz dieser Strategie ermöglicht ein „Ausnutzen“ entfernter MPS. Durch den Einsatz von Strategien auf Basis von Kreditsystemen, welche im Kapitel 5.10.2 vorgestellt werden, kann dies verhindert werden. Bei Simulationen bei denen als Strategie $ODIN_{AWFE}$ und Strategien auf Basis von Kreditsystemen eingesetzt wurden, wurde das Verhalten der MPS mit $ODIN_{AWFE}$ nicht geduldet. MPS mit Strategien auf Basis von Kreditsystemen haben keine Jobs der MPS mit egois-

tischem Verhalten zur lokalen Ausführung angenommen.
Als Maßstab für den Ressourcenbedarf dient bei den drei Strategien lediglich die Anzahl der CPUs, die ein Job anfordert. Dies ist zwar sehr ungenau, es sollte aber auch kein zweiter Scheduler realisiert werden.

5.5 ODIN_KillGaps (ODIN_{KG})

Das Ziel dieser Strategie ist es Lücken im Schedule durch die Annahme bestimmter Jobs zu füllen. Damit sollen die Ressourcen einer Site besser ausgenutzt werden und die Auslastung soll verbessert werden.
Ein externer Job j wird dann abgelehnt, wenn er mehr Prozessoren benötigt als die Differenz zwischen der Anzahl der Prozessoren dieser Site und dem Durchschnitt der Anzahl der Prozessoren von allen Jobs, die diese Site angenommen hat.
Der formale Ausdruck sieht wie folgt aus:

$$(m - \bar{m}) \geq m_j$$

Dabei steht m für die Anzahl der CPUs, die die eigene Site besitzt, $\bar{m}$ für die durchschnittliche Anzahl der angeforderten CPUs von bereits angenommenen Jobs und m_j für die Anzahl der CPUs, die ein Job angefordert hat.
Diese Entscheidungsstrategie versucht einen lokalen Job erst dann extern auszuführen, wenn mehr Prozessoren benötigt werden als gegenwärtig zur Verfügung stehen.
Diese Strategie wurde nicht weiter verfolgt, da die ersten Experimente nicht zufriedenstellende Ergebnisse geliefert haben.

5.6 ODIN_WaitTimeQueueEasy ($\text{ODIN}_{\text{WTQE}}$)

Das Ziel dieser Strategie ist es, die durchschnittliche Wartezeit der Jobs bzw. die Anzahl der wartenden Jobs zu verringern. Durch die Verfolgung dieses Ziels wird implizit die Auslastung des Gesamtsystems verbessert, da die Wartezeit aller Jobs sinkt und die Ausführungszeit aller Jobs gleich bleibt. Durch folgendes Verhalten wird versucht dieses Ziel zu erreichen. Beim Überschreiten einer bestimmten Queue-Größe und/oder bei Überschreitung der durchschnittlichen Wartezeit eines Jobs bzgl. eines konfigurierbaren Werts wird der Job abgelehnt bzw. zur Ausführung an eine entfernte MPS geschickt. Hierbei werden lokal eingereichte Jobs und Jobs entfernter MPS gleich behandelt.
Von Seiten des Benutzers können folgende Parameter konfiguriert werden:

Queue-Größe: Falls die Queue-Größe zum Zeitpunkt der Anfrage diesen Wert übersteigt, so wird die Anfrage abgelehnt bzw. es wird versucht den Job auf einer entfernen MPS ausführen zu lassen. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird die Queuegröße nicht zur Entscheidungsfindung herangezogen. Der Standardwert beträgt 10.

Maximale Wartezeit: Dieser Wert muss in Minuten angegeben werden. Falls die durchschnittliche Wartezeit zum Zeitpunkt der Anfrage diesen Wert übersteigt, so wird die Anfrage abgelehnt bzw. es wird versucht, den Job auf einer entfernen MPS ausführen zu lassen. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird die durchschnittliche Wartezeit der Jobs nicht zur Entscheidungsfindung herangezogen. Der Standardwert beträgt 30 Minuten.

Index zur Werteabfrage: Dieser Parameter bestimmt, welcher Index $i \in [1,15]$ zum laden der Werte aus dem Informationsdienst benutzt werden soll. Der Standardwert beträgt 10. Die Zeitspanne zwischen einem Index i und $i+1$ beträgt jeweils eine Stunde. Somit wird z.B. durch Angabe einer 5 jeweils der Durchschnitt der letzten 5 Stunden aus dem Informationsdienst geholt.

5.7 ODIN_WaitTimeQueueHard ($\text{ODIN}_{\text{WTQH}}$)

Das Ziel dieser Strategie ist, ebenso wie bei $\text{ODIN}_{\text{WTQE}}$, die Verringerung der durchschnittlichen Wartezeit der lokal ausgeführten Jobs bzw. die Verringerung der Anzahl wartender Jobs. Bei den zur Entscheidungsfindung benutzten Werten handelt es sich, wie bei $\text{ODIN}_{\text{WTQE}}$, um die Queuegröße und die durchschnittliche Wartezeit der lokal ausgeführten Jobs. Im Gegenteil zu $\text{ODIN}_{\text{WTQE}}$ wird jedoch bei dieser Strategie zwischen lokal eingereichten Jobs und Jobs entfernter MPS unterschieden. Somit ist es möglich, den lokalen Jobs und Jobs entfernter MPS eine unterschiedliche Priorität zu geben. Dem Benutzer stehen mehr Konfigurationsmöglichkeiten als bei $\text{ODIN}_{\text{WTQE}}$ zur Verfügung. Des Weiteren ist es möglich, die Strategie entscheiden zu lassen, mit welchem Index die Werte aus dem Informationsdienst geholt werden sollen.
Es können die folgenden Parameter konfiguriert werden:

Wartezeit für entfernte Jobs: Dieser Wert muss in Minuten angegeben werden. Falls die durchschnittliche Wartezeit zum Zeitpunkt der Anfrage diesen Wert übersteigt, so wird die Anfrage, einen remote eingereichten Job lokal ausführen zu lassen, abgelehnt. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird die durchschnittliche Wartezeit der Jobs nicht zur Entscheidungsfindung herangezogen. Der Standardwert beträgt 30 Minuten.

Queuegröße für entfernte Jobs: Falls die Queue-Größe zum Zeitpunkt der Anfrage diesen Wert übersteigt, so wird die Anfrage, einen remote eingereichten Job lokal ausführen zu lassen, abgelehnt. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird die Queuegröße nicht zur Entscheidungsfindung herangezogen. Der Defaultwert beträgt 10.

Maximale Wartezeit für lokale Jobs: Dieser Wert muss in Minuten angegeben werden. Falls die durchschnittliche Wartezeit zum Zeitpunkt der Anfrage diesen Wert übersteigt, so wird die Anfrage einen lokal eingereichten Job lokal ausführen zu lassen abgelehnt. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird die

durchschnittliche Wartezeit der Jobs nicht zur Entscheidungsfindung herangezogen. Der Standardwert beträgt 40 Minuten.

Queuegröße für lokale Jobs: Falls die Queue-Größe zum Zeitpunkt der Anfrage diesen Wert übersteigt, so wird die Anfrage, einen lokal eingereichten Job lokal ausführen zu lassen abgelehnt. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird die Queuegröße nicht zur Entscheidungsfindung herangezogen. Der Defaultwert beträgt 10.

Intervallgröße: Zeitspanne in Minuten zwischen zwei Werten aus dem Informationsdienst mit Index i und $i + 1$.

Index zur Werteabfrage: Dieser Parameter bestimmt, welcher Index $i \in$[1,15] zum laden der Werte aus dem Informationsdienst benutzt werden soll. Der Standardwert beträgt 10. Falls der Benutzer diesen Wert mit einer „0“ konfiguriert, so wird der zu verwendete Index i von der Strategie ermittelt. Dazu wird der Index i zu Anfang mit einer 1 initialisiert und bei jeder Anfrage auf die folgende Weise berechnet, wobei $\bar{p}$ die durchschnittliche Rechenzeit aller Jobs der lokalen MPS ist:

$$i = \begin{cases} i + 1, & \text{falls } \bar{p} > i \cdot \text{Intervall-Größe} \\ i - 1, & \text{falls } \bar{p} < i \cdot \text{Intervall-Größe} \end{cases}$$

Die Strategien $\text{ODIN}_{\text{WTQE}}$ und $\text{ODIN}_{\text{WTQH}}$ erfordern ein gewisses Maß an Kenntnissen über die Charakteristik der eingereichten Jobs und wie die Parameter in Bezug auf diese eingestellt werden müssen.

5.8 ODIN_RessourcesOfQueue (ODIN_{ROQ})

Diese Strategie nutzt die „wartende Arbeitslast“ als Entscheidungskriterium. Dem Sitebetreiber soll so ermöglicht werden, die Annahme externer Jobs bzw. die Abgabe lokal eingereichter Jobs davon abhängig zu machen, wie viel Arbeit die Site noch verrichten muss. Bei der Entscheidung eines Jobs wird die Summe aus den Produkten der angefragten CPUs und der angefragten Laufzeit aller Jobs gebildet, die sich in der Queue befinden und noch nicht in den Schedule einsortiert wurden. Die Entscheidung zur lokalen Ausführung eines Jobs hängt dann von einem durch den Nutzer zuvor konfigurierten Schwellwert ab. Übersteigt die derzeitige „wartende Arbeitslast“ diesen Schwellwert, so wird versucht, den Job an eine andere Site abzugeben bzw. ein extern anfragender Job wird abgelehnt. Da das Verhalten bei lokal eingereichten Jobs und extern eingereichten Jobs jedoch unterschiedlich konfiguriert werden kann, müssen zwei Schwellwerte konfiguriert werden. Einer dient der Entscheidungsfindung bei lokal eingereichten Jobs und der andere der bei extern eingereichten Jobs. Standardmäßig sind die beiden Schwellwerte auf 10 für lokal eingereichte Jobs und 7 für extern eingereichte Jobs vorkonfiguriert. Intern werden diese Parameter mit 1000 multipliziert und ergeben somit die Summenschwellwerte 10000 und 7000. Letztenendes ist es jedoch immer vom jeweiligen Setup und den

verwendeten Sites abhängig, welche Einstellungen man nutzen sollte. Jedoch liefert diese Entscheidungsstrategie bei annähernd jedem Setup brauchbare Resultate, wenn sie mit den Standardeinstellungen konfiguriert ist.

5.9 ODIN_ScheduleTime ($ODIN_{ST}$)

Diese Strategie ist eine Erweiterung der $ODIN_{AWF}$-Strategie. Wo die $ODIN_{AWF}$-Strategie einen externen Job nur dann angenommen hat, wenn er sofort auf der Site ausgeführt werden konnte, ist es dem Administrator mit dieser Strategie nun möglich diese Entscheidung davon abhängig zu machen, wann ein Job frühestmöglich ausgeführt werden kann. Zur Konfiguration gibt man jeweils für lokal als auch für extern eingereichte Jobs eine Grenze an. Wenn ein entsprechender Job eintrifft wird überprüft, ob der Zeitpunkt, zu dem der Job bei der derzeitigen Lage in den Schedule einsortiert werden kann, nicht weiter in der Zukunft liegt als die eingestellte Grenze in Sekunden. Wenn man also bei beiden Parametern eine Grenze von Null einstellt, so verhält sich dieser ODIN exakt wie ein $ODIN_{AWF}$. Auch hier gilt, wie für den $ODIN_{ROQ}$, dass sinnige Parameterkonfigurationen von den jeweiligen Sites, auf denen sie eingesetzt werden, abhängen.

5.10 Entscheidungssysteme auf Basis von Kreditsystemen

Neben den bisher erwähnten Strategien wurden zwei Entscheidungssysteme implementiert, die zur Entscheidung, ob ein Job einer entfernten Site lokal ausgeführt werden soll, ein jeweils unterschiedliches Kreditsystem benutzen. Bei beiden Kreditsystemen werden für jede bekannte MPS zwei Werte gespeichert, anhand derer ODIN die Entscheidung fällt.
Als Beispiel aus der Realität dient die Peer-to-Peer-basierte Filesharing-Anwendung BitTorrent (s. Abschnitt 5.10.1). Bei Filesharing-Anwendungen kommt es darauf an, dass alle Parteien nicht nur Daten herunterladen, sondern diese auch selbst zur Verfügung stellen. Das einseitige Herunterladen zerstört den kooperativen Gedanken von Filesharing.
Das Ziel der folgenden Entscheidungstrategien ist es, allen Teilnehmern zu ermöglichen von der Gridteilnahme zu profitieren. D.h. die $AWRT$ der Jobs einer teilnehmenden Site soll verbessert werden. Dadurch soll verhindert werden, dass egoistische Sites andere Sites ausbeuten. Deshalb wird egoistisches Verhalten bestraft, kooperatives Verhalten wird dagegen belohnt.

5.10.1 BitTorrent

Die Funktionsweise von BitTorrent ist in Cohen (2003) beschrieben. Bei BitTorrent gibt es keine zentrale Ressourcenverwaltung. Jeder Peer ist selbst für sich verantwortlich. Ein Peer lädt von jedem Peer herunter, von dem er Daten bekommen kann. Er entscheidet aber mit Hilfe eines Tit-For-Tat-Algorithmus zu wem er Daten hochlädt. Ein Peer ko-

operiert mit anderen Peers, indem er zu ihnen hochlädt und verweigert die Kooperation, indem er andere Peers „abwürgt“ (choke).
Ein guter Choking-Algorithmus soll für die Auslastung der verfügbaren Ressourcen sorgen, für konsistente Downloadraten der Peers und resistent gegen Peers sein, die nur herunter- und nichts hochladen.
So genannte pareto-effiziente Systeme haben die oben genannten Eigenschaften. In so einem System gibt es keine zwei Parteien, die einen Austausch vornehmen und beide besser dastehen können. Der Choking-Algorithmus von BitTorrent versucht Pareto-Effizienz zu erreichen, indem er eine Version des Tit-For-Tat-Algorithmus benutzt, welche auch für das Spiel „Gefangenen-Dilemma“ benutzt wird. Ein Peer lädt zu anderen Peers hoch, die ihrerseits zum Peer hochladen, um mehrere bidirektionale Verbindungen aufzubauen. Versuchweise wird auch zu Peers hochgeladen, die selbst nichts hochladen, mit dem Ziel bessere Downloadraten zu erzielen.
Der Algorithmus für das Erreichen der Pareto-Effizienz in einem System sieht so aus: zwei Parteien, die für ihren Upload im Austausch einen schlechten Upload bekommen, versuchen zu einander hoch zu laden, mit dem Ziel eine bessere Downloadrate bekommen. Das ist also ein lokaler Optimierungsalgorithmus, in dem jeweils zwei Parteien versuchen, ihr Los zu verbessern.
Zu jeder Zeit gibt es eine fixe Anzahl an Peers, zu denen nicht hochgeladen wird (Standardwert ist 4). Welche Peers das sind, hängt von der durchschnittlichen Downloadrate in den letzten 20 Sekunden ab. Alle 10 Sekunden werden die Peers, zu denen hochgeladen wird, neu bestimmt.
Optimistic Unchoking (OU) bedeutet, dass zu jeder Zeit ein Peer existiert, der nicht abgewürgt ist, unabhängig von seiner momentanen Downloadrate. OU dient dazu, Peers zu finden, die für eine bessere Downloadrate sorgen. Das entspricht der Strategie, des immer Kooperierens beim ersten Zug im Spiel „Gefangenen-Dilemma“.
Tit-For-Tat ist eine Strategie für das Spiel „Gefangenen-Dilemma“. Beim ersten Zug wird kooperiert. Beim nächsten wird kooperiert wenn die andere Partei kooperiert hat und Rache geübt, wenn die andere Partei verraten hat.

5.10.2 ODIN_SiteScores (ODIN_{SS})

Lokale Jobs werden genauso behandelt, wie bei der Strategie $\text{ODIN}_{\text{AWFB}}$. D.h. ein lokaler Job wird dann weggeschickt, wenn nicht genug Ressourcen zur Verfügung stehen, um ihn sofort auszuführen.
Bei diesem Kreditsystem berechnet jede Site zwei Werte M_i und D_i. Wir betrachten eine Site j auf der ODIN_{SS} läuft. Dann steht M_i für die Arbeit, die die Site j für die Site i geleistet hat. D_i steht für die Arbeit, die die Site i für die Site j geleistet hat. Um zu entscheiden, ob ein entfernter Job angenommen wird oder nicht, vergleicht die Site j den Wert M_i mit D_i. Falls $M_i \geq D_i$ gilt, nimmt die Site den externen Job an.
Der Entscheider einer Site j speichert die Arbeit, also die Werte M_i und D_i für jede Site i.
Wenn für eine Site i noch keine Punkte gespeichert sind, dann nimmt ODIN_{SS} den ersten Job, der von dieser Site empfangen wird, immer an. Das entspricht auch dem Verhalten

von BitTorrent.
Es ist auch möglich das Kreditsystem mit der Strategie $\text{ODIN}_{\text{AWFB}}$ (siehe Abschnitt 5.4) zu kombinieren. $\text{ODIN}_{\text{AWFB}}$ ist die Strategie, die nur dann einen externen Job annimmt wenn genug Ressourcen verfügbar sind, um ihn sofort auszuführen. Auf diese Weise wird die Wartezeit für externe Jobs minimiert.
Die Konfiguration bei der ODIN_{SS} sowohl die Werte M_i, D_i wie auch die Strategie $\text{ODIN}_{\text{AWFB}}$ berücksichtigt, ist das Standardverhalten von ODIN_{SS}. Es ist auch möglich den Entscheider so zu konfigurieren, dass er nur die Werte M_i und D_i benutzt.
Es kann der folgende Parameter konfiguriert werden:

AWFB Drei unterschiedliche Werte sind möglich:

- **"none"** Es werden nur die Werte M_i und D_i verwendet. Die Strategie AWFB wird nicht verwendet.
- **"and"** Die Strategie AWFB wird durch ein $\wedge$ mit der Strategie von ODIN_{SS} verknüpft. Dies ist die Standardkonfiguration.
- **"or"** Die Strategie AWFB wird durch ein $\vee$ mit der Strategie von ODIN_{SS} verknüpft.

5.10.3 ODIN_SiteScoresRel (ODIN_{SSR})

Bei diesem Kreditsystem wird ein „relativer“ Wert verwendet. Es seien E_j die Anzahl der empfangenen Anfragen einer MPS_j, E_j^+ die Anzahl der angenommenen Jobs einer MPS_j, V_j die Anzahl verschickter Anfragen an eine MPS j und V_j^+ die Anzahl der lokal eingereichten Jobs, die remote auf MPS j ausgeführt wurden. Für jede Site werden nun vom Informationsservice die beiden Werte E_{Remote} und E_{Lokal} zur Verfügung gestellt.

$$E_{\text{Lokal}} = \frac{E_j^+}{E_j}$$

$$E_{\text{Remote}} = \frac{V_j^+}{V_j}$$

Zur Entscheidungsfindung wird das Verhältnis $E_{\text{Remote}}/E_{\text{Lokal}}$ der beiden Werte betrachtet sowie eine Schranke S. Der Wert der Schranke S kann vom Benutzer festgelegt werden. Falls dies nicht geschehen ist, so wird sie standardmäßig auf 1.2 gesetzt. Gilt $(E_{\text{Remote}}/E_{\text{Lokal}}) > S$, so wird die Anfrage, einen Job einer entfernten MPS auszuführen, positiv beantwortet. Ansonsten wird die Anfrage abgelehnt. Bei dieser Strategie werden immer nur die letzten 64 Anfragen einer entfernten MPS betrachtet, und in Relation mit den letzten 64, an diese entfernte MPS gestellten Anfragen gesetzt. Dabei ist nicht der Arbeitsaufwand $m_j \cdot p_j$ entscheidend, sondern die Anzahl der angenommenen bzw. abgelehnten Jobs.
Es können folgende Parameter konfiguriert werden:

Freundesliste: Bei Angabe von true wird eine Textdatei mit Sites verwendet deren Jobs unabhängig von ihrem Score immer angenommen werden.

Grenzwert: Wenn der Score einer Site größer als dieser Wert ist, so wird die Anfrage positiv beantwortet. Ansonsten wird die Anfrage abgelehnt. Der Standardwert beträgt 1.2.

6 Evaluierung

Inhalt

Mit Hilfe der entwickelten Simulationsumgebung für P2P-basierte Computational Grids wurden zahlreiche Experimente mit unterschiedlichen Konfigurationen und Implementierungen von Teilkomponenten des Systems durchgeführt. Um die Leistung der unterschiedlichen Konfigurationen und Implementierungen bewerten und Abläufe innerhalb des Systems nachvollziehen zu können, werden so genannte Metriken benötigt, die verschiedene Charakteristika des Systemverhaltens in einem skalaren Wert erfassen.
Nachfolgend werden die grundlegenden Notationen sowie die verwendeten Metriken eingeführt. Im Anschluß daran wird das generelle Vorgehen hinsichtlich der Evaluierung detailliert erläutert.

6.1 Notationen

Zunächst wird das allgemeine Szenario beschrieben, im Rahmen dessen die Experimente und somit auch die Evaluierung durchgeführt werden; l vernetzte MPS-Systeme arbeiten jeweils lokal eine Menge von Jobs ab. Ein einzelnes MPS k mit m_k Prozessoren kann weiterhin sowohl lokal eingereichte Jobs an andere MPS weiterreichen als auch Jobs von anderen MPS zur lokalen Berechnung annehmen. Die Menge π_k enthält dann all jene Jobs, die von k berechnet worden sind. Die Menge τ_k hingegen beinhaltet alle lokal auf k eingereichten Jobs. Abhängig vom verwendeten Scheduling-Algorithmus resultiert für k der Schedule S_k. Für einen Schedule S_k wird der Zeitpunkt, zu dem ein Job j in diesem Schedule beendet wurde, mit $C_j(S_k)$ bezeichnet. Der Zeitpunkt, zu dem alle Jobs im Schedule beendet worden sind, wird mit $C_{\max,k} = \max_{j \in \pi_k}\{C_j(S_k)\}$ oder *Makespan* bezeichnet.
Um das Austauschverhalten eines MPS k bestimmen zu können, wird die Menge $\Omega_k = \pi_k \backslash \tau_k$ definiert, d.h. die Menge der Jobs, die auf k gerechnet, dort aber nicht lokal eingereicht worden sind. Die Menge Ψ_k beinhaltet all jene Jobs, die von k für die Berechnung auf externen MPSs angefragt wurden.
Zusätzlich zu den Betrachtungen bzgl. des Verhaltens eines einzelnen MPS k werden auch Eigenschaften des Gesamt-Systems untersucht. In diesem Fall werden die oben

vorgestellten Mengen für alle l MPSs verallgemeinert, d.h. konkret werden die folgenden Mengen verwendet:

- $\pi = \bigcup_{k=1}^{l} \pi_k$
- $\tau = \bigcup_{k=1}^{l} \tau_k$
- $\Omega = \bigcup_{k=1}^{l} \Omega_k$

Die Anzahl der insgesamt verfügbaren Prozessoren aller MPSs wird mit m bezeichnet.

6.2 Metriken

Die verwendeten Metriken lassen sich in zwei Klassen aufteilen. Die erste Klasse enthält die Metriken, die qualitativ bewertbar sind. Hierunter fallen z.B. die AWRT und die Auslastung. Bewertbar sind diese Metriken in dem Sinne, dass man bei einer Verkleinerung der AWRT bzw. Vergrößerung der Auslastung von einer Verbesserung sprechen kann. Die zweite Klasse wird von Metriken gebildet, die keiner qualitativen Bewertung unterliegen. Sie dienen als Grundlage für die Untersuchung auftretender Effekte im System. Unter diese Klasse fallen z.B. Austauschmetriken bzgl. Arbeitsleistung oder prozentualem Jobaustausch, ebenso wie Balance-Werte, die Aufschluss darüber geben, welche MPS maßgeblich an der Bewältigung der Gesamtarbeit des Systems sind. Im Folgenden werden die hier eingesetzten Metriken formal vorgestellt.

Die Auslastung (Utilization U) stellt die prozentuale Systemauslastung über die gesamte Laufzeit dar, und kann sowohl lokal für ein einzelnes MPS als auch auch global für einen Verbund mehrerer MPS berechnet werden. Anschaulich wird das Verhältnis der von Jobs belegten Fläche $\sum_{j \in \pi_k} p_j \cdot m_j$ (Squashed Area (SA)) zu der gesamten, vom Schedule definierten Fläche gebildet (vgl. Abbildung 6.1).
Die Auslastung U des Gesamtsystems errechnet sich durch:

$$U = \frac{\sum_{j \in \pi} p_j \cdot m_j}{m \cdot (\max_{j \in \pi}\{C_j(S)\} - \min_{j \in \pi}\{C_j(S) - p_j\})} \cdot 100$$

Die prozentuale Auslastung U_{π_k} (Utilization over π_k) eines MPS errechnet sich durch:

$$U_{\pi_k} = \frac{\sum_{j \in \pi_k} p_j \cdot m_j}{m_k \cdot (\max_{j \in \pi_k}\{C_j(S_k)\} - \min_{j \in \pi_k}\{C_j(S_k) - p_j\})} \cdot 100$$

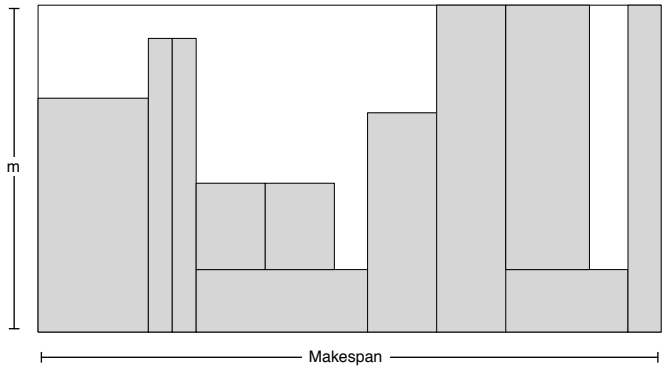


Abbildung 6.1: Illustration der Auslastung U. Graue Flächen repräsentieren Jobs, das äußere Rechteck beschreibt den gesamten Schedule.

Eine der wichtigsten und aussagekräftigsten Metriken für die Bewertung von MPS ist die AWRT. Sie gibt Aufschluss über die durchschnittliche Wartezeit der Jobs in der lokalen Job-Queue. Je höher der ermittelte AWRT-Wert ist, umso länger musste der Job auf seine Ausführung warten. Mit Bezug auf Schwiegelshohn u. Yahyapour (2000) wird ein Job j mit seinem Ressourcenverbrauch $(p_j \cdot m_j)$ gewichtet. Diese Gewichtung stellt sicher, dass weder das Aufteilen von Jobs in mehrere kleine Jobs, noch das Zusammenfassen mehrerer kleiner zu einem großen Job von Vorteil sind.
Die AWRT über alle Jobs in dem System wird gebildet durch:

$$\text{AWRT} = \frac{\sum\limits_{j \in \tau} p_j \cdot m_j \cdot (C_j(S) - r_j)}{\sum\limits_{j \in \tau} p_j \cdot m_j}$$

Die lokale AWRT_{π_k} für einesMPS k über alle Jobs, die tatsächlich auf k gerechnet wurden, wird berechnet durch:

$$\text{AWRT}_{\pi_k} = \frac{\sum\limits_{j \in \pi_k} p_j \cdot m_j \cdot (C_j(S_k) - r_j)}{\sum\limits_{j \in \pi_k} p_j \cdot m_j}$$

Die lokale AWRT_{τ_k} über alle Jobs, die auf k lokal eingereicht wurden, wird berechnet durch:

$$\text{AWRT}_{\tau_k} = \frac{\sum\limits_{j \in \tau_k} p_j \cdot m_j \cdot (C_j(S_j) - r_j)}{\sum\limits_{j \in \tau_k} p_j \cdot m_j}$$

Wie bereits erwähnt, wird eine Maximierung der Auslastung bei gleichzeitiger Minimierung der AWRT angestrebt. Diese Zielsetzung spiegelt den Wunsch eines MPS-Betreibers wider, vorhandene Ressourcen so gut als möglich zu nutzen und gleichzeitig (zahlende) Nutzer des Systems hinsichtlich kurzer Wartezeiten zufrieden zu stellen.

Für einen Verbund mehrerer MPSs lassen sich die lokalen Auslastungswerte nur dann miteinander vergleichen, falls die Makespans der einzelnen Systeme näherungsweise gleich sind. Ist lediglich die tatsächliche Änderung der Auslastung $\Delta\mathrm{SA}_k$, falls Jobs zwischen mehreren MPS-Systemen ausgetauscht werden, von Interesse, so kann der Makespan vernachlässigt werden. Die Änderung berechnet sich aus dem Verhältnis aus der SA der lokal gerechneten Jobs zu den lokal eingereichten Jobs. Wurden keine Jobs abgegeben und auch keine externen Jobs angenommen, so beträgt der Wert genau 100. Formal definiert sich die $\Delta\mathrm{SA}_k$ wie folgt:

$$\Delta\mathrm{SA}_k = \frac{\sum\limits_{j \in \pi_k} p_j \cdot m_j}{\sum\limits_{j \in \tau_k} p_j \cdot m_j} \cdot 100$$

Die bisher vorgestellten Leistungsindikatoren reichen nicht aus, um das globale Verhalten des Systems sowie den Austausch von Jobs ausreichend bewerten bzw. verstehen zu können. Aus diesem Grund wurden weitere Metriken entworfen, die die Untersuchung globaler Effekte des Systems erleichtern, und im folgenden genauer vorgestellt werden.
Die erste der neuen Metriken ist die so genannte Balance. Sie basiert auf der Vermutung, dass der Anteil der Arbeit eines MPS an der verrichteten Gesamtarbeit ungefähr gleich ihrem Potenzial, also der Anteil eigener Ressourcen an den Gesamtressourcen, ist. Aus diesem Grund wird zunächst das Verhältnis zwischen der eigenen Arbeit eines MPS zur Gesamtarbeit errechnet. Anschließend wird dieser Wert dem Potenzial gegenübergestellt. Das Potenzial wiederum ist der Quotient aus eigener Prozessorenzahl dividiert durch die Anzahl der insgesamt verfügbaren Prozessoren m des Systems. Die Ausgeglichenheit (Balance) eines MPS k berechnet sich durch:

$$\mathcal{B}_k = \frac{\sum\limits_{j \in \pi_k} p_j \cdot m_j}{\sum\limits_{k'=1}^{l} \sum\limits_{j \in \pi_{k'}} p_j \cdot m_j} \cdot \frac{m}{m_k}$$

Die Balance an sich ist jedoch nicht ausreichend. Um zu berücksichtigen, ob ein MPS sein „Arbeitspensum“ in kürzerer oder längerer Zeit als das “langsamste” MPS durchgeführt hat, wird der errechnete Balance-Wert mit einem Zeitfaktor multipliziert, der aus dem Verhältnis der $C_{\max,k}$ von k zu der $\max\{C_{\max,k'} | k' \in \{1, \ldots, l\}\})$ aller MPS besteht. Die Ausgeglichenheit (Balance) von k mit korrigierendem Zeitfaktor berechnet sich dann durch:

$$\mathcal{B}_k^T = \mathcal{B}_k \cdot \frac{C_{\max,k}}{\max\limits_{k' \in \{1,\ldots,l\}} \{C_{\max,k'}\}}$$

Um nachzuvollziehen, in wie weit die Austauschstrategien Einfluss auf das Annahme- und Ablehnverhalten bzgl. Jobs fremder MPS haben, wurde die nächste Metrik entworfen. Anhand ihrer kann das generelle Akzeptanzverhalten der einzelnen MPS beurteilt werden. Der prozentuale Anteil an erfolgreich extern angefragten Jobs wird berechnet durch:

$$\mathcal{M} = \frac{|\Omega|}{|\Psi|} \cdot 100$$

Die oben beschriebene Metrik bezieht sich auf das globale Verhalten des MPS-Verbundes. Um das Austauschverhalten einzelner MPS untereinander besser beurteilen zu können, wird diese Metrik auch für je zwei MPS berechnet. Die Werte des gesamten Systems können dann in einer Matrix dargestellt werden. Tabelle A.28 zeigt ein Beispiel für eine Austauschmatrix für drei MPS.
Weiterhin werden unterschiedliche Eigenschaften ausgetauschter Jobs mit Hilfe der folgenden Kenngrößen erfasst. Der Prozesszeitbedarf ausgetauschter Jobs wird berechnet durch:

$$\mathcal{M}^T = \sum_{j \in \Omega} p_j$$

Der Prozessorenbedarf ausgetauschter Jobs wird berechnet durch:

$$\mathcal{M}^C = \sum_{j \in \Omega} m_j$$

Sowohl für $\mathcal{M}^T$ als auch für $\mathcal{M}^C$ erlaubt die Betrachtung der jeweiligen Mittelwerte Rückschlüsse auf die Größe der ausgetauschten Jobs.
Der Ressourcenbedarf eines Jobs ergibt sich aus dem Produkt von tatsächlich benötigter Laufzeit und genutzter Prozessoren. Der Ressourcenbedarf aller ausgetauschter Jobs wird dementsprechend wie folgt berechnet:

$$\mathcal{M}^R = \sum_{j \in \Omega} p_j \cdot m_j$$

Auch über diese Metrik ist ein Mittelwert praktikabel.

6.3 Ablauf der Evaluierung

In diesem Abschnitt wird der Ablauf der Evaluierung der experimentellen Resultate erläutert. Nach der zunächst informellen Spezifikation eines experimentellen Setups, wird die Konfiguration der Simulation mit Hilfe einer eXtensible Markup Language (XML)-Datei vorgenommen. Formell müssen dabei die einzelnen MPS und ihre Komponenten spezifiziert werden. Weiterhin wird innerhalb der Definition für jedes der MPS ein Workload-Trace definiert, der im Rahmen des Experiments als Grundlage der Simulation dient.
In der aktuellen Konfiguration wird auf die von Feitelson zur Verfügung gestellten Workload-Archive zurückgegriffen (vgl. Abschnitt 3.2). Aufgrund von Unregelmäßigkeiten während der Aufzeichnung der Archive finden sich ungültige Jobs in diesen Daten, die in einem Vorverarbeitungs-Schritt herausgefiltert werden, wie in Abschnitt 7.1 beschrieben.
Anschließend wird der Simulator mit dieser Konfiguration ausgeführt. Als Ergebnis wird wiederum ein Workload-Trace des Experiments erzeugt, der die Basis für die weitere Auswertung darstellt. Um die Auswertung des Job-Austausches in einem globalen Kontext zu ermöglichen, wurde das von Chapin et al. (1999a) vorgestellte Archiv-Format um

eine Liste von MPS-Namen erweitert. Der erste Eintrag der Liste stellt dabei das MPS dar, an der ein Job ursprünglich eingegeben wurde. Der letzte Eintrag der Liste ist der Name des MPS, auf der ein Job schlussendlich ausgeführt wurde. Alle weiteren Einträge repräsentieren abgelehnte Anfragen an die entsprechenden MPS.
Die Evaluierungs-Komponente benötigt als Eingabe die Ergebnistrace-Rohdaten sowie die Konfiguration des Experiments, und berechnet die vorab eingeführten Metriken, um einerseits die Leistung der einzelnen MPS bewerten zu können, andererseits aber auch, um Rückschlüsse über die Vorgänge während des Experiments ziehen zu können. Grob lassen sich dabei die ermittelten Kennwerte in globale und lokale Metriken unterteilen, d.h. Metriken pro Multi-Processor-System und Metriken, die das Verhalten und die Leistung des globalen Rechnerverbundes beschreiben.
Abbildung 6.2 stellt schematisch den oben beschriebenen Ablauf für ein experimentelles Setup mit drei MPS dar.

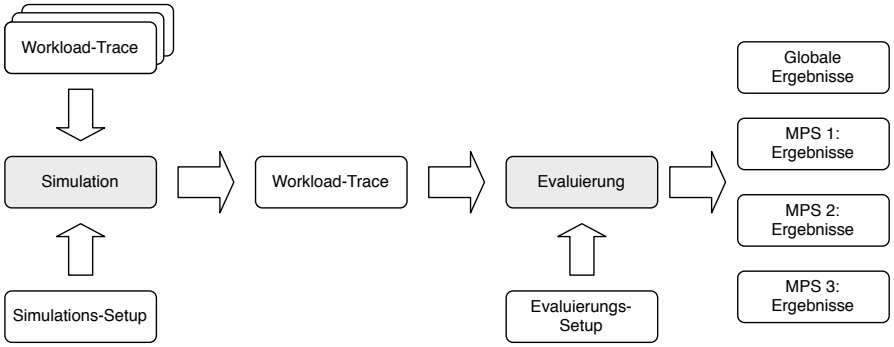


Abbildung 6.2: Schematische Darstellung der Evaluierung

7 Durchgeführte Experimente

Inhalt

In diesem Kapitel wird auf einige von der PG durchgeführte Experimente eingegangen. In einem ersten Experiment haben wir eine Gridumgebung mit zwei Sites nachgebaut, vergleiche (Grimme, Lepping, u. Papaspyrou, 2007). Im Gegensatz zum darin vorkommenden zentralen Jobpool nutzen wir zum Austausch der Jobs einen P2P-Ansatz (siehe Kapitel 3.1). Die Unterschiede zwischen den beiden Ansätzen werden vorgestellt und ein Erklärungsversuch bzgl. der unterschiedlichen Resultate vorgenommen. Das zweite Experiment beschäftigt sich mit der Evaluierung einseitigen egoistischen Verhaltens einer Site in einem Multi-Site-Modell und dessen Auswirkungen. Das dritte Experiment befasst sich mit dem Vergleich der in Kapitel 5 beschriebenen Entscheidungsstrategien. Abschließend beschreibt das letzte Experiment einen Versuch, den Verlust an Effektivität

durch das egoistische Verhalten einer einzelnen Site auszugleichen, indem eine spezielle Entscheidungsstrategie angewandt wird[1].

7.1 Verwendete Eingabedaten

Als Eingabedaten wurden Workload Traces aus dem Parallel Workloads Archive[2] genommen. Diese Daten befinden sich, wie bereits in Kapitel 3.2 beschrieben, im SWF. Für die Experimente wurden nicht alle auf der Webseite zur Verfügung stehenden Traces benutzt. Bei der Auswahl der Traces sollten sowohl kleine Maschinen, als auch große Maschinen berücksichtigt werden, um den Einfluss zwischen solchen in einer Gridumgebung untersuchen zu können. Die Wahl fiel auf die in Tabelle 7.1 aufgeführten Traces der Kungliga Tekniska högskolan (KTH)[3], des Cornell Theory Center (CTC) und des San Diego Supercomputer Center (SDSC). 'Tracename' ist hierbei der im Feitelson-Archiv verwendete Name. Ein Zahlenzusatz bei Bezeichnungen der Workload-Traces im diesem Bericht gibt die Dauer in Monaten an, die von dem Workload verwendet wurde.

Tracename	Name im Endbericht	CPU-Anzahl
KTH SP2	KTH	100
CTC SP2	CTC	430
SDSC SP2	SDSC00	128
SDSC BLUE	SDSC03	1152
SDSC DataStar	SDSC05	1664

Tabelle 7.1: Die bei den Experimenten verwendeten Traces.

Kürzen der Workload-Traces

Die für die Experimente verwendeten Workload-Daten sind zuvor auf eine zeitliche Länge von elf Monaten zurechtgeschnitten worden. Um aussagefähige Ergebnisse zu erhalten, ist es wichtig, dass alle simulierten Systeme für die Dauer der Simulation mit Rechenjobs versorgt werden können. Da die kürzeste Dauer eines der in den Experimenten verwendeten Traces elf Monate beträgt, sind alle längeren Traces auf eine Dauer von elf Monaten gekürzt worden. Dadurch ist es möglich, eine maximale Anzahl von Vergleichen mit unterschiedlichen Workload-Daten durchzuführen.

Preprocessing von Jobs

Die auf Basis der Workload-Daten generierten Rechenjobs werden vor dem Eintritt in das System gefiltert bzw. bei eindeutigen Fehlern auch korrigiert.

[1] Für eine Übersicht über alle Ergebnisse der Versuche sei an dieser Stelle auf Kapitel A.4 im Anhang verwiesen

[2] http://www.cs.huji.ac.il/labs/parallel/workload/

[3] Royal Institute of Technology in Stockholm, Schweden

Falls ein Workload-Datensatz weniger als 18 Felder mit Werten enthält, wird dieser verworfen, da ansonsten die Werte nicht mehr den einzelnen im SWF definierten Spalten zuzuordnen sind. Es ist davon auszugehen, dass der Datensatz fehlerhaft ist, weswegen er verworfen werden muss. Auch gelangen keine Jobs ins System, bei denen entweder die angeforderte („requested runtime“) oder die tatsächliche Ausführungszeit („runtime“) kleiner oder gleich Null ist. Tritt dieses bei einem Datensatz auf, ist offensichtlich, dass der Job nicht ausgeführt worden ist. Da nicht ausgeführte Jobs keinen Einfluss auf die Simulationsergebnisse haben (aber die Leistungsfähigkeit des Simulators beeinträchtigen), können sie vom System ignoriert werden. Datensätze von Jobs, die mehr Prozessoren anfordern als das System aufweist oder die keine positive Anzahl von Prozessoren zur Ausführung zugewiesen bekommen haben, werden ebenfalls verworfen.
Anschließend können Datensätze mit unterschiedlichen Werten für zugewiesene CPUs („allocated CPUs“) und angeforderte CPUs („requested CPUs“) angepasst werden. Dabei gelten folgende Regeln:

- Hat ein Job Null oder weniger CPUs zugewiesen bekommen und eine positive Anzahl von CPUs angefordert, so wird die Zahl der zugewiesenen CPUs auf die Zahl der angeforderten CPUs gesetzt. Hierbei ist zu beachten, dass es auch Jobs mit diesen Eigenschaften geben kann, die gar nicht erst ausgeführt worden sind. Solche Jobs werden aber zuvor durch den oben beschriebenen Ausführungszeit-Filter ausgefiltert und werden an dieser Stelle bereits nicht mehr betrachtet.

- Hat ein Job Null oder weniger CPUs angefordert und eine positive Anzahl von CPUs zugewiesen bekommen, so wird die Anzahl der angeforderten CPUs auf die Anzahl der zugewiesenen CPUs gesetzt. Dies ist legitim, da es auch Systeme gibt, die vom Benutzer keine explizite Anforderung einer bestimmten Anzahl CPUs erfordern.

- In allen anderen Fällen wird die Anzahl der angeforderten CPUs auf die Anzahl der zugewiesenen CPUs gesetzt. Es ergibt keinen Sinn, in der Simulation mit einem anderen Wert als der einem Job tatsächlich zugewiesenen CPU-Anzahl zu arbeiten, da die Workload-Traces dafür keine Daten bereitstellen.

Zu guter Letzt wird der Wert für die Ausführungszeit eines Jobs auf den kleineren der beiden Werte der tatsächlichen Ausführungszeit und der angeforderten Ausführungszeit gesetzt. Dies geschieht basierend auf der Annahme, dass das System bei Überschreitung der angeforderten Rechenzeit die Bearbeitung des Jobs abbricht.

7.2 Vergleich des P2P-Ansatzes mit der Verwendung eines zentralen Job-Pools

In diesem Kapitel wird der von der PG entworfene P2P-Ansatz zum Austausch von Jobs abgegrenzt von einem Ansatz, der auf der Nutzung eines zentralen Job-Pools für den Jobaustausch basiert. Hierzu werden einzelne Resultate aus den Simulationen genannt

und Ergebnissen eines Ansatzes mit zentralem Job-Pool(Grimme et al., 2007) gegenübergestellt. Im letzten Abschnitt wird auf die unterschiedlichen Verbesserungen der beiden Ansätze bzgl. der AWRT und deren mögliche Ursachen eingegangen. Der Vergleich soll Erkenntnisse darüber liefern, in wie weit sich die beiden Ansätze in ihrer Leistungsfähigkeit unterscheiden.

7.2.1 Umsetzung des Experiments mit der P2P-Lösung

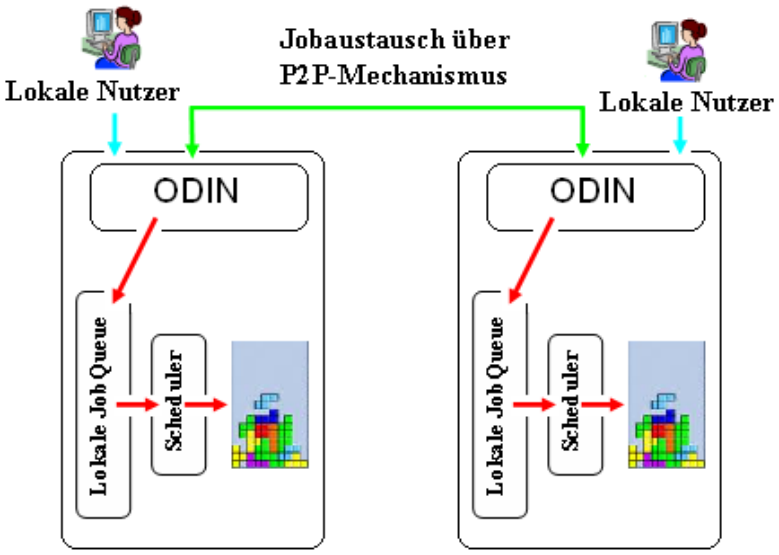


Abbildung 7.1: Aufbau der P2P-Variante

Mithilfe unseres Frameworks wurden mehrere Teilversuche der Form wie in Abbildung 7.1 zu erkennen aufgebaut. Sie kombinieren jeweils zwei Sites mit unterschiedlichen Traces (KTH, CTC, SDSC00, SDSC03, SDSC05). Der grundlegende lokale Scheduling-Algorithmus ist EASY-Backfilling. Im Anschluss daran wird jeder Teilversuch mit Jobaustausch seinem Referenzversuch ohne Jobaustausch gegenüber gestellt. Besonders wichtig ist die Betrachtung der Veränderungen der AWRT und der Auslastung sowohl lokal für die einzelnen Sites als auch global für einen gesamten Teilversuch. Denn für jeden Sitebetreiber ist es von Interesse, welche Verbesserungen er durch seine Teilnahme am Grid erwarten kann. Die Güte des Modells und der mögliche Grad an Verbesserung lässt sich hingegen über die globalen Metriken ermitteln.
Die Resultate haben gezeigt, dass die Teilnahme einer Site am Grid große Verbesserungen mit sich bringen kann. Besonders stark ausgeprägt sind diese Verbesserungen bei kleinen Sites (9% - 32% Verbesserung der AWRT beim KTH und 20% - 62% Verbesserung der AWRT beim SDSC00). Größere Sites profitieren je nach Partner im Grid ebenfalls von der Teilnahme. Die größte Site SDSC05 kann sich bzgl. ihrer AWRT beispielsweise um 0,3% verschlechtern, wenn sie mit einer kleinen Site zusammenarbeitet jedoch auch um 7% verbessern, wenn sie Jobs mit dem größeren SDSC03 austauscht.

7.2.2 Unterschiede zwischen beiden Ansätzen und deren mögliche Ursachen

Im direkten Vergleich zwischen beiden Ansätzen hat sich gezeigt, dass die Job-Pool-Lösung bzgl. der AWRT ein stärkeres Verbesserungspotential gegenüber der lokalen Ausführung besitzt als die P2P-Lösung. Die Verbesserungen der Teilversuche des P2P-Ansatzes sind im Durchschnitt zwei bis drei Prozentpunkte geringer als beim Job-Pool-Ansatz. Für diese Unterschiede gibt es mehrere Erklärungsansätze. Zum Einen können die Scheduler der beiden Sites im Job-Pool-Ansatz einen Schedule auf Basis von zwei Queues aufbauen (lokale Queue und Job-Pool). Diese Möglichkeit erlaubt es ihnen, effizientere Entscheidungen zu treffen. Ein Scheduler des P2P-Ansatzes hingegen arbeitet nur auf seiner lokalen Queue. Darüber hinaus können Jobs, die einmal in einer lokalen Queue eingefügt wurden, egal ob es sich nun um lokal eingereichte Jobs oder um angenommene Jobs externer Sites handelt, diese nicht mehr verlassen. Die Entscheidung, in welcher lokalen Queue und damit auch auf welcher Site ein Job gerechnet wird, findet schon bei Eintritt des Jobs in das System statt. Sie wird von den vorgelagerten Entscheidungskomponenten (ODIN) getroffen und ist im nachhinein nicht mehr änderbar. Die Idee hinter diesem Konzept ist, das lokale Scheduling-System einer Site zu kapseln und die Austauschstrategie als zusätzliche Schicht für den Jobaustausch zwischen den Sites hinzuzufügen.
Ein Job, der in der Job-Pool-Lösung aufgrund mangelnder Ressourcen in den Job-Pool eingereiht wird, kann zu einem späteren Zeitpunkt durchaus von der gleichen Site wieder aus dem Pool entnommen werden, sobald er dort in den Schedule aufgenommen werden kann - anders als ein Job, der bei der P2P-Lösung seinen Weg über die Austauschstrategien in die lokale Queue einer Site - egal ob die ursprüngliche oder eine fremde - gefunden hat. Ein großer Nachteil der Job-Pool-Lösung ist jedoch die mangelnde Skalierbarkeit, die ein System mit einer zentralen Komponente wie dem Job-Pool mit sich bringt. Interessant wäre die Kombination beider Ansätze mit Verwendung des Job-Pool-Ansatzes in überschaubaren Domänen und peerbasiertem Austausch von Jobs zwischen den Domänen.

7.3 Evaluierung egoistischen Verhaltens

Im Vordergrund dieser Versuchsreihe steht die Frage der Stabilität des erstellten Modells, also seine Tolleranz gegenüber Ausfällen und anormalem Verhalten. In diesem Experiment besteht das anormale Verhalten darin, dass eine der teilnehmenden Parteien eine egoistische Strategie verfolgt und sich nicht fair an dem Jobaustausch beteiligt. Während die anderen Sites neben der Abgabe von lokal eingereichten Jobs auch Jobs der übrigen annehmen, lehnt diese Site Jobs anderer Sites grundsätzlich ab. Diese Situation würde im praktischen Einsatz eines Grids, durch die egoistische Konfiguration einer Site durch den Administrator geschehen. Als Grundlage dienen drei Versuche, die aufeinander aufbauen. Im ersten Teilversuch besteht das simulierte System aus drei Sites mit der jeweils

geringsten CPU-Anzahl[4] über elf Monate. Im zweiten Teilversuch wird dasselbe Experiment mit den drei größten Sites durchgeführt. Im Anschluss daran wird ein System von fünf Sites in Hinblick auf egoistisches Verhalten untersucht.

7.3.1 Egoistisches Verhalten in einem Grid bestehend aus drei kleinen Sites

In diesem Versuch wählen wir zunächst nur die drei kleinsten Traces als Grundlage, um den Simulationsaufwand in Grenzen zu halten und grundlegende Verhaltensweisen des Systems bei einseitigem lokalen Verhalten zu ermitteln. Für die Auswahl der drei größten Traces oder die Verwendung von fünf Traces auf fünf verschiedenen Sites sei auf die beiden nachfolgenden Versuche, die in Kapitel 7.3.2 und 7.3.3 beschrieben sind, verwiesen.
Unser untersuchtes System besteht aus drei Sites mit unterschiedlichen Traces (KTH, SDSC00 und CTC). Der Scheduling-Algorithmus für das lokale Scheduling ist für alle Sites EASY-Backfilling, da es sich hierbei um einen Algorithmus handelt, der auch in einer Single-Site-Umgebung zu kurzen AWRTs führt und in der Praxis sehr weit verbreitet ist. Als Grundlage für den Versuch dient eine Simulation, bei der auf jeder Site die Entscheidungsstrategie ODIN_{OL} (vgl. Kapitel 5.3) eingesetzt wird. Dadurch lassen sich in einem Setup alle drei Single-Site-Referenzversuche parallel durchführen, da in diesem System kein Jobaustausch stattfindet. In einem weiteren Referenzversuch werden dann alle Entscheidungsstrategien gegen eine möglichst einfache Variante ausgetauscht, die den Jobaustausch zwischen den einzelnen Sites ermöglicht. In dieser Versuchsreihe handelt es sich um die Strategie $\text{ODIN}_{\text{AWFB}}$ (vgl. Kapitel 5.4). Die Ergebnisse des Teilversuchs geben uns Aufschluss darüber, welche Verbesserungen bzgl. AWRT und Auslastung in unserem System durch den Austausch von Jobs zu erwarten sind. Die drei nach den Referenzversuchen durchgeführten Teilversuche dienen der Überprüfung, ob das egoistische Verhalten einer einzelnen Site die Verbesserungen gegenüber der lokalen Ausführung einschränkt oder zu Verschlechterungen führen kann.
In den dargestellten Balkendiagrammen geben die Bezeichnungen der einzelnen Sites in der Legende Aufschluss über das verwendete Setup, den zugrundeliegenden Trace, den verwendeten lokalen Schedulingalgorithmus und die eingesetzte Austauschstrategie. Dargestellt sind die prozentualen Unterschiede zweier Setups bezüglich lokaler Auslastung, lokaler AWRT, globaler Auslastung und globaler AWRT. Nach Durchführung der Referenz- und Egoismus-Versuche sind folgende Ergebnisse hervorzuheben:

- Der Austausch von Jobs führt erwartungsgemäß zur Verbesserung[5] der Antwortzeiten einzelner Sites und auch der globalen AWRT zwischen den beiden Referenz-Setups (13% Verbesserung). Besonders bemerkenswert sind diese Verbesserungen z.B. bei kleinen Sites wie dem KTH (22%) und dem SDSC00 (56%) wie auch in Abbildung 7.2 zu sehen.

[4] Die Größe einer Site bemisst sich im Folgenden durch die Anzahl ihrer Ressourcen - der CTC mit 430 CPUs ist größer als der KTH mit 100 CPUs

[5] Wird in den folgenden Experimenten ohne Angabe einer Metrik von Verbesserungen gesprochen, so beziehen sich diese auf die Verringerung der AWRT

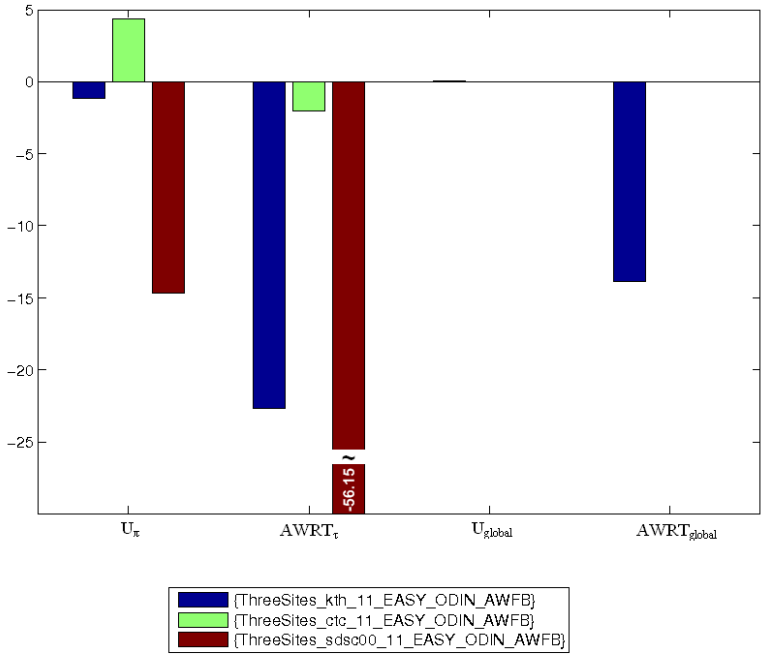


Abbildung 7.2: Verbesserungen durch Jobaustausch

- Egoistisches Verhalten führt bei diesem Teilexperiment generell zu einer Verschlechterung des Gesamtsystems gegenüber dem Referenzsystem mit Jobaustausch. Abhängig von der Größe der egoistischen Sites beläuft sich diese Verschlechterung auf 1,8% beim egoistisch konfigurierten KTH bis 13% beim CTC mit egoistischem Verhalten. Dennoch sind die Systeme mit einer egoistischen Partei immernoch durchweg besser als der Referenzversuch ohne Jobaustausch.

- Bei subjektiver Betrachtung der Resultate, also aus der Sicht einzelner Sites, fällt auf, dass egoistisches Verhalten kleiner Sites wie SDSC00 und KTH nur zu geringen Verschlechterungen großer Sites führt. Für den größeren CTC liegen die Verschlechterungen zwischen 2,6% gegenüber fairem Jobaustauschverhalten wenn der KTH egoistisch handelt und 2,9% bei einem egoistischen SDSC00. Egoistisches Verhalten des großen CTC wiederum schlägt sich in erheblichen Verschlechterungen der AWRT auf den beiden kleinen Sites nieder, wie in Abbildung 7.3 zu sehen. Dies lässt sich über den zuvor auch von Grimme et al. (2007) entdeckten Zusammenhang erklären, dass kleine Sites einen höheren Prozentsatz von Jobs an größere Sites in dem System abtreten als umgekehrt. Wenn eine große Site bei fairem Austausch noch viele Jobs der kleineren Sites übernommen hat und auf einmal egoistisch handelt, dann müssen die kleineren Sites erheblich einbüßen, da sie auf der einen Seite ihre Jobs nicht mehr an die egoistische Site abtreten können, im Gegenzug dazu jedoch noch Jobs von dieser übernehmen, wie in Tabelle A.3 im Anhang zu erkennen ist.

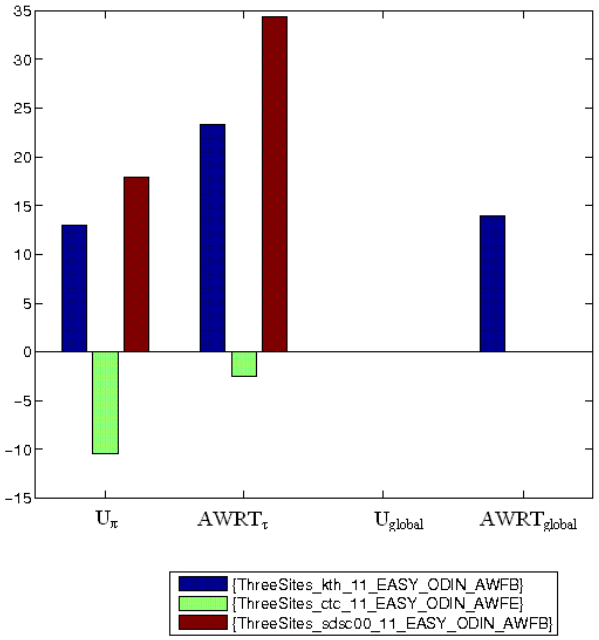


Abbildung 7.3: Verschlechterung durch egoistischen CTC gegenüber fairem Jobaustausch

- Wenn auch das Gesamtsystem bei egoistischem Verhalten einer einzelnen Site noch vom Jobaustausch profitiert (1%-13% Verbesserung der AWRT gegenüber der lokalen Ausführung), so können sich einzelne Sites bzgl. ihrer Antwortzeiten gegenüber dem Einsatz in einer Single-Site-Umgebung verschlechtern. Vor allem der große CTC verliert, bei egoistischem Verhalten einer der beiden kleineren Sites, seinen Vorteil durch Teilnahme am Grid und verschlechtert sich gegenüber dem Single-Site-Fall. Diese Verschlechterungen belaufen sich auf 0,7% in Zusammenspiel mit einem egoistischen KTH und 1% mit einem egoistischen SDSC00. Somit ist nachgewiesen, dass egoistisches Verhalten einer Site dazu führen kann, dass sich die Teilnahme am Grid für mindestens eine der Sites nicht lohnt.

7.3.2 Egoistisches Verhalten in einem Grid bestehend aus drei großen Sites

Für diesen Versuch wählen wir die drei größten Sites als Grundlage (CTC, SDSC03 und SDSC05). Hauptsächlich soll überprüft werden, in wie weit sich die Resultate aus dem vorherigen Versuch (vgl. Kapitel 7.3.1) auf das Szenario mit drei großen Sites übertragen lassen. Aus diesem Grunde ist der restliche Aufbau des Versuchs identisch mit dem vorherigen Versuch auf drei kleinen Sites. Es wird also wieder EASY-Backfilling als lokaler Scheduling-Algorithmus ausgewählt, $ODIN_{AWFB}$ als Standard-Strategie für den Jobaus-

tausch verwendet und $ODIN_{OL}$ als Grundlage für den Referenz-Teilversuch genutzt. Die im Anschluss an die Simulation durchgeführte Prüfung aller im vorherigen Versuch gemachten Aussagen auf ihre Gültigkeit, ergibt folgendes:

- Erneut führt die Teilnahme am Grid bei fairem Jobaustausch für alle Sites zu Verbesserungen gegenüber ihrer lokalen Ausführung. Auch hier gilt, dass die Größe einer Site Einfluss auf den Grad der Verbesserung nimmt. Kleine Sites wie der CTC profitieren wie in Abbildung 7.4 zu sehen stark von der Teilnahme am Grid (22%). Beim SDSC03 sind es immerhin noch 18%, wohingegen der SDSC05 lediglich auf ca. 8% Verbesserung verweisen kann. Mit 8% bis 22% liegen die Verbesserungen jedoch weit weniger auseinander als bei dem Versuch mit den drei kleinen Sites (2% bis 56 %).

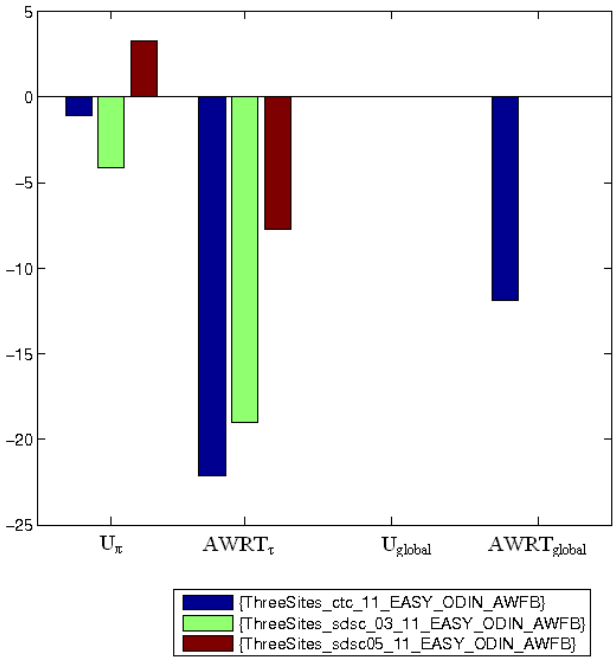


Abbildung 7.4: Verbesserungen durch Jobaustausch gegenüber lokaler Ausführung

- Auch in diesem Versuch führt egoistisches Verhalten einer der Parteien zur generellen Verschlechterung des Gesamtsystems gegenüber dem Versuch mit fairem Jobaustausch. Insgesamt vergrößert sich die AWRT um 2% bei einer kleinen egoistischen Site bis 8% bei einer Großen.

- Die Tendenz, dass egoistisches Verhalten großer Sites zwar zur Verbesserung der Ergebnisse dieser Site aber zu einer weitaus größeren Verschlechterung der Ergebnisse von kleineren Sites führt, lässt sich auch in diesem Versuch wiederfinden. Jedoch

werden die Auswirkungen der egoistischen Strategien in diesem Versuch hauptsächlich von der jeweils anderen großen Site getragen. Im Falle des egoistischen SDSC03 wird die Arbeitslast vom SDSC05 übernommen und beim egoistischen SDSC05 vom SDSC03. Im Gegensatz zum vorherigen Teilversuch existieren dieses Mal zwei große und eine verhältnismäßig kleine Site und nicht umgekehrt und somit gibt es nun auch immer eine größere Site, welche die Arbeitslast übernehmen kann.

- Ein nicht übertragbares Resultat ist, dass bei egoistischem Verhalten einer Site mindestens eine andere Site auch gegenüber ihrer lokalen Ausführung verliert. Egoistisches Verhalten einer der beiden großen Sites führt zwar auf der jeweils anderen großen Site zum genannten Effekt, doch das egoistische Verhalten eines kleinen CTC wird von den beiden anderen Sites so abgefangen, dass diese beide noch von der Teilnahme an dem Grid profitieren.

7.3.3 Egoistisches Verhalten in einem Grid bestehend aus fünf Sites

Im letzten Versuch der Versuchsreihe zur Untersuchung von einseitigem egoistischen Verhalten simulieren wir ein Grid mit fünf Sites. Hierbei bedienen wir uns aller Sites, die in den beiden vorherigen Versuchen eingesetzt wurden (KTH, SDSC00, CTC, SDSC03, SDSC05). Alle anderen Versuchsbedingungen entsprechen denen der vorangehenden Versuche, um die Vergleichbarkeit der Ergebnisse zwischen den Versuchen sicherzustellen. Die Auswertung der Ergebnisse dieses Teilversuches bringt neben den erwarteten Ergebnissen auch einige überraschende Effekte zum Vorschein:

- Im Gegensatz zu den beiden vorherigen Versuchen gilt nun keine eindeutige Kopplung zwischen einseitigem egoistischen Verhalten in dem System und der Verschlechterung des Gesamtsystems. Wie in Abbildung 7.5 zu sehen, scheint eine egoistische Strategie des KTH zu einer Verbesserung der globalen AWRT zu führen. Dieser Effekt scheint mit der Größe der Site in Zusammenhang zu stehen, denn auch die egoistische Strategie auf dem SDSC00 hat nur wenig Einfluss auf die Antwortzeit, wohingegen ein solches Verhalten durch größere Sites wie dem SDSC03 zu eindeutigen Verschlechterungen der Antwortzeit (2,4% bis 5%) führt.

- Zusätzlich zu der auftretenden globalen Verbesserung im Fall des egoistischen KTH, tritt hier ebenfalls der Effekt auf, dass einseitiges egoistisches Verhalten nicht zur generellen Verschlechterungen der AWRT bei allen anderen Sites führt. Stattdessen scheint in diesem Fall - wie in Abbildung 7.5 zu erkennen - auf allen Sites außer dem SDSC00 eine Verbesserung der AWRT gegenüber dem System mit fairem Jobaustausch einzutreten. Diese weitere Verbesserung der globalen AWRT ist äußerst gering, doch stellt sie ein klares Gegenbeispiel zur generellen Verschlechterung der globalen AWRT durch egoistisches Verhalten gegenüber fairem Jobaustausch dar.

- Wenig überraschend ist die erneute Verbesserung der AWRT bei fairem Jobaustausch auf allen Sites gegenüber der lokalen Ausführung. Wie in Bild 7.6 zu sehen, ist der Grad der Verbesserung wieder von der Größe der Site abhängig. Die einzige Ausnahme bildet hier der SDSC00. Er profitiert mit 76% am meisten von der

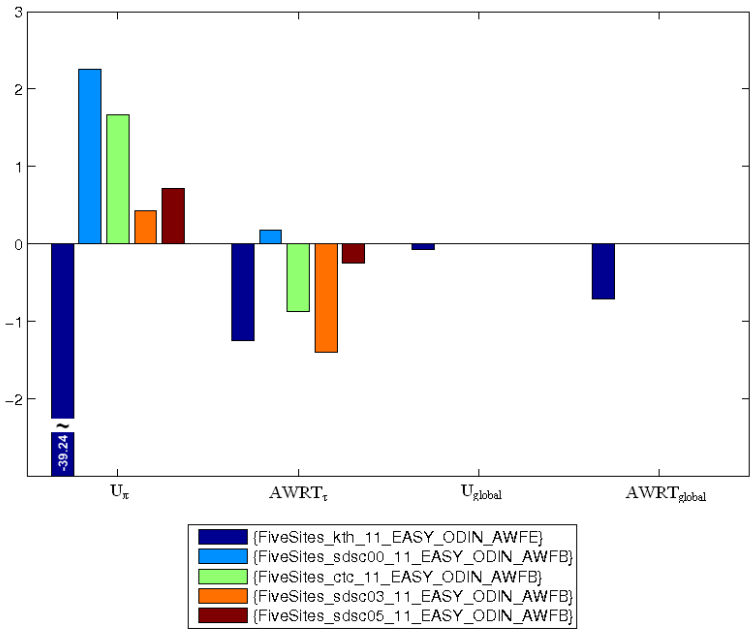


Abbildung 7.5: Egoistisches Verhalten des KTH verglichen mit fairem Jobaustausch

Teilnahme am Grid, obwohl der KTH 28 CPUs kleiner ist. Eine Erklärung für diese starke Verbesserung könnte sein, dass der SDSC00 bei lokaler Ausführung mit 73% schon überdurchschnittlich stark ausgelastet ist und bei Teilnahme am Grid viel Arbeit an andere Sites abgibt. Ein Blick in die Austauschmatrizen 7.2 und 7.3 bestätigt diese Vermutung. Der SDSC00 berechnet mit 77% der bei ihm lokal eingereichten Jobs nur 53% der lokal eingereichten Arbeitslast. Gleichzeitig nimmt er auch nur wenige fremde Jobs an, abgesehen von Jobs des KTH, die aber aufgrund der Größe dieser Site verhältnismäßig klein sind.

- Erkennbar ist in Abbildung 7.6 ebenfalls, dass es zwei Möglichkeiten gibt, eine bessere Antwortzeit zu erreichen. Dies erfolgt zum einen durch überwiegende Abgabe von Arbeitslast bei einem damit verbundenen Rückgang der Auslastung (z.B. SDSC00) und zweitens durch geschicktes Annehmen und Ablehnen von Jobs bei einer Zunahme der Auslastung (z.B. SDSC05). Geschickt bedeutet in diesem Fall, dass Jobs vor allem dann angenommen werden, wenn Sie in die Lücken im Schedule passen. Das erhöht die Auslastung auf der annehmenden Site und führt unter Umständen dazu, dass die abgebende Site sich im weiteren Verlauf durch Übernahme von Arbeitslast gegenüber der annehmenden Site revanchieren kann und somit deren Antwortzeit verkürzt wird.

- Weiterhin ist auch in diesem Versuch ein geringerer Einfluss kleiner Sites auf das System gegenüber großen Sites erkennbar. Egoistisches Verhalten kleiner Sites führt zu weitaus geringeren Verschlechterungen der Antwortzeit als der Egoismus großer

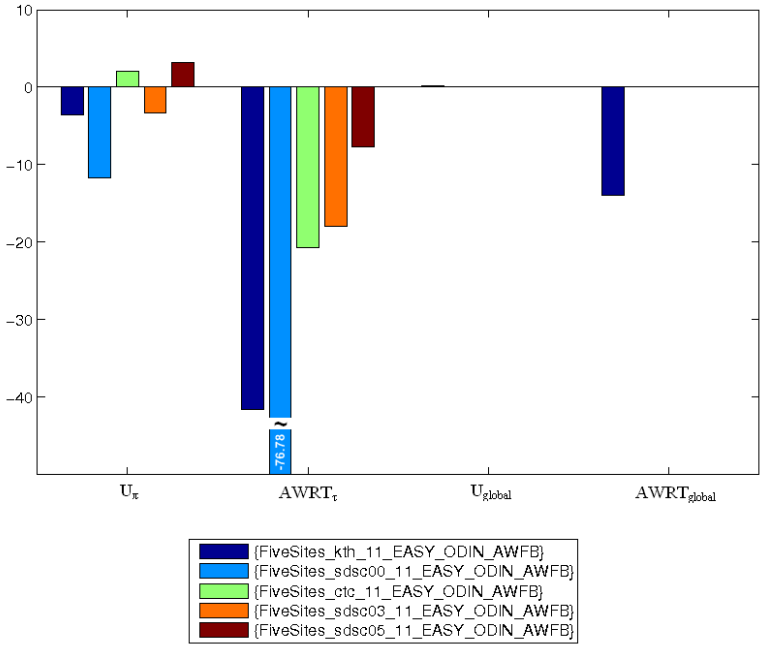


Abbildung 7.6: Veränderungen wichtiger Metriken durch Jobaustausch gegenüber der lokalen Ausführung

$\mathcal{M}$	CTC-11	SDSC03-11	SDSC05-11	KTH-11	SDSC00-11
CTC-11	84.3133	5.6595	1.8692	4.8719	3.2864
SDSC03-11	3.7890	84.5176	6.0914	3.2860	2.3163
SDSC05-11	4.2575	4.9905	85.9231	2.4231	2.4061
KTH-11	7.8093	3.5114	1.4502	71.4210	15.8087
SDSC00-11	8.8125	3.9349	1.4324	8.5072	77.3130

Tabelle 7.2: Austauschmatrix für migrierte Jobs in Prozent

$\mathcal{M}^R$	CTC-11	SDSC03-11	SDSC05-11	KTH-11	SDSC00-11
CTC-11	62.8579	20.8534	9.9103	3.2133	3.1887
SDSC03-11	3.9214	76.9449	17.9678	0.5620	0.6052
SDSC05-11	3.4610	8.2326	86.5215	0.8190	0.9666
KTH-11	18.3968	10.4577	4.6759	47.6596	18.8820
SDSC00-11	20.1573	13.1221	3.2099	10.2739	53.2369

Tabelle 7.3: Austauschmatrix für migrierte Arbeit in Prozent

Sites. Während die egoistische Strategie z.B. bei SDSC00 nur zu einer Verschlech-

terung der globalen AWRT um 0,02% gegenüber dem System mit fairem Jobaustausch führt, so resultiert dasselbe Verhalten beim SDSC03 beispielsweise in 5,4% Verschlechterung. Ebenfalls aufgefallen ist die Schwachstelle der AcceptWhenFit-Austauschstrategie. Diese Strategie ist egoistischem Verhalten schutzlos ausgeliefert, denn eine Site, die ausschließlich mit dieser Strategie betrieben wird, nimmt von dem offensichtlichen Egoismus keinerlei Notiz und nimmt auch weiterhin Jobs von der egoistischen Site an. Abhilfe kann an dieser Stelle nur eine Kombination mit einer der Score-Strategien bringen (z.B. ODIN_{OL}).

7.3.4 Fazit dieser Versuchsreihe

Die Versuchsreihe zum einseitigen egoistischen Verhalten hat einige zuvor schon bekannte Vermutungen bestätigt. Hierzu gehört die unterschiedliche Einflussnahme kleiner und großer Sites auf den Jobaustausch in einem Grid. Andere Vermutungen z.B. eine generelle Verschlechterung der globalen Metriken durch egoistisches Verhalten oder die Vermutung, dass die anderen Sites nicht durch egoistisches Verhalten einer Site profitieren können, da sie dadurch mehr Arbeit haben, konnten widerlegt werden. Weiterhin konnte nachgewiesen werden, dass egoistisches Verhalten durchaus zu einer subjektiven Verschlechterung der Antwortzeit einer einzelnen Site gegenüber der AWRT bei Nichtteilnahme dieser am Grid führen kann. Dieser Effekt scheint bei größeren Grids mit mehr Sites zu verschwinden, da das egoistische Verhalten einer Site in einem größeren Grid nicht mehr auf dem Rücken einzelner anderer Sites ausgetragen wird, sondern im Allgemeinen von mehreren anderen Sites getragen wird. Die Überprüfung dieser Aussagen auf Beständigkeit in Grids mit mehreren egoistischen Sites könnte Teil einer weiteren Untersuchungsreihe sein.

7.4 Mehrere Entscheidungsstrategien im Vergleich

Diese Versuchsreihe soll aufzeigen, wie sich verschiedene Entscheidungsstrategien in Kombination miteinander verhalten. Zu diesem Zweck wird, wie bei den Versuchen mit den egoistischen Entscheidungsstrategien (vgl. Abschnitt 7.3.1), die $\text{ODIN}_{\text{AWFB}}$-Strategie als Referenz-Entscheidungsstrategie betrachtet.
Einige der in Abschnitt 5 vorgestellten Entscheidungsstrategien haben einstellbare Parameter. In einem realen P2P-System würde man nicht simulieren können, wie sich das Gesamtsystem aufgrund unterschiedlich eingestellter Parameter der Entscheidungsstrategien einzelner Sites, verhält. Dies hat zur Folge das man vorher nicht weiß, wie gut oder schlecht die Ergebnisse mit den eingestellten Parametern werden. Hier wäre es besonders positiv, wenn man Standardeinstellungen der Entscheidungsstrategien finden könnte, die sich in nahezu allen Site-Konfigurationen positiv auf die Ergebnisse auswirken. Eine Alternative wäre es, dass sich die Parameter in Abhängigkeit der aktuell bekannten Sites selbst verändern, um möglichst gute Ergebnisse für die eigene Site, oder auch das Gesamtsystem zu erzielen.
Bei dieser Versuchsreihe werden verschiedene Parameter bei den Entscheidungsstrategien ausprobiert, um eine optimale Gesamt-AWRT, also die AWRT über alle verwendeten

Sites, zu erreichen. Im folgenden werden nur die Setups betrachtet, die die besten Ergebnisse, bezogen auf eine Entscheidungsstrategie, erzielt haben.

7.4.1 Verschiedene Entscheidungsstrategien in einem Grid bestehend aus drei kleinen Sites

Wie eingangs beschrieben, sollen die verschiedenen Entscheidungsstrategien untereinander verglichen werden. Hierzu wird eine feste Site-Konfiguration gewählt, in der nur die Entscheidungskomponenten ausgewechselt werden. Jede Site-Konfiguration besteht hierbei aus 3 Sites. In diesem Versuch werden KTH, CTC und SDSC00 verwendet. Als Scheduling-Algorithmus für das lokale Scheduling wird EASY-Backfilling gewählt. Folgende Entscheidungsstrategien wurden untersucht: ODIN_{OL}, $\text{ODIN}_{\text{AWFB}}$, ODIN_{KG}, ODIN_{ROQ}, ODIN_{ST}, $\text{ODIN}_{\text{WTQE}}$ und $\text{ODIN}_{\text{WTQH}}$. Erläuterungen zu diesen Strategien befinden sich in Kapitel 5. Die Ergebnisse werden nun im Einzelnen vorgestellt:

- **ODIN-KillGaps**: Bei dieser Strategie werden Jobs nur dann von einer Site angenommen, wenn die Differenz aus durchschnittlich benötigter CPU-Anzahl auf dieser Site gerechneter Jobs und den CPUs der Site größer gleich der vom aktuellen Job geforderten CPUs ist. Lokal eingereichte Jobs, die nicht abgewiesen werden können müssen jedoch trotzdem lokal gerechnet werden. Bei den hier erzielten Ergebnissen wurden maximal 3% aller Jobs versendet. Dies hat zur Folge, dass in diesem Versuch nur ein mäßiger Gewinn, gegenüber der Variante, dass alle Jobs lokal gerechnet werden, erzielt werden kann. Ein Vergleich zum $\text{ODIN}_{\text{AWFB}}$-Fall, bei dem manche Sites bis zu 37% ihrer Jobs abgegeben haben, zeigt deutlich, dass bei so einem geringen Jobaustausch kaum Verbesserungen möglich sind.

- **ODIN-RessourcesOfQueue**: Diese Strategie wird maßgeblich von dem einstellbaren Schwellwert beeinflusst. Übersteigt das Produkt von angeforderten CPUs und der angeforderten Ausführungszeit eines Jobs, summiert über alle Jobs in der Queue, den Schwellwert, so wird versucht den Job loszuwerden, bzw. ein „Remote“-Job wird nicht angenommen. Da hier die Schwellwerte für lokal-eingereichte Jobs und „Remote“-Jobs unabhängig einstellbar sind, können durch verschiedene Parameter, die AWRT und die Auslastung einer einzelnen Site signifikant verändert werden. Unter allen ausgeführten Experimenten mit dieser Entscheidungsstrategie wird hier die betrachtet, für die die Gesamt-AWRT das beste Ergebnis erzielte (siehe Abbildung 7.7). Hier wurde die größte Site (CTC-Trace mit 430 CPUs) so eingestellt, dass die Site viele Jobs annimmt und die größeren Jobs behält. Es ist zu erkennen, dass die Auslastung dieser Site gegenüber dem $\text{ODIN}_{\text{AWFB}}$-Fall noch weiter steigt. Hinzu kommt in diesem Fall, dass die AWRT ebenfalls um ca. 1% sinkt. Beide kleineren Sites profitieren in diesem Experiment sowohl bei der Auslastung als auch bei der AWRT. In Abbildung 7.8 ist zusätzlich noch der Vergleich zu dem Fall, dass jede Site ihre Jobs lokal rechnet, dargestellt. Selbst die größte Site, die sehr viele Jobs annimmt (50% von der KTH-Site und 19% von der SDSC00-Site), gewinnt, wenn auch nur geringfügig, gegenüber der lokalen Ausführung.

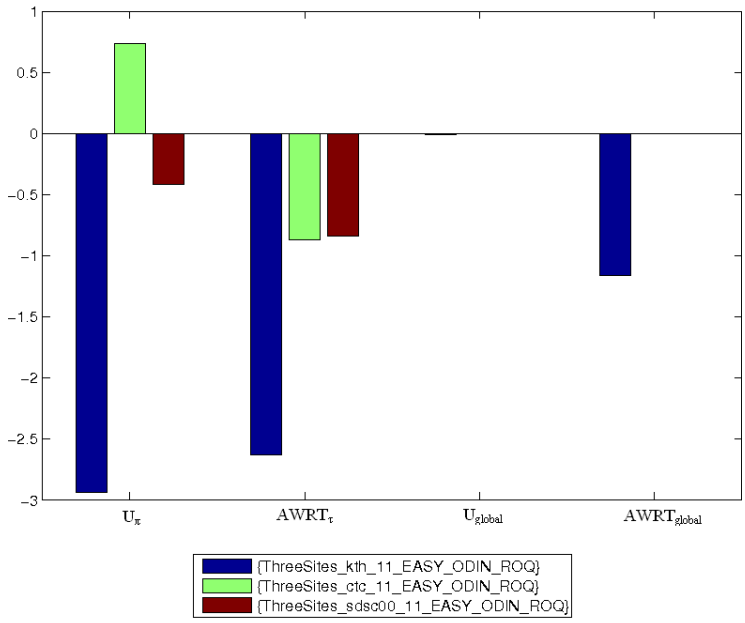


Abbildung 7.7: Verbesserungen durch Jobaustausch ODIN_{ROQ} gegenüber $\text{ODIN}_{\text{AWFB}}$

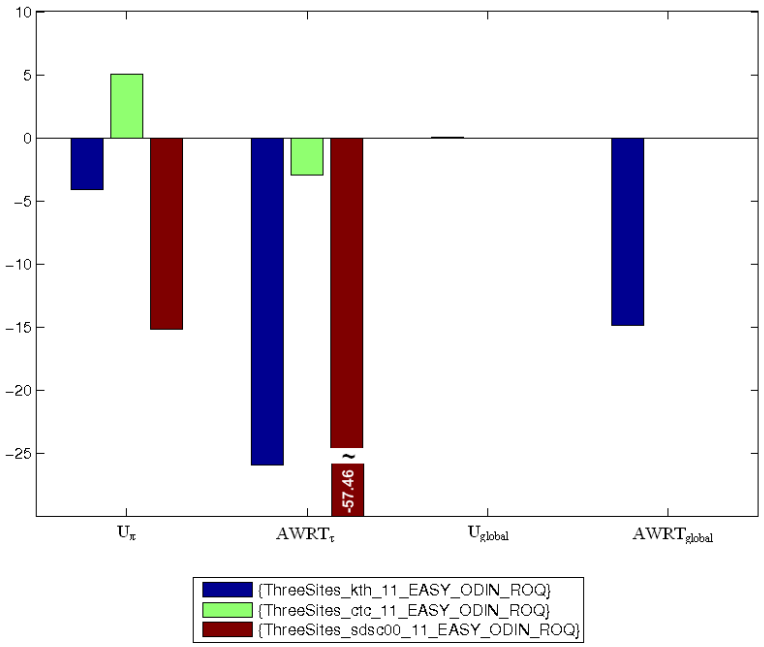


Abbildung 7.8: Verbesserungen durch Jobaustausch ODIN_{ROQ} gegenüber ODIN_{OL}

- **ODIN-ScheduleTime**: Diese Entscheidungsstrategie ist eine Erweiterung der Entscheidungsstrategie $\text{ODIN}_{\text{AWFB}}$ und erzielt somit auch vergleichsweise ähnliche Ergebnisse. Für jeden Job wird überprüft ob der frühestmögliche Ausführungszeitpunkt innerhalb einer bestimmten Grenze liegt und dementsprechend wird der Job angenommen oder abgelehnt. Durch die einstellbaren Grenzen konnten hier geringfügig bessere Ergebnisse als bei der Entscheidungsstrategie $\text{ODIN}_{\text{AWFB}}$ erzielt werden. Die Gesamt-AWRT des Setups ist um 0,5% gefallen. Die recht ähnlichen Ergebnisse zum $\text{ODIN}_{\text{AWFB}}$ sind auf die ähnlichen Strategien zurückzuführen. Beide Entscheidungsstrategien fällen ihre Entscheidungen aufgrund der aktuell im Schedule befindlichen Jobs, wobei die Strategie ODIN_{ST} etwas geschickter vorgeht und zusätzlich überprüft, ob der Job wirklich in eine bestehende Lücke im Schedule hineinpasst.
- **ODIN-WaitTimeQueueEasy und ODIN-WaitTimeQueueHard**: Diese beiden Entscheidungsstrategien basieren auf 2 Entscheidungskriterien. Einerseits wird anhand der Queue-Größe überprüft ob es sinnvoll ist den Job noch aufzunehmen, andererseits wird überprüft, ob die durchschnittliche Wartezeit eines Jobs einen vorher eingestellten Schwellwert bereits übersteigt. Die Entscheidungsstrategie $\text{ODIN}_{\text{WTQH}}$ erlaubt es, die Schwellwerte für lokal-eingereichte Jobs und entfernte Jobs getrennt einzustellen. Mit diesen Entscheidungsstrategien verliefen die Versuche weit weniger erfolgreich. Bei beiden Strategien kam es nur selten zum Jobaustausch zwischen den verschiedenen Sites, so dass die Ergebnisse sehr ähnlich zum dem Fall lagen, dass alle Sites einzeln rechnen.

7.4.2 Verschiedene Entscheidungsstrategien in einem Grid bestehend aus fünf Sites

Dieses Experiment ist sehr ähnlich zum vorherigen Experiment aufgebaut. Der einzige Unterschied besteht in der Anzahl der verwendeten Sites. Es werden, wie auch in anderen 5-Site-Experimenten, alle fünf verschiedenen Sites genutzt. Alle anderen Einstellungen werden nicht verändert. So wird das Experiment auch für dieselben Entscheidungsstrategien ausgeführt. Auch hier werden die Ergebnisse für jede Entscheidungsstrategie einzeln vorgestellt:

- **ODIN-RessourcesOfQueue**: Anders als im 3-Site-Experiment konnten mit dieser Entscheidungsstrategie keine Parameter gefunden werden, um eine Verbesserung gegenüber der $\text{ODIN}_{\text{AWFB}}$-Strategie zu erreichen. Vor allem die kleineren Sites KTH und SDSC00 können in diesem Versuch nicht die gleichen Verbesserungen erzielen. Solange sich keine weiteren Jobs in der Queue befinden werden alle Jobs angenommen, und erst wenn sich die Queue füllt wird die Entscheidungsstrategie wirklich aktiv und bewertet die Jobs aufgrund des Produktes aus erwarteten Prozessoren und erwarteter Laufzeit. Anhand dieser Bewertung wird über die Annahme oder Ablehnung des Jobs entschieden . So mussten eventuell größere Jobs, die die Entscheidungsstrategie $\text{ODIN}_{\text{AWFB}}$ abgelehnt hätte, angenommen werden, so dass die AWRT insbesondere bei kleineren Sites angestiegen ist.

- **ODIN-ScheduleTime**: Wie schon im Experiment mit den 3 Sites (vgl. Abschnitt 7.4.1), ist bei dieser Strategie im Allgemeinen von einer Verbesserung gegenüber der $\text{ODIN}_{\text{AWFB}}$-Strategie auszugehen. Aus den Matrizen 7.4 und 7.5 lässt sich eine eindeutige Tendenz erkennen, dass die kleineren Sites versuchen ihre größeren Jobs abzugeben und gleichzeitig auch kaum größere Jobs annehmen. Die größeren Sites hingegen nehmen aufgrund der eingestellten Schwellwerte auch die größeren Jobs an.

$\mathcal{M}$	CTC-11	SDSC03-11	SDSC05-11	KTH-11	SDSC00-11
CTC-11	84.7874	5.4833	1.8187	4.8084	3.1025
SDSC03-11	3.8973	84.5725	6.0823	3.0741	2.3742
SDSC05-11	4.2094	5.0051	86.1127	2.4525	2.2205
KTH-11	8.0480	3.5465	1.3870	71.0804	15.9386
SDSC00-11	9.0641	4.0356	1.4592	8.9198	76.5213

Tabelle 7.4: Austauschmatrix für das Experiment mit 5 Sites und ODIN_{ST}über ausgetauschte Jobs in Prozent

$\mathcal{M}^R$	CTC-11	SDSC03-11	SDSC05-11	KTH-11	SDSC00-11
CTC-11	62.0063	20.8181	10.5280	3.3746	3.2973
SDSC03-11	4.2012	76.7647	17.8190	0.5508	0.6657
SDSC05-11	3.5374	8.3723	86.3861	0.8027	0.9024
KTH-11	19.0846	9.9904	4.2699	47.3255	19.4036
SDSC00-11	19.8703	13.0164	3.5208	10.2180	53.3746

Tabelle 7.5: Austauschmatrix für das Experiment mit 5 Sites und ODIN_{ST}über die ausgetauschte Arbeitslast in Prozent

So nimmt der SDSC03 ca. 1,8% aller Jobs von dem CTC an, dies entspricht aber einem Ressourcenprodukt (CPUs * angeforderte Zeit) von über 10%. Der KTH als Beispiel für eine kleine Site behält selbst 71% aller Jobs, wobei dies nur einem Ressourcenprodukt von ca 47% entspricht. Das dazugehörige Vergleichsdiagramm in Abbildung 7.9 zeigt zudem, dass beinahe alle Sites gegenüber der $\text{ODIN}_{\text{AWFB}}$-Betrachtung bei der AWRT gewonnen haben, und diese im Durchschnitt um 0.7% verbessern konnten.

- **ODIN-WaitTimeQueueEasy und ODIN-WaitTimeQueueHard**: Analog zum 3-Sites-Fall lässt sich festhalten, dass beide Strategien einen zu geringen Jobaustausch haben. So werden bei der Strategie $\text{ODIN}_{\text{WTQE}}$ auf allen Sites nur unter 0.1% aller Jobanfragen von anderen Sites positiv beantwortet, was dazu führt, dass eine Site, die einen Job abgeben möchte, gar keine Möglichkeit finden wird, diesen auf einer anderen Site rechnen zu lassen. Bei der Strategie $\text{ODIN}_{\text{WTQH}}$ ist der Jobaustausch geringfügig höher, da die Parameter für externe und lokale Jobs unabhängig voneinander eingestellt werden können. Eine Vermutung, warum diese

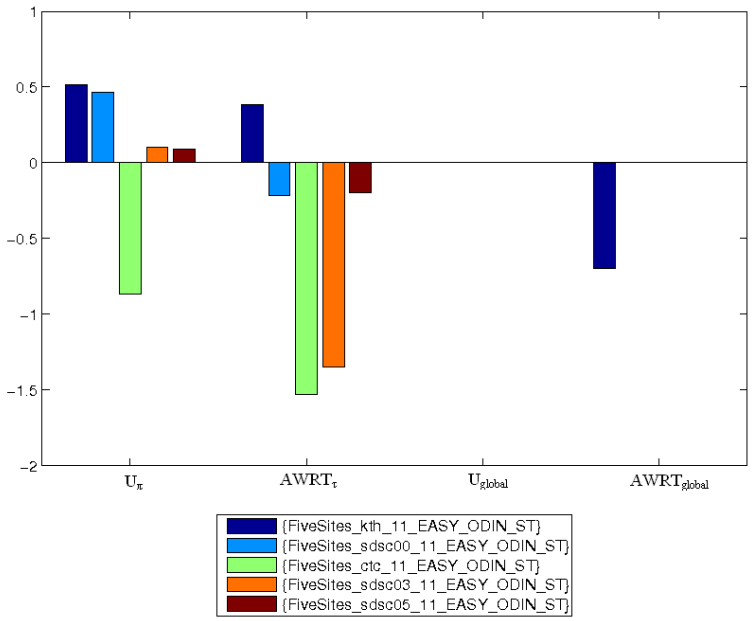


Abbildung 7.9: Verbesserungen im 5-Sites-Experiment durch Jobaustausch ODIN_{ST} gegenüber $\text{ODIN}_{\text{AWFB}}$

Strategien nur unzureichende Ergebnisse liefern, besteht darin, dass größtenteils Durchschnittswerte als Entscheidungskriterium herangezogen werden, und so zu wenig Rücksicht auf die momentane Situation des Schedules genommen wird.

7.4.3 Fazit dieser Versuchsreihe

Die hier verglichenen Entscheidungsstrategien haben gegebenüber der $\text{ODIN}_{\text{AWFB}}$-Strategie nur eine geringe Verbesserung erzielen können. Es waren erste Versuche sinnvolle Entscheidungsstrategien zu entwickeln, wenn auch nicht alle Entscheidungsstrategien gute Ergebnisse lieferten. So zeigte vor allem die Entscheidungsstrategie ODIN_{ST}, dass durch einen geschickte Auswahl der Jobs, bessere Ergebnisse als durch die simple $\text{ODIN}_{\text{AWFB}}$-Strategie in beiden Experimenten erzielt werden konnte. Verbesserungspotential ist bei den Entscheidungsstrategien noch vorhanden, da die eingesetzten Entscheidungsstrategien meist entweder nur auf die aktuelle Situation oder nur auf die Durchschnittswerte achteten. So wäre eine Entscheidungsstrategie, die beide Ansätze kombiniert, durchaus denkbar und vorteilhaft.

Es lässt sich feststellen, dass nicht die Anzahl der Sites, sondern gerade die unterschiedlichen Entscheidungsstrategien erheblich für die unterschiedlichen Ergebnisse in diesem Versuch verantwortlich sind. So haben nahezu alle Entscheidungsstrategien bei beiden Experimenten ihre Charakteristik Trace-übergreifend behalten. Sie haben signifikanten Einfluss auf im Grid-Kontext wichtige Metriken, wie z.B. AWRT und Auslastung.

7.5 Experiment mit 3 kleinen Sites und der Entscheidungsstrategie ODIN_SiteScore

Der im Abschnitt 7.3.1 beschriebene Versuch wurde mit einer anderen Entscheidungstrategie noch einmal wiederholt. Diesmal wurde statt der Strategie $\text{ODIN}_{\text{AWFB}}$ die Strategie ODIN_{SS} benutzt. Der Versuch wird in diesem Kapitel beschrieben. Die Tabellen mit den Ergebnissen des Experiments befinden sich im Anhang im Abschnitt A.4.6. Die Strategie ODIN_{SS} wird im Detail in Abschnitt 5.10.2 beschrieben.
Die Strategie ODIN_{SS} ist inspiriert von BitTorrent (siehe Abschnitt 5.10.1). Diese Strategie soll dafür sorgen, dass jede Site, die am Grid teilnimmt, sich kooperativ verhält. Auf diese Weise sollen alle Sites von der Teilnahme am Grid profitieren.
Wenn eine Site nur Jobs an andere Sites verschickt, aber selbst keine Jobs von anderen Sites annimmt, dann wird solches Verhalten als egoistisch bezeichnet. Wenn eine Site Jobs von den anderen Sites annimmt und versucht diese sobald wie möglich auszuführen, dann wird solches Verhalten als kooperativ bezeichnet.
Kooperatives Verhalten ist erwünscht, egoistisches ist dagegen unerwünscht. Letzteres schadet anderen und kann dazu führen, dass andere Teilnehmer durch die Teilnahme am Grid verlieren. Kooperatives Verhalten soll belohnt werden, egoistisches dagegen bestraft werden.
Die Site-Score-Strategie erkennt, wenn eine Site sich egoistisch verhält und nimmt ihrerseits keine Jobs von der egoistischen Site an. Auf diese Weise wird egoistisches Verhalten bestraft.
Die Strategie ODIN_{SS} lässt sich konfigurieren. Es ist möglich, den Algorithmus, der egoistisches Verhalten bestraft, mit dem Entscheidungsalgorithmus von $\text{ODIN}_{\text{AWFB}}$ zu kombinieren. Das heißt: eine Site i nimmt einen remote Job nur dann an, wenn die Site j, von der der Job kommt, mit Site i kooperiert hat und Site i genug freie Ressourcen zur Verfügung hat, um den Job sofort auszuführen. Auf diese Weise soll nicht nur die Kooperation zwischen Sites erzwungen, sondern auch die AWRT_τ jeder Site gesenkt werden.
In den folgenden vier Versuchen wurde diese kombinierte Entscheidungsstrategie getestet. Die Ergebnisse von jedem Versuch werden mit dem Versuch verglichen, bei dem alle Sites die Strategie ODIN_{OL} benutzen.

Alle Sites mit ODIN_{SS}: Beim ersten Versuch haben alle drei Sites die Strategie ODIN_{SS} benutzt (vgl. Abbildung 7.10). Die globale AWRT hat sich um 5% verbessert. Die AWRT_τ der Site KTH hat sich um 13% verbessert, die AWRT_τ der Site CTC hat sich um 1% unmerklich verschlechtert, die AWRT_τ der Site SDSC hat sich um 12% verbessert.

Man sieht, dass die kleineren Sites durch die Teilnahme am Grid am meisten gewinnen. Die Site KTH und die Site SDSC sind beide viel kleiner als die Site CTC und haben beide ungefähr gleich viele Prozessoren. Die Site CTC verliert sogar durch die Teilnahme am Grid.

Site CTC - egoistisch: Beim zweiten Versuch haben die Sites KTH und SDSC die Strategie ODIN_{SS} benutzt (vgl. Abbildung 7.11).

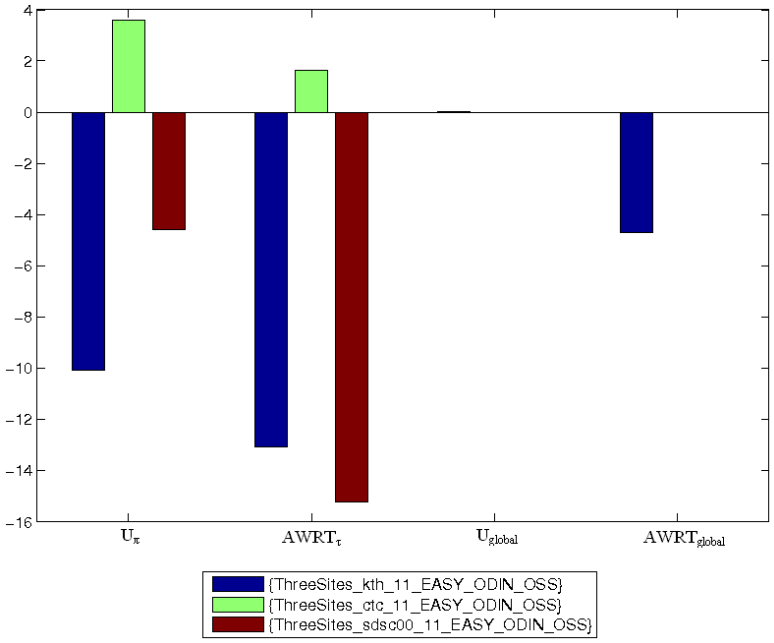


Abbildung 7.10: Alle Sites benutzen die OSS Strategie

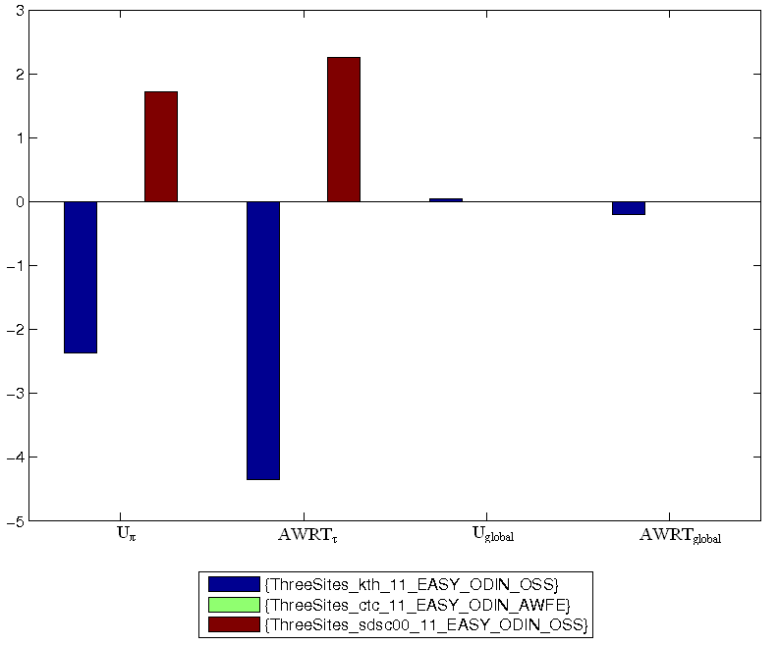


Abbildung 7.11: Alle OSS, CTC - egoistisch

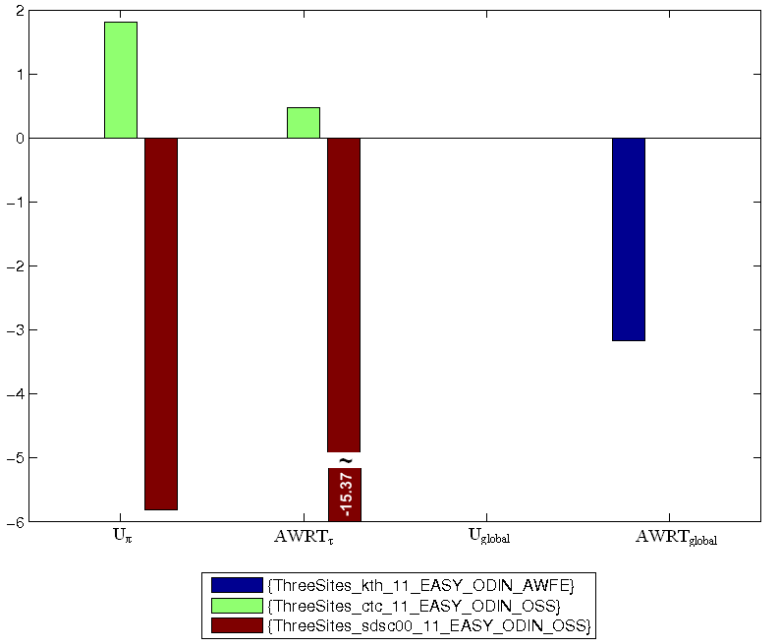


Abbildung 7.12: Alle OSS, KTH - egoistisch

Die Site CTC hat dagegen die egoistische Strategie $\text{ODIN}_{\text{AWFE}}$ benutzt. Die globale AWRT ist gleich geblieben. Die AWRT_τ der Site KTH hat sich um 6% verbessert, die AWRT_τ der Site CTC hat sich wie erwartet nicht verändert, die AWRT_τ der Site SDSC hat sich um 3% verschlechtert. Die Werte der Site CTC haben sich nicht verändert weil sie keine Jobs mit den anderen Sites ausgetauscht hat. D.h. bei diesem Versuch haben nur die Site KTH und die Site SDSC miteinander Jobs ausgetauscht. Dabei hat wieder die kleinere der beiden Sites also KTH gewonnen. Die größere Site SDSC hat minimal verloren.

Site KTH - egoistisch: Beim dritten Versuch haben die Sites CTC und SDSC die Strategie ODIN_{SS} benutzt (vgl. Abbildung 7.12). Die Site KTH hat dagegen die egoistische Strategie $\text{ODIN}_{\text{AWFE}}$ benutzt. Die globale AWRT hat sich um 3% verbessert. Die AWRT_τ der Site CTC ist gleich geblieben, die AWRT_τ der Site CTC hat sich um 1% unmerklich verschlechtert, die AWRT_τ der Site SDSC hat sich um 13% verbessert. Die Werte der Site KTH haben sich nicht verändert. Bei diesem Versuch hat wiederum die kleinere Site SDSC stark gewonnen, die größere Site CTC hat wieder minimal verloren.

Site SDSC - egoistisch: Beim dritten Versuch haben die Sites KTH und CTC die Strategie ODIN_{SS} benutzt (vgl. Abbildung 7.13). Die Site SDSC hat dagegen die egoistische Strategie $\text{ODIN}_{\text{AWFE}}$ benutzt. Die globale AWRT hat sich um 1% verbessert. Die AWRT_τ der Site KTH hat sich um 12% verbessert, die AWRT_τ der Site CTC hat sich um 1.5% unmerklich verschlechtert. Die Werte der Site SDSC haben sich

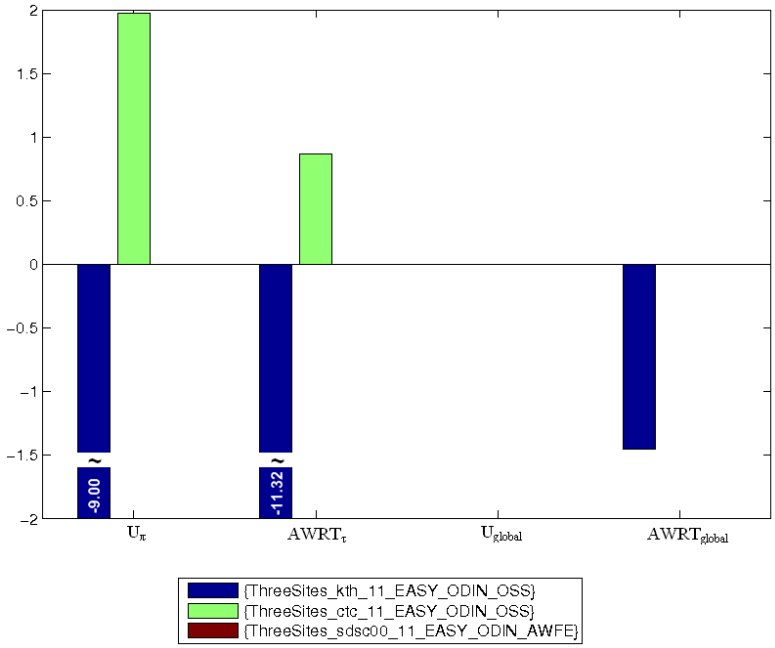


Abbildung 7.13: Alle OSS, SDSC - egoistisch

nicht verändert. Auch bei diesem Versuch hat sich das früher beobachtete Verhalten gezeigt. Die größere Site CTC hat minimal verloren, die kleinere Site KTH hat sich dagegen stark verbessern können.

In diesem Versuch wurden nur Sites mir großen Unterschieden bezüglich der Prozessoranzahl untersucht. So hat die Site CTC vier mal so viele Prozessoren, wie die Site KTH und 3,3 mal so viele wie die Site SDSC. Einerseits hat sich gezeigt, dass eine Site, die sich egoistisch verhält, von der Teilnahme am Grid ausgeschlossen wird und keine Möglichkeit bekommt, ihre $AWRT_\tau$ auf Kosten anderer Sites zu verbessern. D.h. egoistisches Verhalten im Grid lohnt sich nicht. Andererseits ist beobachtet worden, dass die größte Site ihre $AWRT_\tau$ durch die Teilnahme am Grid nicht verbessern kann. Sie verliert, aber nur minimal im Vergleich zur lokalen Ausführung. Kleinere haben sich dagegen verbessern können. Es ist nicht gelungen mit Hilfe der $ODIN_{SS}$-Strategie das Verhalten eines Grid-Systems so zu manipulieren, dass alle Teilnehmer unabhängig von ihrer Größe von der Gridteilnahme profitieren können.

8 Fazit

Im Rahmen der PG wurde ein Rahmenwerk für die Simulation eines oder mehrerer MPS-Systeme entwickelt, die mit Hilfe einer Peer-To-Peer-Architektur zu einem Computational Grid vernetzt werden. Zielsetzung war dabei ein erweiterbares System, dessen Kernkomponenten wie die Kommunikation, das Scheduling oder die Entscheidung über lokale und entfernte Ausführung leicht durch alternative Implementierungen ersetzt werden können.
Die Arbeit im Team gestaltete sich im Großen und Ganzen einfach. Durch die klare Strukturierung in Teilaufgaben und Zuweisung der Teilnehmer in entsprechende Teilgruppen, sowie einem Scrum-Master, der als zentraler Ansprechpartner bei Fragen diente, gab es eine eindeutige Kontaktperson, die, wenn sie nicht selber die Lösung eines Problems wusste, zumindest an die entsprechenden Teilgruppen, bzw. den konkreten Teilnehmer, verweisen konnte. Nicht immer ging die Gruppenarbeit ohne Differenzen zwischen den PG-Teilnehmern vonstatten. Jedoch war es immer möglich, diese Differenzen in einer weitgehend sachlichen und beherrschten Art und Weise auszuräumen, sodass das Arbeitsklima im Allgemeinen gut und ungestört war. Jeder Teilnehmer hat in der Regel seine ihm übertragenen Aufgaben innerhalb der vorgesehenen Zeitrahmen erledigt, sodass die anderen Teilnehmer ihre Aufgaben auch entsprechend erledigen konnten. Einige Teilnehmer haben, mehr oder weniger freiwillig, durch ihre Aufgaben einen erhöhten Arbeitsaufwand gehabt. So ist zum Beispiel das gesamte Scheduling-System nicht nur bereits in der Planungphase komplex gewesen, sondern musste im Laufe der PG mehr als einmal komplett neu geschrieben werden, um unerwartet aufgetretene Probleme behandeln zu können.
Generell hat die Arbeit in der PG gezeigt, dass eine klare Planung und Aufgabenzuteilung gut funktioniert hat, sondern auch nur nötig war. Durch die Einteilung in Teilgruppen und entsprechender Kompetenzbildung, stand immer ein adäquater Ansprechpartner bei Fragen, Anregungen oder Fehlerhinweisen aus anderen Teilgruppen zur Verfügung.
Die im Laufe der PG entwickelte Anwendung kam für zahlreiche Experimente zum Einsatz, deren Durchführung und Auswertung durch eine eigens entwickelte Evaluierungskomponente weitestgehend automatisiert wurde. Unterschiedliche ODIN-Strategien wurden evaluiert und mit der zentralen Job-Pool-Lösung verglichen. Hierbei weist die zentrale Lösung eine signifikant bessere Performanz als der P2P-Ansatz auf, muss allerdings hinsichtlich ihrer Realisierbarkeit kritisch bewertet werden. Die dezentrale Architektur ist besonders hinsichtlich der Ausfallsicherheit klar im Vorteil und erfordert auch im Realbetrieb nur minimale Anpassung bereits vorhandener Systeme.
Der P2P-Ansatz sowie der entwickelte Simulator können hinsichtlich mehrerer Aspekte als Grundlage für zukünftige Entwicklungen und Untersuchungen dienen. Überraschend ist jedoch, dass anscheinend die einfachsten Strategien wie AcceptWhenFit durch besonders gute Resultate hervorstechen. Die große Schwachstelle einfacher Strategien ist

die mangelnde Anpassungsfähigkeit an unregelmäßiges Verhalten wie den Egoismus einer einzelnen Site. Hier scheinen jene Entscheider-Strategien, die auf Basis eines Site-Scores arbeiten nach ersten Experimenten tendenziell den vielversprechendsten Ansatz darzustellen. Allerdings sei an dieser Stelle ausdrücklich darauf hingewiesen, dass weitere Experimente notwendig sind, um die tatsächliche Leistungsfähigkeit einwandfrei nachzuweisen. Ein durchgängiges Problem, vorkonfigurierter Austauschstrategien ist die mangelnde Einsatzfähigkeit in reellen Systemen. Denn dort müsste man vor Ersatz einer solchen Strategie wissen, von welcher Art lokal eingereichte Jobs sind und aus welchen Sites das Grid besteht, um eine optimale Konfiguration vornehmen zu können. Das Problem hierbei ist, dass diese Vorkonfiguration für den kompletten Einsatz der Strategie im Grid beibehalten werden muss. Tendenziell sollte die Forschung in diesem Bereich in die Richtung von Austauschstrategien gehen, die ihr Verhalten während des Betriebs an die jeweiligen Umstände anpassen können, beispielsweise durch den Einsatz von Online-Lernverfahren.
Weiterhin haben die Experimente gezeigt, dass neben den Austauschstrategien auch die Größe der teilnehmenden Sites maßgeblichen Einfluss auf die Verteilung der Jobs innerhalb des Systems und die Ausprägung qualitativ bewertbarer Metriken wie AWRT und Auslastung hat. Diese Erkenntnisse decken sich mit den Ergebnissen anderer ähnlich gelagerter Experimente.
Die bisherige Implementierung des P2P-Ansatzes bietet ebenfalls Raum für weitere Untersuchungen. Im Gegensatz zu der bisher verwendeten, unstrukturierten Architektur, könnte ein hybrider Ansatz zum Einsatz kommen, der sich weitestgehend an der Kazaa-Architektur orientiert (vgl. Liang et al. (2004)). Dieses Szenario erlaubt die Definition von Inseln von MPS-Systemen, die dynamisch über einen Super-Knoten verwaltet werden.
Ein weiteres Feld, dass in den bisherigen Untersuchungen nicht berücksichtigt wurde, ist der Ausfall von Peers. Sämtliche durchgeführten Simulationen basierten auf der Annahme, dass alle eingesetzten Maschinen auch bis zum Ende der Simulation erreichbar und verfügbar sind. Auch wenn Rechenzentren, für die diese Grid-Anwendung angedacht ist, in der Regel über eine sehr geringe Ausfallwahrscheinlichkeit verfügen, ist ein Ausfall nicht gänzlich ausgeschlossen.
Eine Erweiterung der lokalen Scheduling-Algorithmen ist durchaus möglich. Hierbei wäre von der Kapselung des lokalen Scheduling-Systems abzusehen. So könnte es den lokalen Schedulern ermöglicht werden, Jobs auch nach Zuteilung durch eine Entscheidungsstrategie zu versenden bzw. auf gegebene freie Ressourcen passende Jobs von anderen Sites anzufragen.

A Anhang

Inhalt

A.1 Installation

Auf der Projekt-Homepage[1] steht ein Archiv zum Download bereit, welches die Simulations- und Evaluationsumgebung inklusive aller benötigten Werkzeuge in einer einheitlichen Umgebung bereitstellt. Das Paket ist sowohl unter Windows- als auch unter Linux-Systemen lauffähig und enthält folgende Verzeichnisstruktur:

[1] `http://www-ds.e-technik.uni-dortmund.de/~pg505/content/` (08/2007)

jars Enthält die entwickelte Simulations- und Evaluationskomponente[2], sowie zusätzlich benötigte Java-Komponenten.

logs Hier werden die von den enthaltenen Werkzeugen erzeugten Logdateien gespeichert.

results Unterhalb dieses Verzeichnisses liegen die Ergebnisse der durchgeführten Simulationen.

setups In diesem Verzeichnis werden die Site-Setup-Dateien im XML-Format gespeichert.

tmp Ein Verzeichnis zur Ablage von temporären Dateien, die von den enthaltenen Werkzeugen während der Ausführung angelegt und verwendet werden.

traces Dieses Verzeichnis dient zur Speicherung von Workload-Traces im Standard Workload Format, die als Eingaben für die simulierten Multiprozessorsysteme dienen können.

xslt Enthält XSLT- und XSD-Dateien, die von einigen Skripten zur Transformation der Ergebnisdaten in verschiedene Formate verwendet werden (siehe auch Kapitel A.3).

A.1.1 Zusätzlich benötigte Programme

Zur Arbeit mit dem System sind ebenfalls noch folgende zusätzliche Programmpakete erforderlich:

- Java Runtime Environment (JRE) 5.0
- Perl 5.8
- Matlab 2006

Auf die Installation und Konfiguration dieser Programme wird nicht eingegangen. Java, Perl und und Matlab sind für verschiedene Betriebssysteme (z.B. Linux und Windows) verfügbar.
Die Kompatibilität des Systems mit anderen als den genannten Programmversionen kann nicht garantiert werden.

A.1.2 Benötigte Perl-Module

Um die korrekte Funktion der Perl-Skripte (siehe Kapitel A.3) zu gewährleisten, müssen die folgenden Perl-Module auf dem zur Simulation verwendeten Rechner installiert sein:

- `Date::Format`
- `Date::Simple`
- `File::Basename`

[2] `GridScheduling.jar` und `Evaluation.jar`

- `File::Glob`
- `Getopt::Std`
- `IO::File`
- `Spreadsheet::WriteExcel`
- `XML::SAX::ParserFactory`
- `XML::Simple`
- `XML::Tidy`
- `XML::Validator::Schema`
- `XML::Writer`
- `XML::XSLT`

Nicht vorhandene Module lassen sich über die CPAN-Shell von Perl installieren. Dazu wird auf der Kommandozeile folgendes Kommando aufgerufen:

```
perl -MCPAN -e shell
```

Anschließend lassen sich die fehlenden Module mit dem `install`-Kommando nachinstallieren. Beispiel:

```
install Date::Format
```

Dabei werden auch ggf. bestehende Abhängigkeiten aufgelöst und entsprechend behandelt. Mit `Control-d` kann die CPAN-Shell anschließend wieder verlassen werden.

A.1.3 Konfiguration der config.pl Datei

Die Konfiguration der enthaltenen Perl-Skripte erfolgt in der von allen Skripten gemeinsam verwendeten Datei `config.pl`, die sich im Hauptverzeichnis der Distribution befinden muss. Diese Datei enthält die Pfadangaben zu den verschiedenen verwendeten externen Programmen, sowie ggf. Aufrufparameter für diese. Die Datei muss nur einmal erzeugt und vom Anwender angepasst werden. Für Linux- und Windows-Systeme werden die Dateien `config.pl_linux` bzw. `config.pl_windows` mitgeliefert, die einfach als `config.pl` kopiert werden und dann als Ausgangsbasis für die Konfiguration dienen können.
Im folgenden soll auf die einzelnen zu konfigurierenden Variablen eingegangen werden. Variablen, die nicht aufgeführt sind, müssen vom Anwender in der Regel nicht verändert werden. Es ist zu beachten, das hierbei auch unter Windows-Systemen der Schrägstrich („/“) als Verzeichnistrenner dient.

$java Enthält den kompletten Pfad zum Java-Programm.
Beispiel: “/usr/bin/java”, “C:/Program Files/Java/jre1.5.0_11/bin/java.exe”

$javaopts Optionale Parameter für den Aufruf von Java.
Beispiel: “-Xmx1500m”

$zip Kompletter Pfad zum verwendenden Kompressionsprogramm.
Beispiel: “/usr/bin/zip”, “/usr/bin/7z”

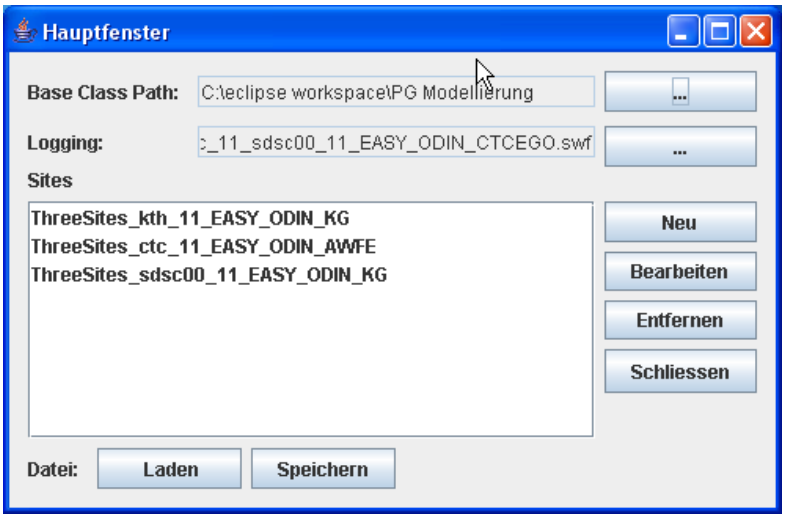


Abbildung A.1: Das Hauptfenster

$zipopts Optionale Parameter für den Aufruf des Kompressionsprogramms.
Beispiel: "-t zip"

$matlab Pfad zum Matlab-Programm.
Beispiel: "/opt/matlab2006/bin/matlab"

A.2 GUI

Im Folgenden wird eine grafische Benutzeroberfläche (GUI) beschrieben, die es erlaubt die Konfiguration für die Gridsimulation zu erstellen und zu bearbeiten. Die Anwendung wurde mit Hilfe der GUI-Bibliothek Swing erstellt. Die Anwendung kann von der Projekt-Homepage[3] heruntergeladen werden. Um sie zu starten, muss der Benutzer zuerst das Archiv „gui.zip" entpacken und im Ordner „dist" die Datei „srg.bat" ausführen.
Der Benutzer ist in der Lage neue Sites zu erstellen, sowie vorhandene Sites zu entfernen und zu bearbeiten. Die Konfiguration kann in einer XML-Datei abgespeichert, und wieder aus einer XML-Datei geladen werden.
Wenn die GUI startet, erscheint zunächst das Hauptfenster (vgl. Abbildung A.1). In diesem Fenster ist die Liste der Sites zu sehen, aus denen das Grid besteht, welches später simuliert werden soll. Nach dem Start der GUI ist diese Liste zunächst leer.

Hauptfenster

Zunächst werden die Funktionen, die im Hauptfenster angeboten werden, beschrieben:

Site hinzufügen: Um eine neue Site zur Liste der Sites hinzuzufügen, muss die Schaltfläche „Neu" ausgewählt werden. Daraufhin öffnet sich das Fenster „Neue Site". In

[3] `http://www-ds.e-technik.uni-dortmund.de/~pg505/content/docs/` (08/2007)

diesem Fenster muss der Benutzer den Namen für die neue Site eingeben. Daraufhin wird die neue Site zur Liste der Sites hinzugefügt und es öffnet sich der Dialog, in dem die neue Site bearbeitet werden kann.

Site bearbeiten: Um eine Site zu bearbeiten, muss die Site erst in der Liste markiert und dann die Schaltfläche „Bearbeiten“ ausgewählt werden. Daraufhin erscheint der Site-Dialog (vgl. Abbildung A.2).

Site entfernen: Um eine vorhandene Site aus der Liste der Sites zu entfernen, muss zuerst eine Site in der Liste markiert werden. Wenn daraufhin die Schaltfläche „Entfernen“ ausgewählt wird, wird die markierte Site aus der Liste der Sites entfernt.

Programm beenden: Um das Programm zu beenden, muss die Schaltfläche „Schließen“ ausgewählt werden. Daraufhin wird das Programm beendet und alle Änderungen, die an der Konfiguration vorgenommen wurden, gehen verloren, sofern sie nicht vorher gespeichert wurde.

Konfiguration speichern: Die Konfiguration kann in einer XML-Datei abgespeichert werden. Dafür muss die Schaltfläche „Speichern“ ausgewählt werden. Daraufhin erscheint der standard Dateidialog. Nachdem der Name für die Datei vergeben wurde, wird die Konfiguration gespeichert.

Konfiguration laden: Es besteht die Möglichkeit, eine vorhandene Konfiguration aus einer XML-Datei zu laden. Dazu muss die Schaltfläche „Laden“ ausgewählt werden. Daraufhin öffnet sich der standard Dateidialog. In diesem kann die Datei ausgewählt werden, aus der die Konfiguration eingelesen werden soll. Dabei geht die aktuelle Konfiguration im Programm verloren, sofern sie nicht vorher abgespeichert worden ist.

Logging: Logging ist ein Pfad zu einer Log-Datei, in der die für eine Auswertung benötigte Daten geloggt werden. Es ist nicht nötig, die Log-Datei vorher zu erzeugen, sie wird bei der Simulation automatisch erstellt. Der Benutzer kann den Pfad auswählen und dann einen Namen für die Log-Datei eingeben.

Base-Class-Path: Der Base-Class-Path ist der Pfad zu dem Verzeichnis, in dem sich das Java-Paket für die Gridimplementierung befindet. In dem Verzeichnis befindet sich das Hauptpaket „edu“. Die GUI durchsucht dieses Paket nach Klassendateien, die für die Konfiguration benötigt werden. Die GUI merkt sich den zuletzt ausgewählten Pfad in einer eigenen Konfigurationsdatei.

Der Pfad zum Grid-Paket muss vom Benutzer ausgewählt werden, damit die GUI in den Auswahlmenüs des Site-Dialogs die passenden Klassen anzeigen kann.

Site-Dialog

Der Site-Dialog (vgl. Abbildung A.2) dient dazu, die Konfiguration einer einzelnen Site zu bearbeiten. Oben im Dialog steht der Name der Site, die gerade bearbeitet wird. Auf der

linken Seite befinden sich die Registerkarten, mit denen die einzelnen Komponenten, aus denen eine Site besteht, konfiguriert werden können. Auf der rechten Seite befindet sich der Inhalt der gerade ausgewählten Registerkarte. Folgende Komponente können in diesem Dialog konfiguriert werden: „Kommunikationsdienst“, „Informationsdienst“, „ODIN“, „Scheduling-Algorithmus“ und „Schedule“.
Im Folgenden werden die einzelnen Registerkarten beschrieben.

Registerkarte „Site“: Diese Registerkarte erlaubt es, die Anzahl der Prozessoren, sowie die IP-Adresse der Site zu konfigurieren. In diesem Dialog muss der Benutzer zusätzlich die Workload-Datei auswählen, die den Trace von einer Grid-Site enthält. Dieser Trace wird für die Simulation der Site benutzt. Die Anzahl der Prozessoren hängt vom ausgewählten Trace ab.

Registerkarten für Sitekomponenten: Jede dieser Registerkarten entspricht einer Gridkomponente. In jeder Registerkarte kann der Benutzer in einem Auswahlmenü die Klasse auswählen, die für die entsprechende Komponente in der Gridsimulation verwendet werden soll. Darüber hinaus kann eine ausgewählte Klasse individuell konfiguriert werden, sofern für diese Klasse ein Konfigurationsdialog existiert. Falls es ihn gibt, wird er dynamisch geladen und angezeigt.

In einem Auswahlmenü werden nicht alle Klassen aus dem Grid-Paket angezeigt, sondern nur solche Klassen, die eine bestimmte Schnittstelle besitzen und sich in einem bestimmten Unterpaket befinden. So werden z.B. in der Registerkarte „Kommunikationsdienst“ nur solche Klassen angezeigt, die die Schnittstelle `ICommunicationService` implementieren und sich im Unterpaket `edu.udo.pg505.com` befinden. So kann der Benutzer nur die passenden Klassen für die Konfiguration des „Kommunikationsdienstes“ auswählen.

Damit dies funktioniert, muss der GUI der Pfad zum Java-Paket, in dem sich das Programm befindet, bekannt sein. Deswegen sollte im Hauptfenster der „Base-Class-Path“ ausgewählt werden, bevor eine Site konfiguriert wird.

A.3 Tools

Es existieren mehrere Werkzeuge, die dem Anwender der Simulationsumgebung eine effiziente Nutzung ermöglichen und ihn bei der Auswertung der Simulationsdaten unterstützen. Auf Grund der hohen Flexibilität und der guten potentiellen Anpassbarkeit durch den Anwender sind die meisten dieser Werkzeuge als Perl-Skripte implementiert worden. Folgende Werkzeuge stehen zur Verfügung:

- `Evaluation.jar`: Auswertung der von der Simulation generierten Logdateien und Berechnung der im Kapitel 6.2 definierten Metriken.
- `start_eval.pl`: Startet die Simulation und die Auswertung der Simulationsdaten.
- `differences.pl`: Ermöglicht Vergleiche zwischen verschiedenen Simulationsläufen.

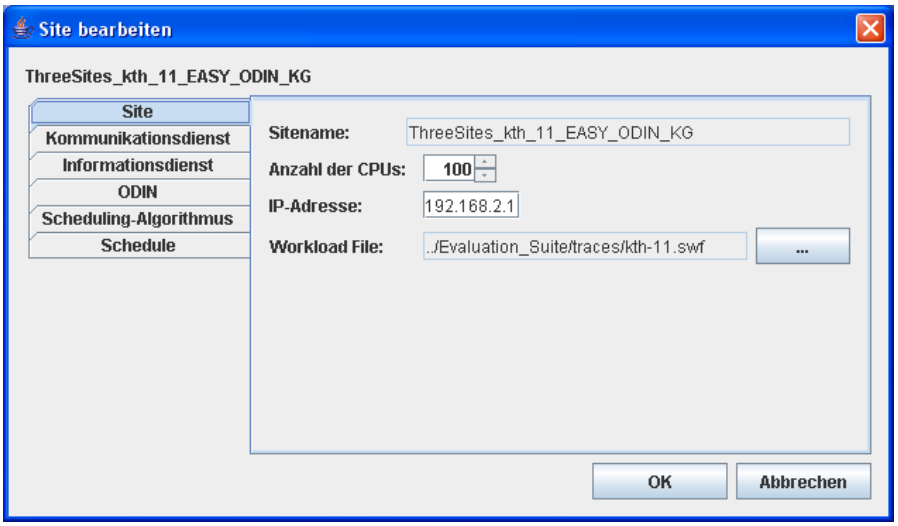


Abbildung A.2: Der Site-Dialog

- `ppr.pl`: Darstellung der Simulationsergebnisse in verschiedenen Formaten.
- `pdp.pl`: Darstellung der Differenzen beim Vergleich von Simulationsergebnissen mittels `differences.pl` in verschiedenen Formaten.

Die gemeinsame Konfiguration (Pfadangaben, Programmparameter etc.) der Perl-Skripte erfolgt in der Datei `config.pl`. Für Windows- und Linux-Systeme stehen jeweils Beispielkonfigurationen in den Dateien `config.pl_linux` und `config.pl_windows` zur Verfügung (siehe auch Abschnitt A.1).

A.3.1 Evaluation.jar

Dieses Java-Programm wertet die von einer Simulation erzeugten Logdateien aus und berechnet verschiedene Metriken (vgl. Kapitel 6.2 und 6.3).
Dem Programm wird beim Aufruf eine XML-Datei übergeben, in der alle für die jeweilige Auswertung benötigten Pfade eingetragen sind. Für jedes auszuwertende Experiment wird hierbei eine eigene XML-Datei benötigt. Die XML-Datei ist denkbar einfach aufgebaut; es sind lediglich drei Tags zu spezifizieren:

- `<Archive></Archive>`: Dient zur Angabe der von der Simulation erzeugten Logdatei im Standard Workload Format.
- `<ResultDir></ResultDir>`: Gibt das Verzeichnis an, in dem die Ausgabedateien abgelegt werden sollen.
- `<SiteSetup></SiteSetup>`: Spezifiziert die zur Simulation verwendete Site-Setup-Datei.

Das `Evaluation.jar` wird normalerweise vom Benutzer nicht separat aufgerufen. In der Regel wird das Programm durch das `start_eval.pl`-Skript gestartet, welches auch die Erzeugung der XML-Datei übernimmt.

A.3.2 start_eval.pl

Dieses Programm stellt die zentrale Komponente des Werkzeug-Sets dar und hat drei Hauptaufgaben:

- Starten der Simulationsumgebung und Ausführung der Simulation.
- Auswertung der Simulations-Logs mittels `Evaluation.jar` (siehe Abschnitt A.3.1).
- Auswertung der erhaltenen Ergebnisse in Form von Plots und Zusammenfassung der Ergebnisse.

Die Aufrufparameter des Programms:

```
Usage:  ./start_eval.pl [Options] -f <XML file>
        -f <file>:      Specify site-setup file that should be processed.

Options:
        -h:             Show this message and exit.
        -v:             Show program version and exit.
        -e:             Evaluation and plotting only. Skip simulation.
        -p:             Plot only. Skip simulation and evaluation.
        -n:             Do not create plots.
        -x:             Do not compress results.
```

Die Parameter `-h` und `-v` dienen zur Anzeige eines kurzen Hilfetextes bzw. der Programmversion. Mit dem Parameter `-f` wird die Site-Setup-Datei spezifiziert, die das zu simulierende Setup beschreibt. Der Dateiname kann dabei entweder als absoluter Pfad oder relativ zum Speicherort des Skriptes angegeben werden. Die optionalen Parameter `-e`, `-p` und `-n` dienen dazu, einzelne Arbeitsschritte des Programms zu überspringen. Mittels `-e` lässt sich die Ausführung der Simulation unterbinden, für den Fall, dass die Simulationsdaten bereits vorliegen. Mit `-p` wird zusätzlich der Aufruf der Evaluationskomponente `Evaluation.jar` übersprungen und es werden lediglich die Plots erzeugt. Der Parameter `-n` verhindert die Erzeugung der Plot-Grafiken und mit `-x` wird das Komprimieren der Ergebnisdateien verhindert.

Die Ergebnisse der Simulation werden in einem Verzeichnis unterhalb des in der Variable `$resultdir`[4] angegebenen Verzeichnisses (standardmäßig `results`) abgelegt, welches den Namen der Site-Setup-Datei trägt. Falls der Parameter `-x` nicht angegeben wurde, wird außerdem eine komprimierte Kopie dieses Verzeichnisses in `results` erzeugt.

[4] Konfigurierbar in der Datei config.pl

A.3.3 differences.pl

Mit Hilfe dieses Programms lassen sich die Ergebnisse zweier Simulationsläufe miteinander vergleichen. Dabei dienen die Ergebnisse eines Simulationslaufes als Referenz und die Ergebnisse des anderen Laufes werden dazu in Beziehung gesetzt. Das Programm erzeugt anschließend ein „Bar Chart“, in dem die Abweichungen visualisiert werden. Auch bei diesem Programm werden die erzeugten Dateien unterhalb des in Abschnitt A.3.2 beschriebenen Verzeichnisses abgelegt.

```
Usage:  ./differences.pl [Options] -s <file> -r <file>
        -r <file>:      Specify reference site-setup file.
        -s <file>:      Specify site-setup file.

Options:
        -h:             Show this message and exit.
        -v:             Show program version and exit.
        -d:             Enable debug mode.
        -x:             Do not compress results.
        -o <file>:      Specify file name suffix for output files.
```

A.3.4 ppr.pl

Mit ppr.pl („ppr“ steht für „pretty print results“) lassen sich die Daten der von `Evaluation.jar` (siehe Abschnitt A.3.1) erzeugten Ausgabedatei zusammenfassen und in verschiedenen Formaten ausgeben. Die unterstützten Formate umfassen:

- Wiki-Formatierung für das Trac[5]-Wiki
- Kommaseparierte Werte (CSV)
- LaTeX-Format
- Microsoft Excel 97

```
Usage:  ./ppr.pl  [Options] -s <file>
        -s      site-setup file

Options:
        -w      Output in Trac WikiFormatting.
        -c      Output in CSV format
        -l      Output in LaTeX format
        -x      Output in MS Excel format
If no options are specified, all possible variants are created.
```

Dieses Skript wird standardmäßig von `start_eval.pl` (Abschnitt A.3.2) aufgerufen, kann aber auch unabhängig davon verwendet werden.

[5] `http://trac.edgewall.org` (08/2007)

Die Parameter `-w`, `-c`, `-l` und `-x` können dazu verwendet werden, die Ausgabe auf ein bestimmtes Format zu beschränken. Hierbei erzeugt `-w` eine Ausgabe in Wiki-Formatierung passend für die Wiki-Komponente von Trac, `-c` eine Datei im Character Separated Values (CSV)-Format, `-l` eine Datei im LaTeX-Format und `-x` erzeugt ein Spreadsheet kompatibel zu Microsoft Excel 97. Wird keiner dieser Parameter angegeben, so werden alle möglichen Ausgabedateien erzeugt.

A.3.5 pdp.pl

Dieses Programm ist von seiner Funktionsweise angelehnt an `ppr.pl` (Abschnitt A.3.4) und erstellt Tabellen in denen die prozentualen Abweichungen der berechneten Metriken zwischen einer Referenz-Site und einer Vergleichs-Site dargestellt sind. Dieses Skript wird normalerweise von `differences.pl` (siehe Abschnitt A.3.3) aufgerufen. Als Ausgabeformate sind Dateien im Trac Wiki-Format und im CSV-Format fest eingestellt.

```
Usage:  ./pdp.pl  [Options] -s <SiteSetup.xml>
        -s      SiteSetup XML File

Options:
        -o      File name suffix for difference files.
        -h      Show this message and exit.
        -v      Show program version and exit.
```

Mit dem Parameter `-s` wird hier der Pfad zu derjenigen Site-Setup-Datei angegeben, die die zu vergleichende Site beschreibt. Der optionale Parameter `-o` erlaubt es, ein Suffix für die Ausgabedateien anzugeben.

A.4 Ergebnistabellen der Experimente

Nachfolgend diverse Anhänge zu den in Kapitel 7 durchgeführten Experimenten:

A.4.1 Experiment mit 3 kleinen Sites und egoistischem Verhalten einer Site

Dieser Anhangabschnitt liefert alle Ergebnisdaten zum Versuch in Unterkapitel 7.3.1. Dazu gehören sowohl alle lokalen und globalen Metriken als auch die Austauschmatrizen für ausgetauschte Jobs und Arbeit zwischen den einzelnen Sites sowie auch die Verhältnisse zwischen den zum Austausch angefragten Jobs und wirklich ausgetauschten Jobs.

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Alle ODIN_{OL}								
KTH	15.9804	68.7155	75157.6314	75157.6314	1.0000	1.0155	1.0155	-1.0000
CTC	65.6995	65.6995	52937.9601	52937.9601	1.0000	0.9698	0.9698	-1.0000
SDSC00	22.0094	73.9377	73605.8602	73605.8602	1.0000	1.0928	1.0928	-1.0000
Alle $\text{ODIN}_{\text{AWFB}}$								

Tabelle A.1 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
KTH	15.9804	67.9233	61264.4200	60531.5123	0.6257	1.0036	1.0036	80.8272
Δ zu ODIN_{OL}	-	-1.1664	-22.6775	-24.1628	-	-	-	-
CTC	65.6995	68.6727	51877.3016	51806.7493	0.9118	1.0136	1.0136	57.9143
Δ zu ODIN_{OL}	-	4.3295	-2.0446	-2.1835	-	-	-	-
SDSC00	22.0404	64.4818	47139.2725	47383.1845	0.7022	0.9517	0.9517	71.2361
Δ zu ODIN_{OL}	-	-14.6644	-56.1455	-55.3417	-	-	-	-
KTH mit $\text{ODIN}_{\text{AWFE}}$								
KTH-Ego	15.9804	52.1661	57317.8583	59072.7141	0.7592	0.7707	0.7707	73.1130
Δ zu ODIN_{OL}	-	-31.7244	-31.1243	-27.2290	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-30.2058	-6.8854	-2.4695	-	-	-	-
CTC	65.6995	71.1209	53288.8735	52730.6543	0.9378	1.0493	1.0493	40.9739
Δ zu ODIN_{OL}	-	7.6227	0.6585	-0.3931	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.4423	2.6489	1.7521	-	-	-	-
SDSC00	22.0404	68.7831	50303.4458	51729.7558	0.7198	1.0152	1.0152	55.6673
Δ zu ODIN_{OL}	-	-7.4941	-46.3237	-42.2892	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	6.2533	6.2902	8.4025	-	-	-	-
CTC mit $\text{ODIN}_{\text{AWFE}}$								
KTH	15.9804	78.0543	79875.9826	76055.3523	0.8267	1.1532	1.1532	44.0445
Δ zu ODIN_{OL}	-	11.9645	5.9071	1.1804	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	12.9795	23.3006	20.4112	-	-	-	-
CTC-Ego	65.6995	62.1644	50623.2282	50336.8876	0.9462	0.9174	0.9174	46.0992
Δ zu ODIN_{OL}	-	-5.6867	-4.5725	-5.1673	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-10.4695	-2.4773	-2.9200	-	-	-	-
SDSC00	22.0265	78.5537	71814.5773	71593.9153	0.8635	1.1602	1.1602	37.0578
Δ zu ODIN_{OL}	-	5.8762	-2.4943	-2.8102	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	17.9137	34.3597	33.8167	-	-	-	-
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$								
KTH	15.9804	69.9925	63171.1145	61086.8172	0.6868	1.0341	1.0341	61.7333
Δ zu ODIN_{OL}	-	1.8244	-18.9747	-23.0341	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	2.9563	3.0183	0.9090	-	-	-	-
CTC	65.6995	70.8068	53467.7721	53255.5637	0.9508	1.0451	1.0451	35.4920
Δ zu ODIN_{OL}	-	7.2130	0.9909	0.5964	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.0139	2.9746	2.7205	-	-	-	-
SDSC00-Ego	22.0404	55.6948	45617.2016	45811.8426	0.7522	0.8220	0.8220	70.0815
Δ zu ODIN_{OL}	-	-32.7553	-61.3555	-60.6699	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-15.7772	-3.3366	-3.4300	-	-	-	-

Tabelle A.1: Lokale Metriken für das Experiment mit drei kleinen Sites

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
Alle ODIN_{OL}	60760.0598	-1.0000	-1.0000	-1.0000	-1.0000	67.6574

Tabelle A.2 – Fortsetzung

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
Alle $\text{ODIN}_{\text{AWFB}}$	52318.4494	67.2639	8921.8977	11.0805	101858.2097	67.6825
Δ zu ODIN_{OL}	-13.8934	-	-	-	-	0.0372
KTH-Site mit $\text{ODIN}_{\text{AWFE}}$	53275.8658	49.9637	9355.7074	11.5634	107891.1247	67.6825
Δ zu alle ODIN_{OL}	-12.3176	-	-	-	-	0.0372
Δ zu alle $\text{ODIN}_{\text{AWFB}}$	1.8300	-	-	-	-	0.0000
CTC-Site mit $\text{ODIN}_{\text{AWFE}}$	59641.7770	42.6166	9078.3970	7.4278	69173.1509	67.6825
Δ zu alle ODIN_{OL}	-1.8405	-	-	-	-	0.0372
Δ zu alle $\text{ODIN}_{\text{AWFB}}$	13.9976	-	-	-	-	0.0000
SDSC00-Site mit $\text{ODIN}_{\text{AWFE}}$	53296.0284	48.8599	9171.8098	10.6061	99086.2132	67.6825
Δ zu alle ODIN_{OL}	-12.2844	-	-	-	-	0.0372
Δ zu alle $\text{ODIN}_{\text{AWFB}}$	1.8685	-	-	-	-	0.0000

Tabelle A.2: Globale Metriken für das Experiment mit drei kleinen Sites

$\mathcal{M}$	CTC	KTH	SDSC00
Alle $\text{ODIN}_{\text{AWFB}}$			
CTC	87.2887	7.9510	4.7606
KTH	21.9249	72.8958	5.1794
SDSC00	7.4673	9.9128	82.6199
CTC mit $\text{ODIN}_{\text{AWFE}}$			
CTC-Ego	91.5115	5.5494	2.9393
KTH	0.0000	79.4199	20.5808
SDSC00	0.0000	15.3204	84.6796
KTH mit $\text{ODIN}_{\text{AWFE}}$			
CTC	89.8418	0.0000	10.1586
KTH-Ego	11.2960	86.3268	2.3773
SDSC00	15.9644	0.0000	84.0356
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	91.1048	8.8953	0.0000
KTH	22.9362	77.0638	0.0000
SDSC00-Ego	6.4039	7.1587	86.4374

Tabelle A.3: Austauschmatrizen für migrierte Jobs in Prozent für drei kleine Sites

$\mathcal{M}^R$	CTC	KTH	SDSC00
Alle $\text{ODIN}_{\text{AWFB}}$			
CTC	90.9812	4.6086	4.4181
KTH	30.7775	62.6972	6.5503
SDSC00	17.7967	11.1073	71.0960
CTC mit $\text{ODIN}_{\text{AWFE}}$			
CTC-Ego	94.9611	2.7169	2.3264
KTH	0.0000	82.9358	17.1295
SDSC00	0.0000	13.5794	86.4206
KTH mit $\text{ODIN}_{\text{AWFE}}$			
CTC	93.9559	0.0000	6.0512
KTH-Ego	20.7811	75.9207	3.3109
SDSC00	26.6999	0.0000	73.3001
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	95.2926	4.7101	0.0000
KTH	31.1383	68.8617	0.0000
SDSC00-Ego	15.2970	8.8908	75.8123

Tabelle A.4: Austauschmatrizen für migrierte Arbeit in Prozent für drei kleine Sites

$\mathcal{M}$	CTC	KTH	SDSC00	Durchschnitt
Alle $\text{ODIN}_{\text{AWFB}}$				
CTC	0.0000	36.3324	21.6891	29.0108
KTH	65.3822	0.0000	15.4450	40.4136
SDSC00	30.6064	41.1331	0.0000	35.8697
CTC mit $\text{ODIN}_{\text{AWFE}}$				
CTC-Ego	0.0000	30.2414	15.9620	23.1017
KTH	0.0000	0.0000	44.0445	22.0222
SDSC00	0.0000	37.3426	0.0000	18.6713
KTH mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	0.0000	40.9739	20.4870
KTH-Ego	60.4018	0.0000	12.7112	36.5565
SDSC00	55.6673	0.0000	0.0000	27.8337
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	35.5858	0.0000	17.7929

Tabelle A.5 – Fortsetzung				
$\mathcal{M}$	CTC	KTH	SDSC00	Durchschnitt
KTH	61.7333	0.0000	0.0000	30.8666
SDSC00-Ego	33.0907	37.5374	0.0000	35.3140

Tabelle A.5: Verhältnis von Anfragen und wirklich ausgetauschten Jobs für 3 kleine-Site-Experiment

A.4.2 Experiment mit 3 großen Sites und egoistischem Verhalten einer Site

Dieser Anhang-teil liefert alle Ergebnisdaten zum Versuch in Unterkapitel 7.3.2. Dazu gehören sowohl alle lokalen und globalen Metriken als auch die Austauschmatrizen für ausgetauschte Jobs und Arbeit zwischen den einzelnen Sites sowie auch die Verhältnisse zwischen den zum Austausch angefragten Jobs und wirklich ausgetauschten Jobs.

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Alle ODIN_{OL}								
CTC	16.9776	65.6995	52937.9601	52937.9601	1.0000	1.0277	1.0277	-1.0000
SDSC03	47.5875	68.7376	50772.4765	50772.4765	1.0000	1.0778	1.0778	-1.0000
SDSC05	60.1680	60.1680	54953.8424	54953.8424	1.0000	0.9410	0.9410	-1.0000
Alle $\text{ODIN}_{\text{AWFB}}$								
CTC	16.9776	65.0164	43344.8372	42861.3756	0.6430	1.0168	1.0168	88.3772
Δ zu ODIN_{OL}	-	-1.0507	-22.1321	-23.5097	-	-	-	-
SDSC03	47.6038	66.0260	42661.8674	44036.6221	0.7676	1.0350	1.0350	67.6120
Δ zu ODIN_{OL}	-	-4.1067	-19.0114	-15.2960	-	-	-	-
SDSC05	60.1758	62.2257	51008.3170	49851.9301	0.8658	0.9727	0.9727	55.5271
Δ zu ODIN_{OL}	-	3.3069	-7.7351	-10.2341	-	-	-	-
CTC mit $\text{ODIN}_{\text{AWFE}}$								
CTC	16.9776	51.2412	42272.8966	41464.9468	0.7799	0.8016	0.8016	87.5036
Δ zu ODIN_{OL}	-	-28.2162	-25.2291	-27.6692	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-26.8830	-2.5358	-3.3677	-	-	-	-
SDSC03	47.5908	68.0323	44661.1505	45866.9221	0.7865	1.0667	1.0667	55.4289
Δ zu ODIN_{OL}	-	-1.0366	-13.6838	-10.6952	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	2.9490	4.4766	3.9905	-	-	-	-
SDSC05	60.1671	64.3742	52413.5266	51047.2507	0.8823	1.0067	1.0067	42.9660
Δ zu ODIN_{OL}	-	6.5340	-4.8467	-7.6529	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.3374	2.6810	2.3416	-	-	-	-
SDSC03 mit $\text{ODIN}_{\text{AWFE}}$								
CTC	16.9771	68.8092	47353.0305	48107.1470	0.7349	1.0759	1.0759	68.5775
Δ zu ODIN_{OL}	-	4.5193	-11.7942	-10.0418	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	5.5122	8.4645	10.9043	-	-	-	-
SDSC03	47.6757	57.3517	40689.5459	40679.4636	0.8328	0.8976	0.8976	61.9070

Tabelle A.6 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Δ zu ODIN_{OL}	-	-19.8527	-24.7802	-24.8111	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-15.1248	-4.8472	-8.2527	-	-	-	-
SDSC05	60.1758	67.2166	58413.2647	56253.8911	0.9449	1.0516	1.0516	29.1421
Δ zu ODIN_{OL}	-	10.4864	5.9223	2.3110	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	7.4250	12.6768	11.3805	-	-	-	-
SDSC05 mit $\text{ODIN}_{\text{AWFE}}$								
CTC	16.9776	71.4832	49216.5882	49208.3280	0.7763	1.1185	1.1185	61.4872
Δ zu ODIN_{OL}	-	8.0909	-7.5612	7.5793	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	9.0466	11.9304	12.8981	-	-	-	-
SDSC03	47.6221	74.8431	53038.3543	52837.0246	0.9284	1.1727	1.1727	33.3420
Δ zu ODIN_{OL}	-	8.1578	4.2721	3.9074	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	11.7807	19.5641	16.6557	-	-	-	-
SDSC05	60.1758	54.4177	48900.2004	48766.4608	0.9043	0.8507	0.8507	48.1038
Δ zu ODIN_{OL}	-	-10.5670	-12.3796	-12.6878	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-14.3484	-4.3111	-2.2259	-	-	-	-

Tabelle A.6: Lokale Metriken für das Experiment mit drei großen Sites

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
Alle ODIN_{OL}	53080.8027	-1.0000	-1.0000	-1.0000	-1.0000	63.7735
Alle $\text{ODIN}_{\text{AWFB}}$	46775.6169	68.9283	7751.4466	49.0833	396990.8992	63.7952
Δ zu ODIN_{OL}	-11.8785	-	-	-	-	0.0341
CTC-Site mit $\text{ODIN}_{\text{AWFE}}$	48071.9070	54.9466	7073.9540	54.8661	439649.2202	63.7779
Δ zu alle ODIN_{OL}	-9.4364	-	-	-	-	0.0069
Δ zu alle $\text{ODIN}_{\text{AWFB}}$	2.7713	-	-	-	-	-0.0272
SDSC03-Site mit $\text{ODIN}_{\text{AWFE}}$	50132.5762	48.4729	7630.2023	38.4756	316074.8394	63.8916
Δ zu alle ODIN_{OL}	-5.5542	-	-	-	-	0.1852
Δ zu alle $\text{ODIN}_{\text{AWFB}}$	7.1767	-	-	-	-	0.1510
SDSC05-Site mit $\text{ODIN}_{\text{AWFE}}$	50526.0773	46.4279	7509.2468	30.9404	242498.6732	63.8198
Δ zu alle ODIN_{OL}	-4.8129	-	-	-	-	0.0726
Δ zu alle $\text{ODIN}_{\text{AWFB}}$	8.0180	-	-	-	-	0.0385

Tabelle A.7: Globale Metriken für das Experiment mit drei großen Sites

$\mathcal{M}$	CTC	SDSC03	SDSC05
Alle $\text{ODIN}_{\text{AWFB}}$			
CTC	87.2887	7.9510	4.7606
KTH	21.9249	72.8958	5.1794
SDSC00	7.4673	9.9128	82.6199

Tabelle A.8 – Fortsetzung			
$\mathcal{M}$	CTC	SDSC03	SDSC05
CTC mit $\text{ODIN}_{\text{AWFE}}$			
CTC-Ego	92.1087	6.4302	1.4612
SDSC03	0.0000	86.5028	13.4972
SDSC05	0.0000	11.2212	88.7788
SDSC03 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	85.7744	0.0000	14.2258
SDSC03-Ego	4.8975	90.1500	4.9524
SDSC05	9.3014	0.0000	90.6986
SDSC05 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	85.3042	14.6960	0.0000
SDSC03	11.7529	88.2471	0.0000
SDSC05-Ego	4.9811	4.4684	90.5504

Tabelle A.8: Austauschmatrizen für migrierte Jobs in Prozent für drei große Sites

$\mathcal{M}^R$	CTC	SDSC03	SDSC05
Alle $\text{ODIN}_{\text{AWFB}}$			
CTC	85.0089	11.8578	3.1335
SDSC03	7.4134	85.2052	7.3814
SDSC05	7.2480	5.7955	86.9565
CTC mit $\text{ODIN}_{\text{AWFE}}$			
CTC-Ego	76.6820	16.3824	6.9493
SDSC03	0.0000	79.5722	20.4278
SDSC05	0.0000	11.2519	88.7481
SDSC03 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	72.7698	0.0000	27.2462
SDSC03-Ego	3.7136	83.8471	12.4394
SDSC05	5.1371	0.0000	94.8629
SDSC05 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	76.9479	23.0656	0.0000
SDSC03	6.7394	93.2606	0.0000
SDSC05-Ego	2.9679	6.1494	90.8827

Tabelle A.9 – Fortsetzung			
$\mathcal{M}^R$	CTC	SDSC03	SDSC05

Tabelle A.9: Austauschmatrizen für migrierte Arbeit in Prozent für große kleine Sites

$\mathcal{M}$	CTC	SDSC03	SDSC05	Durchschnitt
Alle $\text{ODIN}_{\text{AWFB}}$				
CTC	0.0000	69.9045	18.4727	44.1886
SDSC03	36.2619	0.0000	33.7328	34.9974
SDSC05	34.1060	24.6718	0.0000	29.3889
CTC mit $\text{ODIN}_{\text{AWFE}}$				
CTC-Ego	0.0000	71.3014	16.2022	43.7518
SDSC03	0.0000	0.0000	55.4289	27.7145
SDSC05	0.0000	42.9660	0.0000	21.4830
SDSC03 mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	0.0000	68.5775	34.2887
SDSC03-Ego	33.5247	0.0000	31.1260	32.3254
SDSC05	31.4836	0.0000	0.0000	15.7418
SDSC05 mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	61.4872	0.0000	30.7436
SDSC03	34.8479	0.0000	0.0000	17.4239
S SSC05-Ego	28.1055	22.7470	0.0000	25.4263

Tabelle A.10: Verhältnis von Anfragen und wirklich ausgetauschten Jobs für 3 große-Site-Experiment

A.4.3 Experiment mit 5 Sites und egoistischem Verhalten einer Site

Dieser Anhang-teil liefert alle Ergebnisdaten zum Versuch in Unterkapitel 7.3.3. Dazu gehören sowohl alle lokalen und globalen Metriken als auch die Austauschmatrizen für ausgetauschte Jobs und Arbeit zwischen den einzelnen Sites sowie auch die Verhältnisse zwischen den zum Austausch angefragten Jobs und wirklich ausgetauschten Jobs.

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Alle ODIN_{OL}								
KTH	4.1295	68.7155	75157.6314	75157.6314	1.0000	1.0662	1.0662	-1.0000
SDSC00	5.6875	73.9377	73605.8602	73605.8602	1.0000	1.1475	1.1475	-1.0000
CTC	16.9776	65.6995	52937.9601	52937.9601	1.0000	1.0197	1.0197	-1.0000
SDSC03	47.5875	68.7376	50772.4765	50772.4765	1.0000	1.0694	1.0694	-1.0000

Tabelle A.11 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
SDSC05	60.1680	60.1680	54953.8424	54953.8424	1.0000	0.9336	0.9336	-1.0000
Alle $\text{ODIN}_{\text{AWFB}}$								
KTH	4.1295	66.3107	53068.5629	52681.7561	0.4760	1.0287	1.0287	97.8480
Δ zu ODIN_{OL}	-	-3.6267	-41.6236	-42.6635	-	-	-	-
SDSC00	5.6923	66.1882	41638.1234	41359.0784	0.5098	1.0270	1.0270	96.0790
Δ zu ODIN_{OL}	-	-11.7083	-76.7752	-77.9678	-	-	-	-
CTC	16.9776	67.1014	43850.7403	43657.4458	0.6408	1.0409	1.0409	91.2447
Δ zu ODIN_{OL}	-	2.0892	-20.7231	-21.2576	-	-	-	-
SDSC03	47.6742	66.5080	43049.8309	44230.1898	0.7640	1.0329	1.0329	70.1631
Δ zu ODIN_{OL}	-	-3.3523	-17.9389	-14.7915	-	-	-	-
SDSC05	60.1758	62.1461	51049.8355	50024.2336	0.8571	0.9640	0.9640	59.3861
Δ zu ODIN_{OL}	-	3.1831	-7.6474	-9.8544	-	-	-	-
KTH-Ego								
KTH-Ego	4.1295	47.6233	52414.4578	55532.6615	0.6930	0.7388	0.7388	97.4259
Δ zu ODIN_{OL}	-	-44.2898	-43.3910	-35.3395	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-39.2400	-1.2479	5.1337	-	-	-	-
SDSC00	5.6955	67.7145	41714.4386	42623.0177	0.5008	1.0507	1.0507	94.3332
Δ zu ODIN_{OL}	-	-9.1904	-76.4518	-72.6904	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	2.2541	0.1829	2.9654	-	-	-	-
CTC	16.9776	68.2410	43472.1788	43597.7043	0.6364	1.0595	1.0595	90.9144
Δ zu ODIN_{OL}	-	3.7243	-21.7743	-21.4237	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	1.6700	-0.8708	-0.1370	-	-	-	-
SDSC03	47.6384	66.7940	42456.4884	44254.0323	0.7563	1.0381	1.0381	68.9054
Δ zu ODIN_{OL}	-	-2.9098	-19.5871	-14.7296	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.4281	-1.3975	0.0539	-	-	-	-
SDSC05	60.1758	62.5989	50923.8241	49193.4058	0.8599	0.9710	0.9710	58.2203
Δ zu ODIN_{OL}	-	3.8833	-7.9138	-11.7098	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.7232	-0.2475	-1.6889	-	-	-	-
SDSC00-Ego								
KTH	4.1295	66.0591	53124.9215	51314.6008	0.4772	1.0249	1.0249	96.9817
Δ zu ODIN_{OL}	-	-4.0212	-41.4734	-46.4644	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.3808	0.1061	-2.6643	-	-	-	-
SDSC00-Ego	5.6955	49.9616	40700.4845	40831.5109	0.6748	0.7753	0.7753	93.4170
Δ zu ODIN_{OL}	-	-47.9892	-80.8476	-80.2673	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-32.4782	-2.3038	-1.2921	-	-	-	-
CTC	16.9776	67.5081	43609.2521	43415.2266	0.6391	1.0478	1.0478	90.3080
Δ zu ODIN_{OL}	-	2.6791	-21.3916	-21.9341	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.6025	-0.5538	-0.5579	-	-	-	-
SDSC03	47.5915	66.9580	43115.0113	44554.2103	0.7672	1.0417	1.0417	68.4748
Δ zu ODIN_{OL}	-	-2.6576	-17.7606	-13.9566	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.6721	0.1512	0.7273	-	-	-	-
SDSC05	60.1758	62.9146	51172.1151	49864.8800	0.8675	0.9761	0.9761	55.7972

Tabelle A.11 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Δ zu ODIN_{OL}	-	4.3657	-7.3902	-10.2055	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	1.2215	0.2390	-0.3196	-	-	-	-
CTC-Ego								
KTH	4.1295	66.5261	53947.8157	53447.7585	0.5064	1.0324	1.0324	95.1629
Δ zu ODIN_{OL}	-	-3.2912	-39.3154	-40.6189	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.3237	1.6298	1.4332	-	-	-	-
SDSC00	5.6817	66.6158	42741.0967	43555.9313	0.5244	1.0341	1.0341	92.9466
Δ zu ODIN_{OL}	-	-10.9912	-72.2133	-68.9916	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.6420	2.5806	5.0438	-	-	-	-
CTC-Ego	16.9776	51.0615	42116.0301	41483.3656	0.7772	0.7926	0.7926	91.0532
Δ zu ODIN_{OL}	-	-28.6673	-25.6955	-27.6125	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-31.4127	-4.1189	-5.2408	-	-	-	-
SDSC03	47.5637	68.7294	43944.6944	45135.3177	0.7759	1.0699	1.0699	62.3700
Δ zu ODIN_{OL}	-	-0.0119	-15.5372	-12.4895	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.2320	2.0363	2.0054	-	-	-	-
SDSC05	60.1682	64.6056	53265.4472	51739.9464	0.8765	1.0025	1.0025	51.6038
Δ zu ODIN_{OL}	-	6.8688	-3.1698	-6.2116	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.8069	4.1596	3.3160	-	-	-	-
SDSC03-Ego								
KTH	4.1295	64.7540	57035.7738	62864.1636	0.5604	1.0046	1.0046	92.1714
Δ zu ODIN_{OL}	-	-6.1178	-31.7728	-19.5556	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-2.4040	6.9557	16.1975	-	-	-	-
SDSC00	5.6946	66.7285	44045.3784	46328.6225	0.5562	1.0349	1.0349	88.4769
Δ zu ODIN_{OL}	-	-10.8039	-67.1137	-58.8777	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.8097	5.4654	10.7267	-	-	-	-
CTC	16.9776	69.3286	47561.3817	48595.1247	0.7049	1.0752	1.0752	79.8210
Δ zu ODIN_{OL}	-	5.2346	-11.3045	-8.9368	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.2126	7.8018	10.1609	-	-	-	-
SDSC03-Ego	47.6131	57.5668	41120.3122	41331.4585	0.8370	0.8952	0.8952	63.0068
Δ zu ODIN_{OL}	-	-19.4048	-23.4730	-22.8422	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-15.5319	-4.6924	-7.0134	-	-	-	-
SDSC05	60.1908	67.7815	56731.7098	54060.4578	0.9221	1.0514	1.0514	41.4696
Δ zu ODIN_{OL}	-	11.2324	3.1338	-1.6526	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	8.3140	10.0153	7.4661	-	-	-	-
SDSC05-Ego								
KTH	3.7847	60.5034	60292.0328	62422.6168	0.6051	1.1732	1.1732	84.5610
Δ zu ODIN_{OL}	-	-13.5731	-24.6560	-20.4013	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-9.5984	11.9808	15.6047	-	-	-	-
SDSC00	5.0827	63.2821	43376.1278	43090.5266	0.6318	1.2270	1.2270	90.5714
Δ zu ODIN_{OL}	-	-16.8383	-69.6921	-70.8168	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-4.5923	4.0068	4.0182	-	-	-	-
CTC	16.2460	67.0518	47288.0460	44601.3331	0.7585	1.3001	1.3001	81.8090

Tabelle A.11 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Δ zu ODIN_{OL}	-	2.0168	-11.9479	-18.6914	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.0739	7.2689	2.1163	-	-	-	-
SDSC03	48.9861	72.8654	53542.7437	54001.6692	0.9158	1.4128	1.4128	51.2527
Δ zu ODIN_{OL}	-	5.6650	5.1739	5.9798	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	8.7248	19.5973	18.0948	-	-	-	-
SDSC05-Ego	33.5821	31.4051	38726.8695	38420.9371	0.9352	0.6089	0.6089	51.3554
Δ zu ODIN_{OL}	-	-91.5864	-41.9011	-43.0310	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-97.8853	-31.8202	-30.2005	-	-	-	-

Tabelle A.11: Lokale Metriken für das Experiment mit fünf Sites

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
Alle ODIN_{OL}	54624.7927	-1.0000	-1.0000	-1.0000	-1.0000	64.2740
Alle $\text{ODIN}_{\text{AWFB}}$	46971.3130	78.3982	7772.8345	38.6305	316515.1344	64.3910
Δ zu ODIN_{OL}	-14.0110	-	-	-	-	0.1821
KTH-Site mit $\text{ODIN}_{\text{AWFE}}$	46640.1507	76.0062	7695.8023	40.6526	336601.6274	64.3427
Δ zu ODIN_{OL}	-14.6172	-	-	-	-	0.1069
Δ zu $\text{ODIN}_{\text{AWFB}}$	-0.7050	-	-	-	-	-0.0750
SDSC00-Site mit $\text{ODIN}_{\text{AWFE}}$	46980.7790	75.6311	7788.3839	39.0797	313454.6679	64.2794
Δ zu ODIN_{OL}	-13.9937	-	-	-	-	0.0084
Δ zu $\text{ODIN}_{\text{AWFB}}$	0.0202	-	-	-	-	-0.1734
CTC-Site mit $\text{ODIN}_{\text{AWFE}}$	48133.5610	70.5142	7524.0875	39.7735	321283.1963	64.2417
Δ zu ODIN_{OL}	-11.8833	-	-	-	-	-0.0501
Δ zu $\text{ODIN}_{\text{AWFB}}$	2.4744	-	-	-	-	-0.2318
SDSC03-Site mit $\text{ODIN}_{\text{AWFE}}$	49515.0037	65.7649	8079.4498	28.8752	242316.0177	64.3085
Δ zu ODIN_{OL}	-9.3543	-	-	-	-	0.0537
Δ zu $\text{ODIN}_{\text{AWFB}}$	5.4154	-	-	-	-	-0.1281
SDSC05-Site mit $\text{ODIN}_{\text{AWFE}}$	47735.4242	68.7633	7429.4529	24.3088	171059.6085	51.5726
Δ zu ODIN_{OL}	-12.6122	-	-	-	-	-19.7613
Δ zu $\text{ODIN}_{\text{AWFB}}$	1.6268	-	-	-	-	-19.9071

Tabelle A.12: Globale Metriken für das Experiment mit fünf Sites

$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00
Alle $\text{ODIN}_{\text{AWFB}}$					
CTC	84.3133	5.6595	1.8692	4.8719	3.2864
SDSC03	3.7890	84.5176	6.0914	3.2860	2.3163
SDSC05	4.2575	4.9905	85.9231	2.4231	2.4061
KTH	7.8093	3.5114	1.4502	71.4210	15.8087

Tabelle A.13 – Fortsetzung

$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00
SDSC00	8.8125	3.9349	1.4324	8.5072	77.3130
CTC-Ego					
CTC-Ego	91.8134	3.0571	1.0428	2.6011	1.4858
SDSC03	0.0000	85.2738	8.8436	3.3806	2.5023
SDSC05	0.0000	7.5925	86.3183	3.2963	2.7933
KTH	0.0000	9.1155	2.2964	72.9204	15.6682
SDSC00	0.0000	10.8185	2.3214	9.0070	77.8531
SDSC03-Ego					
CTC	83.4752	0.0000	8.3202	3.3317	4.8733
SDSC03-Ego	2.9611	89.9289	5.1461	0.7304	1.2336
SDSC05	7.0798	0.0000	86.5359	2.5500	3.8348
KTH	10.7272	0.0000	9.5649	73.8720	5.8361
SDSC00	9.8558	0.0000	9.1312	4.7870	76.2261
SDSC05-Ego					
CTC	83.7404	7.1961	0.0000	3.7416	5.3233
SDSC03	6.7633	85.7887	0.0000	3.0124	4.4369
SDSC05-Ego	1.4423	2.4062	95.2016	0.3870	0.5633
KTH	10.2276	10.5528	0.0000	72.7480	6.4726
SDSC00	8.8645	8.5409	0.0000	4.8966	77.6980
KTH-Ego					
CTC	83.9532	7.4328	2.0583	0.0000	6.5560
SDSC03	4.5865	84.6228	6.7379	0.0000	4.0531
SDSC05	5.0772	5.6059	85.9018	0.0000	3.4155
KTH-Ego	3.6518	1.8259	0.7269	86.8429	6.9527
SDSC00	16.6421	6.1154	1.5901	0.0000	75.6525
SDSC00-Ego					
CTC	83.4402	8.3564	2.5506	5.6530	0.0000
SDSC03	5.1552	84.1592	7.2152	3.4705	0.0000
SDSC05	5.3349	5.4404	86.5720	2.6528	0.0000
KTH	19.8181	6.4855	2.0155	71.6809	0.0000
SDSC00-Ego	5.3338	2.6233	0.8688	5.4076	85.7665

Tabelle A.13: Austauschmatrizen für migrierte Jobs in Prozent für fünf Sites

$\mathcal{M}^R$	CTC	SDSC03	SDSC05	KTH	SDSC00
Alle ODIN$_{\text{AWFB}}$					
CTC	62.8579	20.8534	9.9103	3.2133	3.1887
SDSC03	3.9214	76.9449	17.9678	0.5620	0.6052
SDSC05	3.4610	8.2326	86.5215	0.8190	0.9666
KTH	18.3968	10.4577	4.6759	47.6596	18.8820
SDSC00	20.1573	13.1221	3.2099	10.2739	53.2369
CTC-Ego					
CTC-Ego	76.6670	13.2398	6.4217	1.8393	1.8469
SDSC03	0.0000	78.6254	20.0898	0.6251	0.6612
SDSC05	0.0000	9.5071	88.3377	1.0262	1.1299
KTH	0.0000	23.9173	6.2155	50.9441	18.9955
SDSC00	0.0000	26.0926	7.9718	11.3404	54.5952
SDSC03-Ego					
CTC	69.6018	0.0000	24.3855	2.4021	3.6305
SDSC03-Ego	2.5005	83.8917	13.1128	0.2121	0.2835
SDSC05	4.8010	0.0000	92.9216	0.9520	1.3265
KTH	14.9240	0.0000	20.1653	56.3777	8.5656
SDSC00	15.6042	0.0000	21.2824	5.7440	57.3694
SDSC05-Ego					
CTC	73.7788	18.0923	0.0000	3.3578	4.8455
SDSC03	7.1802	90.9201	0.0000	0.7671	1.1405
SDSC05-Ego	1.3544	4.3709	93.8455	0.1708	0.2596
KTH	15.7720	20.2107	0.0000	58.8965	5.2015
SDSC00	13.9892	17.1177	0.0000	5.3428	63.5503
KTH-Ego					
CTC	62.3866	23.4265	9.4215	0.0000	4.7902
SDSC03	4.4061	76.7050	18.0637	0.0000	0.8261
SDSC05	3.7948	8.4958	86.5280	0.0000	1.1824
KTH-Ego	10.7025	8.2391	2.9557	68.7279	9.4106
SDSC00	28.5199	15.2148	4.7630	0.0000	51.5022
SDSC00-Ego					
CTC	63.5511	22.8965	10.0636	3.5102	0.0000
SDSC03	4.1448	76.8978	18.2873	0.6708	0.0000
SDSC05	3.7033	7.8163	87.5472	0.9332	0.0000
KTH	29.5082	16.8147	5.3390	48.3381	0.0000

Tabelle A.14 – Fortsetzung

$\mathcal{M}^R$	CTC	SDSC03	SDSC05	KTH	SDSC00
SDSC00-Ego	12.3351	7.9394	2.8640	7.9007	68.9609

Tabelle A.14: Austauschmatrizen für migrierte Arbeit in Prozent für fünf Sites

$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00	Durchschnitt
Alle $\text{ODIN}_{\text{AWFB}}$						
CTC	0.0000	32.9189	10.8725	30.3258	20.3775	23.6237
SDSC03	18.3815	0.0000	27.6050	22.2532	15.5540	20.9484
SDSC05	19.8593	21.0532	0.0000	13.7719	13.5621	17.0616
KTH	26.7372	12.0221	4.9651	0.0000	54.1236	24.4620
SDSC00	37.3206	16.6643	6.0662	36.5050	0.0000	24.1390
CTC-Ego						
CTC-Ego	0.0000	34.0009	11.5978	31.9643	18.1717	23.9337
SDSC03	0.0000	0.0000	37.4556	21.1425	15.5045	18.5257
SDSC05	0.0000	28.6369	0.0000	16.2595	13.6598	14.6390
KTH	0.0000	32.0336	8.0701	0.0000	55.0592	23.7907
SDSC00	0.0000	45.4034	9.7424	38.3024	0.0000	23.3620
SDSC03-Ego						
CTC	0.0000	0.0000	40.1890	17.0263	24.8186	20.5085
SDSC03-Ego	20.1703	0.0000	32.1950	7.3738	12.3286	18.0169
SDSC05	23.5396	0.0000	0.0000	9.9241	14.8110	12.0687
KTH	37.8422	0.0000	33.7421	0.0000	20.5871	23.0429
SDSC00	36.6792	0.0000	33.9825	18.0245	0.0000	22.1715
SDSC05-Ego						
CTC	0.0000	36.2047	0.0000	20.1957	28.5367	21.2343
SDSC03	25.5100	0.0000	0.0000	14.2723	20.7762	15.1396
SDSC05-Ego	21.4211	25.7530	0.0000	6.9620	10.1010	16.0593
KTH	31.7356	32.7447	0.0000	0.0000	20.0807	21.1403
SDSC00	36.0000	34.6857	0.0000	20.1622	0.0000	22.7120
KTH-Ego						
CTC	0.0000	42.1107	11.6615	0.0000	39.6133	23.3464
SDSC03	21.9787	0.0000	30.1927	0.0000	26.8025	19.7435
SDSC05	23.1565	23.1503	0.0000	0.0000	18.6743	16.2453

Tabelle A.15 – Fortsetzung						
$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00	Durchschnitt
KTH-Ego	27.0411	13.5205	5.3822	0.0000	51.4821	24.3565
SDSC00	64.4788	23.6938	6.1606	0.0000	0.0000	23.5833
SDSC00-Ego						
CTC	0.0000	45.5708	13.9093	32.8170	0.0000	23.0743
SDSC03	23.7747	0.0000	31.1890	22.0992	0.0000	19.2657
SDSC05	24.4793	22.6062	0.0000	14.7875	0.0000	15.4683
KTH	67.8692	22.2102	6.9024	0.0000	0.0000	24.2454
SDSC00-Ego	35.0066	17.2171	5.7023	36.1597	0.0000	23.5214

Tabelle A.15: Verhältnis von Anfragen und wirklich ausgetauschten Jobs für fünf Sites.

A.4.4 Experiment mit 3 kleinen Sites und verschiedenen Entscheidungsstrategien

Dieser Anhangabschnitt liefert alle Ergebnisdaten zum Versuch in Unterkapitel 7.4.2. An dieser Stelle sei nochmal darauf verwiesen, dass hier für jede Entscheidungsstategie nur die Ergebnisse dargestellt werden, die die besten Ergenisse in dem zu diesem Kapitel gehörenden Versuch erzielt haben. Für jede der getesteten Entscheidungsstrategien werden sowohl alle lokalen und globalen Metriken als auch die Austauschmatrizen für ausgetauschte Jobs und Arbeit zwischen den einzelnen Sites sowie auch die Verhältnisse zwischen den zum Austausch angefragten Jobs und wirklich ausgetauschten Jobs dargestellt.

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
ODIN_{OL}								
KTH	15.9804	68.7155	75157.6314	75157.6314	1.0000	1.0155	1.0155	-1.0000
CTC	65.6995	65.6995	52937.9601	52937.9601	1.0000	0.9698	0.9698	-1.0000
SDSC00	22.0094	73.9377	73605.8602	73605.8602	1.0000	1.0928	1.0928	-1.0000
Alle $\text{ODIN}_{\text{AWFB}}$								
KTH	15.9804	67.9233	61264.4200	60531.5123	0.6257	1.0036	1.0036	80.8272
Δ zu ODIN_{OL}	-	-1.1664	-22.6775	-24.1628	-	-	-	-
CTC	65.6995	68.6727	51877.3016	51806.7493	0.9118	1.0136	1.0136	57.9143
Δ zu ODIN_{OL}	-	4.3295	-2.0446	-2.1835	-	-	-	-
SDSC00	22.0404	64.4818	47139.2725	47383.1845	0.7022	0.9517	0.9517	71.2361
Δ zu ODIN_{OL}	-	-14.6644	-56.1455	-55.3417	-	-	-	-
ODIN_{KG}								
KTH	15.9804	67.0385	70341.5612	71315.3189	0.9752	0.9907	0.9907	100.0000
Δ zu ODIN_{OL}	-	-2.5016	-6.8467	-5.3878	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-1.3198	12.9044	15.1213	-	-	-	-

Tabelle A.16 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
SDSC00	22.0094	73.4381	71231.7115	71330.7003	0.9932	1.0854	1.0854	100.0000
Δ zu ODIN_{OL}	-	-0.6804	-3.3330	-3.1896	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	12.1956	33.8226	33.5725	-	-	-	-
CTC	65.6995	66.2378	53106.2580	52986.3492	1.0000	0.9777	0.9777	-1.0000
Δ zu ODIN_{OL}	-	0.8127	0.3169	0.0913	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-3.6760	2.3141	2.2262	-	-	-	-
ODIN_{ROQ}								
KTH	15.9804	65.9849	59695.1967	55697.7177	0.4423	0.9751	0.9751	73.0099
Δ zu ODIN_{OL}	-	-4.1383	-25.9023	-34.9384	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-2.9377	-2.6287	-8.6786	-	-	-	-
SDSC00	22.0146	64.2154	46745.2682	45233.5750	0.6424	0.9489	0.9489	73.8299
Δ zu ODIN_{OL}	-	-15.1403	-57.4616	-62.7240	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.4150	-0.8429	-4.7522	-	-	-	-
CTC	65.6995	69.1810	51428.8910	52614.1716	0.8579	1.0213	1.0213	40.1843
Δ zu ODIN_{OL}	-	5.0324	-2.9343	-0.6154	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.7346	-0.8719	1.5346	-	-	-	-
ODIN_{ST}								
KTH	15.9804	67.6572	61332.3839	60473.3327	0.6340	0.9996	0.9996	80.1193
Δ zu ODIN_{OL}	-	-1.5643	-22.5415	-24.2823	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.3933	0.1108	-0.0962	-	-	-	-
SDSC00	22.0404	65.1789	46620.4284	47642.9753	0.7027	0.9620	0.9620	71.6324
Δ zu ODIN_{OL}	-	-13.4381	-57.8833	-54.4947	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.2124	-0.4246	-0.7782	-	-	-	-
CTC	65.6995	68.5272	51657.9507	51406.7149	0.9097	1.0114	1.0114	59.5572
Δ zu ODIN_{OL}	-	4.1263	-2.4779	-2.9787	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	1.0695	-1.1129	0.5453	-	-	-	-
$\text{ODIN}_{\text{WTQE}}$								
KTH	15.9805	68.9556	75312.9529	75363.0672	0.9989	1.0190	1.0190	0.0984
Δ zu ODIN_{OL}	-	0.3481	0.2062	0.2726	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.3481	0.2062	0.2726	-	-	-	-
SDSC00	22.0094	74.1658	73781.7174	73670.2324	0.9994	1.0962	1.0962	0.1545
Δ zu ODIN_{OL}	-	0.3075	0.2383	0.0874	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.1889	-0.1012	-0.1304	-	-	-	-
CTC	65.6995	65.5757	52884.4426	52869.0114	0.9979	0.9679	0.9679	0.2642
Δ zu ODIN_{OL}	-	-0.1889	-0.1012	-0.1304	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.3075	0.2383	0.0874	-	-	-	-
$\text{ODIN}_{\text{WTQH}}$								
KTH	15.9804	78.7772	83967.2102	79957.0813	0.9185	1.1642	1.1642	7.2626
Δ zu ODIN_{OL}	-	12.7723	10.4917	6.0025	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	13.7780	27.0377	24.2950	-	-	-	-
SDSC00	22.0094	77.3398	67852.0972	67479.2360	0.8942	1.1431	1.1431	10.2456
Δ zu ODIN_{OL}	-	4.3988	-8.4799	-9.0793	-	-	-	-

Tabelle A.16 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	16.6253	30.5264	29.7811	-	-	-	-
CTC	65.6995	62.2935	51994.1848	51852.3330	0.9322	0.9196	0.9196	8.3798
Δ zu ODIN_{OL}	-	-5.4678	-1.8152	-2.0937	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-10.2407	0.2248	0.0879	-	-	-	-

Tabelle A.16: Lokale Metriken für das Experiment mit drei Sites

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
ODIN_{OL}	54624.7927	-1.0000	-1.0000	-1.0000	-1.0000	64.2740
$\text{ODIN}_{\text{AWFB}}$	46971.3130	78.3982	7772.8345	38.6305	316515.1344	64.3910
Δ zu ODIN_{OL}	-14.0110	-	-	-	-	0.1821
ODIN_{KG}	59618.8044	100.0000	7486.2647	93.7353	673947.7843	67.6574
Δ zu ODIN_{OL}	-13.1682	-	-	-	-	0.0547
Δ zu $\text{ODIN}_{\text{AWFB}}$	13.9537	-	-	-	-	-0.0372
ODIN_{ROQ}	51708.6688	55.9692	9116.5271	8.9324	78955.2969	67.6735
Δ zu ODIN_{OL}	-14.8969	-	-	-	-	0.0238
Δ zu $\text{ODIN}_{\text{AWFB}}$	-1.0788	-	-	-	-	-0.0134
ODIN_{ST}	52079.7617	68.0440	9250.4841	11.1518	101683.3838	67.6825
Δ zu ODIN_{OL}	-14.2862	-	-	-	-	0.0372
Δ zu $\text{ODIN}_{\text{AWFB}}$	-0.4562	-	-	-	-	0.0000
$\text{ODIN}_{\text{WTQE}}$	60787.5270	0.2047	8113.2000	10.9455	77055.6582	67.6574
Δ zu ODIN_{OL}	0.0452	-	-	-	-	-0.0000
Δ zu $\text{ODIN}_{\text{AWFB}}$	0.0452	-	-	-	-	-0.0000
$\text{ODIN}_{\text{WTQH}}$	60298.6433	8.5271	10760.2131	9.1051	93351.1783	67.6574
Δ zu ODIN_{OL}	-0.7594	-	-	-	-	0.0000
Δ zu $\text{ODIN}_{\text{AWFB}}$	15.2531	-	-	-	-	-0.0372

Tabelle A.17: Globale Metriken für das Experiment mit drei Sites

$\mathcal{M}$	CTC	KTH	SDSC00
ODIN_{KG}			
CTC	100.0000	0.0000	0.0000
KTH	0.2493	99.7507	0.0000
SDSC00	0.0839	0.0201	99.8960
ODIN_{ROQ}			
CTC	79.6604	10.8850	9.4553
KTH	50.6373	45.1561	4.2068
SDSC00	18.1852	16.7930	65.0218

Tabelle A.18 – Fortsetzung			
$\mathcal{M}$	CTC	KTH	SDSC00
ODIN_{ST}			
CTC	87.2459	7.9575	4.7969
KTH	21.5527	73.5805	4.8669
SDSC00	7.8128	10.2046	81.9826
$\text{ODIN}_{\text{WTQE}}$			
CTC	99.7396	0.1010	0.1593
KTH	0.0246	99.9017	0.0737
SDSC00	0.0436	0.1107	99.8457
$\text{ODIN}_{\text{WTQH}}$			
CTC	91.9546	3.5972	4.4484
KTH	1.6644	93.0019	5.3343
SDSC00	3.1902	5.9343	90.8755

Tabelle A.18: Austauschmatrizen für migrierte Jobs in Prozent für drei Sites

$\mathcal{M}^{\mathcal{S}}$	CTC	KTH	SDSC00
ODIN_{KG}			
CTC	100.0000	0.0000	0.0000
KTH	2.3925	97.6075	0.0000
SDSC00	0.5206	0.1121	99.3673
ODIN_{ROQ}			
CTC	86.1112	7.2083	6.6933
KTH	51.8778	44.7290	3.4061
SDSC00	19.1629	14.5529	66.2841
ODIN_{ST}			
CTC	90.9121	4.5062	4.5897
KTH	30.4516	63.4491	6.1226
SDSC00	17.8086	11.5923	70.5992
$\text{ODIN}_{\text{WTQE}}$			
CTC	99.7872	0.0721	0.1408
KTH	0.0401	99.8896	0.0705
SDSC00	0.0494	0.0365	99.9142
$\text{ODIN}_{\text{WTQH}}$			

Tabelle A.19 – Fortsetzung			
Migrated Work in Percent	CTC	KTH	SDSC00
CTC	92.9208	3.3629	3.7227
KTH	2.0354	92.0265	5.9653
SDSC00	2.9413	6.9911	90.0676

Tabelle A.19: Austauschmatrizen für migrierte Arbeit in Prozent für drei Sites

$\mathcal{M}$	CTC	KTH	SDSC00	Durchschnitt
ODIN_{KG}				
CTC	0.0000	0.0000	0.0000	0.0000
KTH	100.0000	0.0000	0.0000	50.0000
SDSC00	80.6452	85.7143	0.0000	83.1797
ODIN_{ROQ}				
CTC	0.0000	21.5268	18.6795	20.1031
KTH	67.4099	0.0000	5.6000	36.5049
SDSC00	38.3842	35.6299	0.0000	37.0070
ODIN_{ST}				
CTC	0.0000	37.2687	22.3990	29.8338
KTH	65.3605	0.0000	14.7588	40.0596
SDSC00	31.0616	41.0637	0.0000	36.0627
$\text{ODIN}_{\text{WTQE}}$				
CTC	0.0000	0.1026	0.1617	0.1321
KTH	0.0246	0.0000	0.0738	0.0492
SDSC00	0.0437	0.1112	0.0000	0.0774
$\text{ODIN}_{\text{WTQH}}$				
CTC	0.0000	3.7508	4.6331	4.1919
KTH	1.7273	0.0000	5.5353	3.6313
SDSC00	3.5822	6.6863	0.0000	5.1343

Tabelle A.20: Verhältnis von Anfragen und wirklich ausgetauschten Jobs für drei Sites.

A.4.5 Experiment mit 5 Sites und verschiedenen Entscheidungsstrategien

Dieser Anhangabschnitt liefert alle Ergebnisdaten zum Versuch in Unterkapitel 7.4.2. An dieser Stelle sei nochmal darauf verwiesen, dass hier für jede Entscheidungsstategie

nur die Ergebnisse dargestellt werden, die die besten Ergenisse in dem zu diesem Kapitel gehörenden Versuch erzielt haben. Für jede der getesteten Entscheidungsstrategien werden sowohl alle lokalen und globalen Metriken als auch die Austauschmatrizen für ausgetauschte Jobs und Arbeit zwischen den einzelnen Sites sowie auch die Verhältnisse zwischen den zum Austausch angefragten Jobs und wirklich ausgetauschten Jobs dargestellt.

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Alle ODIN_{OL}								
KTH	4.1295	68.7155	75157.6314	75157.6314	1.0000	1.0662	1.0662	-1.0000
SDSC00	5.6875	73.9377	73605.8602	73605.8602	1.0000	1.1475	1.1475	-1.0000
CTC	16.9776	65.6995	52937.9601	52937.9601	1.0000	1.0197	1.0197	-1.0000
SDSC03	47.5875	68.7376	50772.4765	50772.4765	1.0000	1.0694	1.0694	-1.0000
SDSC05	60.1680	60.1680	54953.8424	54953.8424	1.0000	0.9336	0.9336	-1.0000
Alle $\text{ODIN}_{\text{AWFB}}$								
KTH	4.1295	66.3107	53068.5629	52681.7561	0.4760	1.0287	1.0287	97.8480
Δ zu ODIN_{OL}	-	-3.6267	-41.6236	-42.6635	-	-	-	-
SDSC00	5.6923	66.1882	41638.1234	41359.0784	0.5098	1.0270	1.0270	96.0790
Δ zu ODIN_{OL}	-	-11.7083	-76.7752	-77.9678	-	-	-	-
CTC	16.9776	67.1014	43850.7403	43657.4458	0.6408	1.0409	1.0409	91.2447
Δ zu ODIN_{OL}	-	2.0892	-20.7231	-21.2576	-	-	-	-
SDSC03	47.6742	66.5080	43049.8309	44230.1898	0.7640	1.0329	1.0329	70.1631
Δ zu ODIN_{OL}	-	-3.3523	-17.9389	-14.7915	-	-	-	-
SDSC05	60.1758	62.1461	51049.8355	50024.2336	0.8571	0.9640	0.9640	59.3861
Δ zu ODIN_{OL}	-	3.1831	-7.6474	-9.8544	-	-	-	-
ODIN_{ROQ}								
KTH	4.1292	72.6931	63135.5513	64644.4477	0.4346	1.1280	1.1280	62.6898
Δ zu ODIN_{OL}	-	5.4718	-19.0417	-16.2631	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	8.7800	15.9450	18.5054	-	-	-	-
SDSC00	5.6937	73.8204	48342.8349	55251.6565	0.4325	1.1452	1.1452	68.7444
Δ zu ODIN_{OL}	-	-0.1589	-52.2581	-33.2193	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	10.3389	13.8691	25.1442	-	-	-	-
CTC	16.9776	69.3644	47157.5439	48319.3269	0.6463	1.0762	1.0762	54.2871
Δ zu ODIN_{OL}	-	5.2835	-12.2577	-9.5586	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.2625	7.0122	9.6481	-	-	-	-
SDSC03	47.6136	65.7670	42987.2207	43751.7040	0.7710	1.0227	1.0227	43.5593
Δ zu ODIN_{OL}	-	-4.5168	-18.1106	-16.0469	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-1.1268	-0.1456	-1.0936	-	-	-	-
SDSC05	60.1404	60.9891	49873.5720	47965.1568	0.8573	0.9469	0.9469	29.8801
Δ zu ODIN_{OL}	-	1.3463	-10.1863	-14.5703	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-1.8972	-2.3585	-4.2929	-	-	-	-
ODIN_{ST}								
KTH	4.1295	66.6535	53274.6248	50376.9905	0.4740	1.0340	1.0340	97.6755
Δ zu ODIN_{OL}	-	-3.0937	-41.0759	-49.1904	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.5143	0.3868	-4.5750	-	-	-	-

Tabelle A.21 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
SDSC00	5.6955	66.4988	41547.4735	42756.6511	0.5059	1.0318	1.0318	96.1269
Δ zu ODIN_{OL}	-	-11.1866	-77.1609	-72.1507	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.4671	-0.2182	3.2687	-	-	-	-
CTC	16.9776	66.5255	43191.4213	43188.0657	0.6207	1.0319	1.0319	91.0035
Δ zu ODIN_{OL}	-	1.2415	-22.5659	-22.5754	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-0.8657	-1.5265	-1.0868	-	-	-	-
SDSC03	47.6742	66.5780	42477.3275	43739.5198	0.7609	1.0340	1.0340	69.9336
Δ zu ODIN_{OL}	-	-3.2437	-19.5284	-16.0792	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.1050	-1.3478	-1.1218	-	-	-	-
SDSC05	60.1758	62.2020	50950.9157	49833.9617	0.8550	0.9648	0.9648	58.1572
Δ zu ODIN_{OL}	-	3.2699	-7.8564	-10.2739	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	0.0897	-0.1941	-0.3818	-	-	-	-
$\text{ODIN}_{\text{WTQE}}$								
KTH	4.1296	69.0956	74923.4230	74872.8843	0.9969	1.0721	1.0721	0.3830
Δ zu ODIN_{OL}	-	0.5501	-0.3126	-0.3803	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	4.0306	29.1696	29.6384	-	-	-	-
SDSC00	16.9776	65.5194	52748.9641	52780.4974	0.9948	1.0169	1.0169	0.4534
Δ zu ODIN_{OL}	-	-12.8486	-39.5399	-39.4565	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-1.0207	21.0636	21.6395	-	-	-	-
CTC	5.6875	74.1878	73699.2012	73690.8551	0.9954	1.1514	1.1514	1.2355
Δ zu ODIN_{OL}	-	11.4416	28.1702	28.1621	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	9.5520	40.5004	40.7560	-	-	-	-
SDSC03	47.5879	68.6971	50685.0694	50857.3789	0.9968	1.0688	1.0688	0.4016
Δ zu ODIN_{OL}	-	-0.0589	-0.1725	0.1669	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.1865	15.0641	13.0309	-	-	-	-
SDSC05	60.1680	60.2009	54901.6051	54743.1204	0.9981	0.9341	0.9341	0.2969
Δ zu ODIN_{OL}	-	0.0546	-0.0951	-0.3849	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-3.2313	7.0158	8.6201	-	-	-	-
$\text{ODIN}_{\text{WTQH}}$								
KTH	4.1290	81.6178	87690.9628	84738.7480	0.8948	1.2664	1.2664	10.4879
Δ zu ODIN_{OL}	-	15.8081	14.2926	11.3067	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	18.7546	39.4823	37.8304	-	-	-	-
SDSC00	16.9776	62.4457	51265.3380	51087.1130	0.8720	0.9690	0.9690	12.0862
Δ zu ODIN_{OL}	-	-18.4032	-43.5782	-44.0791	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-5.9931	18.7792	19.0421	-	-	-	-
CTC	5.6898	80.7395	69058.2851	72621.6117	0.8168	1.2526	1.2526	18.7755
Δ zu ODIN_{OL}	-	18.6278	23.3431	27.1044	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	16.8915	36.5018	39.8837	-	-	-	-
SDSC03	47.6115	69.1658	48434.1887	48567.3626	0.9270	1.0756	1.0756	9.8432
Δ zu ODIN_{OL}	-	0.6191	-4.8278	-4.5403	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	3.8426	11.1169	8.9302	-	-	-	-
SDSC05	60.1751	59.4181	52716.8819	51928.4542	0.9357	0.9216	0.9216	8.4205

Tabelle A.21 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Δ zu ODIN_{OL}	-	-1.2620	-4.2433	-5.8261	-	-	-	-
Δ zu $\text{ODIN}_{\text{AWFB}}$	-	-4.5912	3.1623	3.6670	-	-	-	-

Tabelle A.21: Lokale Metriken für das Experiment mit fünf Sites

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
ODIN_{OL}	54624.7927	-1.0000	-1.0000	-1.0000	-1.0000	64.2740
$\text{ODIN}_{\text{AWFB}}$	46971.3130	78.3982	7772.8345	38.6305	316515.1344	64.3910
Δ zu ODIN_{OL}	-14.0110	-	-	-	-	0.1821
ODIN_{ROQ}	47431.6850	49.3149	8057.8369	21.8875	165553.5374	64.3091
Δ zu ODIN_{OL}	-13.1682	-	-	-	-	0.0547
Δ zu $\text{ODIN}_{\text{AWFB}}$	0.9801	-	-	-	-	-0.1272
ODIN_{ST}	46643.6389	77.9030	7946.2171	38.9005	324604.4151	64.3910
Δ zu ODIN_{OL}	-13.9937	-	-	-	-	0.0084
Δ zu $\text{ODIN}_{\text{AWFB}}$	-0.6976	-	-	-	-	0.0000
$\text{ODIN}_{\text{WTQE}}$	54543.4474	0.4759	6438.4192	19.7523	146578.9466	64.2744
Δ zu ODIN_{OL}	-0.1489	-	-	-	-	0.0007
Δ zu $\text{ODIN}_{\text{AWFB}}$	16.1208	-	-	-	-	-0.1811
$\text{ODIN}_{\text{WTQH}}$	52778.1661	11.0840	7987.3376	24.4502	182771.9588	64.3064
Δ zu ODIN_{OL}	-3.3806	-	-	-	-	0.0504
Δ zu $\text{ODIN}_{\text{AWFB}}$	12.3626	-	-	-	-	-0.1315

Tabelle A.22: Globale Metriken für das Experiment mit fünf Sites

$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00
$\text{ODIN}_{\text{AWFB}}$					
CTC	84.3133	5.6595	1.8692	4.8719	3.2864
SDSC03	3.7890	84.5176	6.0914	3.2860	2.3163
SDSC05	4.2575	4.9905	85.9231	2.4231	2.4061
KTH	7.8093	3.5114	1.4502	71.4210	15.8087
SDSC00	8.8125	3.9349	1.4324	8.5072	77.3130
ODIN_{ROQ}					
CTC	63.7249	12.8384	5.9302	10.3099	7.1974
SDSC03	5.1217	73.6216	3.9234	13.1044	4.2299
KTH	14.0178	14.1091	41.8604	12.5645	17.4492
SDSC05	4.5659	6.5618	2.7809	82.4640	3.6278
SDSC00	19.3593	14.3677	12.9990	9.4364	43.8376

Tabelle A.23 – Fortsetzung

$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00
ODIN_{ST}					
CTC	84.7874	5.4833	1.8187	4.8084	3.1025
SDSC03	3.8973	84.5725	6.0823	3.0741	2.3742
SDSC05	4.2094	5.0051	86.1127	2.4525	2.2205
KTH	8.0480	3.5465	1.3870	71.0804	15.9386
SDSC00	9.0641	4.0356	1.4592	8.9198	76.5213
$\text{ODIN}_{\text{WTQE}}$					
CTC	99.5466	0.1464	0.1438	0.0609	0.1023
SDSC03	0.0366	99.5990	0.2684	0.0717	0.0244
SDSC05	0.0254	0.2123	99.7036	0.0240	0.0347
KTH	0.0070	0.1475	0.1896	99.6172	0.0386
SDSC00	0.0201	0.9259	0.2348	0.0537	98.7655
$\text{ODIN}_{\text{WTQH}}$					
CTC	88.1825	3.5130	2.1257	3.1659	3.0131
SDSC03	1.2137	90.7706	1.4898	4.4327	2.0936
KTH	1.4010	2.9390	89.6977	2.9005	3.0622
SDSC05	1.1121	3.3176	1.4232	91.8027	2.3447
SDSC00	3.1130	6.2328	3.7638	4.2134	82.6770

Tabelle A.23: Austauschmatrizen für migrierte Jobs in Prozent für fünf Sites

$\mathcal{M}^S$	CTC	SDSC03	SDSC05	KTH	SDSC00
$\text{ODIN}_{\text{AWFB}}$					
CTC	62.8579	20.8534	9.9103	3.2133	3.1887
SDSC03	3.9214	76.9449	17.9678	0.5620	0.6052
SDSC05	3.4610	8.2326	86.5215	0.8190	0.9666
KTH	18.3968	10.4577	4.6759	47.6596	18.8820
SDSC00	20.1573	13.1221	3.2099	10.2739	53.2369
ODIN_{ROQ}					
CTC	64.7526	15.2271	3.6477	11.6373	4.7594
SDSC03	5.8051	77.7315	1.2058	13.6414	1.6343
KTH	13.2275	14.7712	42.5324	10.8779	18.7037
SDSC05	3.5012	7.4179	0.8220	86.9305	1.3294

Tabelle A.24 – Fortsetzung

Migrated Work in Percent	CTC	SDSC03	SDSC05	KTH	SDSC00
SDSC00	18.1943	14.2729	12.7410	10.4714	44.3205
ODIN_{ST}					
CTC	62.0063	20.8181	10.5280	3.3746	3.2973
SDSC03	4.2012	76.7647	17.8190	0.5508	0.6657
SDSC05	3.5374	8.3723	86.3861	0.8027	0.9024
KTH	19.0846	9.9904	4.2699	47.3255	19.4036
SDSC00	19.8703	13.0164	3.5208	10.2180	53.3746
$\text{ODIN}_{\text{WTQE}}$					
CTC	99.3867	0.3696	0.1183	0.0449	0.0809
SDSC03	0.0331	99.6839	0.2210	0.0454	0.0168
SDSC05	0.0116	0.1157	99.8281	0.0091	0.0355
KTH	0.0007	0.1737	0.2624	99.5194	0.0440
SDSC00	0.0307	0.4402	0.2933	0.0231	99.2127
$\text{ODIN}_{\text{WTQH}}$					
CTC	86.1920	5.5373	2.1039	3.8618	2.3144
SDSC03	1.2731	92.6237	0.5281	4.9196	0.6623
KTH	1.6841	2.9118	89.2957	2.7549	3.3795
SDSC05	0.8263	3.7837	0.5503	93.8838	0.9567
SDSC00	3.2780	7.3585	3.7676	4.0615	81.5344

Tabelle A.24: Austauschmatrizen für migrierte Arbeit in Prozent für fünf Sites

$\mathcal{M}$	CTC	SDSC03	SDSC05	KTH	SDSC00	Durchschnitt
$\text{ODIN}_{\text{AWFB}}$						
CTC	0.0000	32.9189	10.8725	30.3258	20.3775	23.6237
SDSC03	18.3815	0.0000	27.6050	22.2532	15.5540	20.9484
SDSC05	19.8593	21.0532	0.0000	13.7719	13.5621	17.0616
KTH	26.7372	12.0221	4.9651	0.0000	54.1236	24.4620
SDSC00	37.3206	16.6643	6.0662	36.5050	0.0000	24.1390
ODIN_{ROQ}						
CTC	0.0000	19.2129	9.0157	15.4289	10.9308	13.6471
SDSC03	8.6305	0.0000	7.4671	21.6386	8.0204	11.4391
KTH	15.1149	15.2134	0.0000	13.5474	18.8141	15.6724

Tabelle A.25 – Fortsetzung						
RM	CTC	SDSC03	SDSC05	KTH	SDSC00	Durchschnitt
SDSC05	8.0543	11.1809	5.3476	0.0000	6.9458	7.8821
SDSC00	23.6963	17.5864	15.9702	11.5505	0.0000	17.2008
ODIN_{ST}						
CTC	0.0000	32.8012	10.8795	30.8562	19.8262	23.5908
SDSC03	18.8997	0.0000	27.5712	20.8501	15.9545	20.8189
SDSC05	19.4798	20.9605	0.0000	13.7716	12.3560	16.6420
KTH	27.1822	11.9782	4.6845	0.0000	53.8306	24.4189
SDSC00	37.1103	16.5225	5.9745	36.9819	0.0000	24.1473
$\text{ODIN}_{\text{WTQE}}$						
CTC	0.0000	0.1464	0.1438	0.0618	0.1038	0.1140
SDSC03	0.0373	0.0000	0.2688	0.0819	0.0278	0.1039
SDSC05	0.0263	0.2126	0.0000	0.0270	0.0388	0.0762
KTH	0.0070	0.1476	0.1897	0.0000	0.0386	0.0957
SDSC00	0.0201	0.9266	0.2350	0.0539	0.0000	0.3089
$\text{ODIN}_{\text{WTQH}}$						
CTC	0.0000	3.5929	2.2068	3.2378	3.1246	3.0405
SDSC03	1.3166	0.0000	1.8091	4.7273	2.5332	2.5966
KTH	1.4263	2.9920	0.0000	2.9526	3.1171	2.6220
SDSC05	1.1797	3.4080	1.6373	0.0000	2.6865	2.2279
SDSC00	3.3741	6.7554	4.0932	4.5666	0.0000	4.6973

Tabelle A.25: Verhältnis von Anfragen und wirklich ausgetauschten Jobs für fünf Sites.

A.4.6 Experiment mit 3 kleinen Sites und ODIN_{SS}

Dieser Anhangabschnitt liefert alle Ergebnisdaten zum Versuch in Unterkapitel 7.5. Dazu gehören sowohl alle lokalen und globalen Metriken als auch die Austauschmatrizen für ausgetauschte Jobs und Arbeit zwischen den einzelnen Sites sowie auch die Verhältnisse zwischen den zum Austausch angefragten Jobs und wirklich ausgetauschten Jobs.

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
Alle ODIN_{OL}								
KTH	15.9804	68.7155	75157.6314	75157.6314	1.0000	1.0155	1.0155	-1.0000
CTC	65.6995	65.6995	52937.9601	52937.9601	1.0000	0.9698	0.9698	-1.0000
SDSC00	22.0094	73.9377	73605.8602	73605.8602	1.0000	1.0928	1.0928	-1.0000
Alle ODIN_{SS}								

Tabelle A.26 – Fortsetzung

	U_τ	U_π	$AWRT_\tau$	$AWRT_\pi$	ΔSA	$\mathcal{B}$	$\mathcal{B}^T$	$\mathcal{M}$
KTH	15.9797	62.4287	66465.4432	66288.7085	0.8594	0.9224	0.9224	37.9734
Δ zu ODIN_{OL}	-	-10.0705	-13.0778	-13.3792	-	-	-	-
CTC	65.6995	68.1464	53827.5493	53645.9690	0.9890	1.0060	1.0060	6.7765
Δ zu ODIN_{OL}	-	3.5907	1.6527	1.3198	-	-	-	-
SDSC00	22.0310	70.6876	63868.7312	65923.8522	0.8950	1.0442	1.0442	18.2646
Δ zu ODIN_{OL}	-	-4.5978	-15.2455	-11.6529	-	-	-	-
KTH mit $\text{ODIN}_{\text{AWFE}}$								
KTH-Ego	15.9804	68.7147	75157.5412	75158.5101	1.0000	1.0155	1.0155	0.0177
Δ zu ODIN_{OL}	-	-0.0013	-0.0001	0.0012	-	-	-	-
CTC	65.6995	66.9126	53191.9461	53016.8945	0.9927	0.9877	0.9877	4.8436
Δ zu ODIN_{OL}	-	1.8130	0.4775	0.1489	-	-	-	-
SDSC	22.0094	69.8725	63797.6996	64976.0735	0.9233	1.0327	1.0327	11.5201
Δ zu ODIN_{OL}	-	-5.8181	-15.3738	-13.2815	-	-	-	-
CTC mit $\text{ODIN}_{\text{AWFE}}$								
KTH	15.9797	67.1233	72024.2476	70953.9607	0.9475	0.9918	0.9918	17.0579
Δ zu ODIN_{OL}	-	-2.3722	-4.3505	-5.9245	-	-	-	-
CTC-Ego	65.6995	65.6981	52937.9339	52939.0452	1.0000	0.9698	0.9698	0.0105
Δ zu ODIN_{OL}	-	-0.0022	-0.0000	0.0020	-	-	-	-
SDSC00	22.0310	75.2299	75298.4932	75986.0302	0.9788	1.1113	1.1113	4.4824
Δ zu ODIN_{OL}	-	1.7176	2.2479	3.1324	-	-	-	-
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$								
KTH	15.9804	63.0430	67514.7636	67192.1734	0.8979	0.9316	0.9316	25.1165
Δ zu ODIN_{OL}	-	-8.9980	-11.3203	-11.8547	-	-	-	-
CTC	65.6995	67.0223	53400.8380	53749.1641	0.9953	0.9893	0.9893	2.1680
Δ zu ODIN_{OL}	-	1.9736	0.8668	1.5092	-	-	-	-
SDSC00-Ego	22.0094	73.9345	73605.6345	73608.7173	1.0000	1.0928	1.0928	0.0162
Δ zu ODIN_{OL}	-	-0.0044	-0.0003	0.0039	-	-	-	-

Tabelle A.26: Lokale Metriken für das Experiment mit drei kleinen Sites

Konfiguration	$AWRT$	$\mathcal{M}^R$	$\bar{\mathcal{M}}^T$	$\bar{\mathcal{M}}^C$	$\bar{\mathcal{M}}^R$	U
Alle ODIN_{OL}	60760.0598	-1.0000	-1.0000	-1.0000	-1.0000	67.6574
Alle ODIN_{SS}	57912.1327	16.5639	7916.1079	11.0210	100836.8238	67.6796
Δ zu ODIN_{OL}	-4.6872	-	-	-	-	0.0329
KTH-Site mit $\text{ODIN}_{\text{AWFE}}$	58835.7694	5.2227	8918.8659	13.7191	127548.8356	67.6574
Δ zu alle ODIN_{OL}	-3.1670	-	-	-	-	0.0000
CTC-Site mit $\text{ODIN}_{\text{AWFE}}$	60636.4075	5.5125	7891.9642	7.7537	71224.2830	67.6796
Δ zu alle ODIN_{OL}	-0.2035	-	-	-	-	0.0329
SDSC00-Site mit $\text{ODIN}_{\text{AWFE}}$	59873.8104	6.4541	7535.2563	11.0356	92986.7301	67.6574
Δ zu alle ODIN_{OL}	-1.4586	-	-	-	-	0.0000

Tabelle A.27: Globale Metriken für das Experiment mit drei kleinen Sites mit ODIN_{SS}

$\mathcal{M}$	CTC	KTH	SDSC00
Alle ODIN_{SS}			
CTC	98.1930	0.4715	1.3356
KTH	5.4286	88.5916	5.9801
SDSC00	4.9547	1.6404	93.4049
CTC mit $\text{ODIN}_{\text{AWFE}}$			
CTC	99.9974	0.0013	0.0013
KTH	0.0000	93.8762	6.1240
SDSC00	0.0000	1.9188	98.0812
KTH mit $\text{ODIN}_{\text{AWFE}}$			
CTC	98.7863	0.0000	1.2138
KTH	0.0035	99.9930	0.0035
SDSC00	4.0523	0.0000	95.9477
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	99.4469	0.5531	0.0000
KTH	7.7566	92.2434	0.0000
SDSC00	0.0034	0.0034	99.9933

Tabelle A.28: Austauschmatrizen für migrierte Jobs in Prozent für drei kleine Sites mit ODIN_{SS}

$\mathcal{M}^R$	CTC	KTH	SDSC00
Alle ODIN_{SS}			
CTC	98.9231	0.3350	0.7429
KTH	8.5756	86.1642	5.2804
SDSC00	7.1565	2.0063	90.8372
CTC mit $\text{ODIN}_{\text{AWFE}}$			
CTC	99.9973	0.0016	0.0011
KTH	0.0000	94.9390	5.0803
SDSC00	0.0000	1.9306	98.0694
KTH mit $\text{ODIN}_{\text{AWFE}}$			
CTC	99.2442	0.0000	0.7567
KTH	0.0392	99.9235	0.0375
SDSC00	7.2457	0.0000	92.7543
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$			
CTC	99.6161	0.3842	0.0000
KTH	9.8160	90.1840	0.0000
SDSC00	0.0009	0.0013	99.9978

Tabelle A.29: Austauschmatrizen für migrierte Arbeit in Prozent für drei kleine Sites mit ODIN_{SS}

$\mathcal{M}$	CTC	KTH	SDSC00	Durchschnitt
Alle ODIN_{SS}				
CTC	0.0000	1.7731	5.0083	3.3907
KTH	18.0692	0.0000	19.9042	18.9867
SDSC00	13.7217	4.5812	0.0000	9.1514
CTC mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	0.0053	0.0052	0.0053
KTH	0.0000	0.0000	17.0579	8.5290
SDSC00	0.0000	4.5146	0.0000	2.2573
KTH mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	0.0000	4.8436	2.4218
KTH	0.0089	0.0000	0.0089	0.0089
SDSC00	11.5201	0.0000	0.0000	5.7601
SDSC00 mit $\text{ODIN}_{\text{AWFE}}$				
CTC	0.0000	2.1737	0.0000	1.0868

Tabelle A.30 – Fortsetzung

$\mathcal{M}$	CTC	KTH	SDSC00	Durchschnitt
KTH	25.1165	0.0000	0.0000	12.5583
SDSC00	0.0081	0.0081	0.0000	0.0081

Tabelle A.30: Verhältnis von Anfragen und wirklich ausgetauschten Jobs für 3 kleine-Site-Experiment mit ODIN_{SS}

A.5 Abkürzungsverzeichnis

AWRT Average Weighted Response Time

CPU Central Processing Unit

CSV Character Separated Values

CTC Cornell Theory Center

DARPA Defense Advanced Research Projects Agency

EASY Extensible Argonne Scheduling System

FCFS First Come First Serve

GT4 Globus Toolkit 4

GUI Graphical User Interface

IP Internet Protocol

KTH Kungliga Tekniska högskolan

NJS Network Job Supervisor

MPS Multi-Processor-Site

ODIN Open Decision Interface Node

OGSI Open Grid Services Infrastructure

P2P Peer-to-Peer

PG Projektgruppe

SA Squashed Area

SDSC San Diego Supercomputer Center

SOA Service Oriented Architecture

SWF Standard Workload Format

UNICORE Uniform Interface to Computing Resources

XML eXtensible Markup Language

Literaturverzeichnis

[Aberer u. Hauswirth 2002] Aberer, Karl ; Hauswirth, Manfred: An Overview of Peer-to-Peer Information Systems. In: Litwin, Witold (Hrsg.) ; Lévy, Gérard (Hrsg.): *Distributed Data and Structures 4, Records of the 4th International Meeting (WDAS 2002), Paris, France, March 20-23, 2002* Bd. 14, Carleton Scientific, 2002 (Proceedings in Informatics), S. 171–188

[Breslau et al. 2000] Breslau, L. ; Estrin, D. ; Fall, K. ; Floyd, S. ; Heidemann, J. ; Helmy, A. ; Huang, P. ; Mccanne, S. ; Varadhan, K. ; Xu, Ya ; Yu, Haobo: Advances in network simulation. In: *Computer* 33 (2000), Nr. 5, S. 59–67

[Buyya u. Murshed 2002] Buyya, Rajkumar ; Murshed, Manzur: GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. In: *j-CCPE* 14 (2002), November/Dezember, Nr. 13–15, S. 1175–1220

[Casanova 2001] Casanova, Henri: Simgrid: A Toolkit for the Simulation of Application Scheduling. In: *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*. Washington, DC, USA : IEEE Computer Society, 2001, S. 430

[Chapin et al. 1999a] Chapin, Steve J. ; Cirne, Walfredo ; Feitelson, Dror G. ; Jones, James P. ; Leutenegger, Scott T. ; Schwiegelshohn, Uwe ; Smith, Warren ; Talby, David: Benchmarks and Standards for the Evaluation of Parallel Job Schedulers. In: *IPPS/SPDP '99/JSSPP '99: Proceedings of the Job Scheduling Strategies for Parallel Processing*. London, UK : Springer-Verlag, 1999, S. 67–90

[Chapin et al. 1999b] Chapin, Steve J. ; Katramatos, Dimitrios ; Karpovich, John ; Grimshaw, Andrew: Resource Management in Legion / Department of Computer Science, University of Virginia. 1999. – Forschungsbericht

[Chetty u. Buyya 2002] Chetty, Madhu ; Buyya, Rajkumar: Weaving Computational Grids: How Analogous Are They with Electrical Grids? In: *Computing in Science and Engineering* 4 (2002), Nr. 4, S. 61–71

[Cohen 2003] Cohen, Bram: Incentives Build Robustness in BitTorrent. In: *Proceedings of Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA : June 5-6 2003*, 2003

[Dornberger u. Fuchs 2004] Dornberger, R. ; Fuchs, D.: *Peer-to-Peer Netzwerke und Geschäftsmodelle*. Fachhochschule Solothurn Nordwestschweiz, Olten, 2004

[Ernemann et al. 2002a] Ernemann, Carsten ; Hamscher, Volker ; Schwiegelshohn, Uwe ; Yahyapour, Ramin ; Streit, Achim: Enhanced Algorithms for Multi-site Scheduling. In: *GRID '02: Proceedings of the Third International Workshop on Grid Computing*. London, UK : Springer-Verlag, 2002, S. 219–231

[Ernemann et al. 2002b] Ernemann, Carsten ; Hamscher, Volker ; Schwiegelshohn, Uwe ; Yahyapour, Ramin ; Streit, Achim: On Advantages of Grid Computing for Parallel Job Scheduling. In: *CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA : IEEE Computer Society, 2002, S. 39

[Erwin u. Snelling 2001] Erwin, Dietmar W. ; Snelling, David F.: UNICORE: A Grid Computing Environment. In: *Euro-Par 2001: Parallel Processing: 7th International Euro-Par Conference Manchester* Bd. 2150, Springer Berlin/Heidelberg, 2001 (Lecture Notes in Computer Science), 825

[Fall u. Floyd 1996] Fall, Kevin ; Floyd, Sally: Simulation-based comparisons of Tahoe, Reno and SACK TCP. In: *SIGCOMM Comput. Commun. Rev.* 26 (1996), Nr. 3, S. 5–21

[Foster 2003] Foster, Ian: On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In: *2nd International Workshop on Peer-to-Peer Systems* (2003), S. 118–128

[Foster 2005] Foster, Ian: A Globus Primer; Or, Everything You Wanted to Know about Globus, but Were Afraid To Ask; Describing Globus Toolkit Version 4 / Globus. 2005. – Forschungsbericht

[Foster u. Kesselman 1999] Foster, Ian (Hrsg.) ; Kesselman, Carl (Hrsg.): *The grid: blueprint for a new computing infrastructure*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999

[Frey et al. 2001] Frey, James ; Tannenbaum, Todd ; Livny, Miron ; Foster, Ian ; Tuecke, Steven: Condor-G: A Computation Management Agent for Multi-Institutional Grids / Department of Computer Science, University of Wisconsin. 2001. – Forschungsbericht

[Garritano 2003] Garritano, Tom: Globus: An Infrastructure for Resource Sharing. In: *Cluster World* (2003)

[Graham 1969] Graham, Ronald L.: Bounds on multiprocessing timing anomalies. In: *SIAM Journal of Applied Mathematics* 17 (1969), Nr. 2, S. 416–429

[Grimme et al. 2007] Grimme, Christian ; Lepping, Joachim ; Papaspyrou, Alexander: Prospects of Collaboration between Compute Providers by means of Job Interchange. In: Frachtenberg, Eitan (Hrsg.) ; Schwiegelshohn, Uwe (Hrsg.): *Proceedings of the 13th Job Scheduling Strategies for Parallel Processing*, Springer, June 2007 (Lecture Notes in Computer Science (LNCS)). – to appear

[Grimshaw u. Wulf 1996] Grimshaw, Andrew S. ; Wulf, Wm. A.: Legion - A View From 50,000 Feet / Department of Computer Science, University of Virginia. 1996. – Forschungsbericht

[Hamscher et al. 2000] Hamscher, Volker ; Schwiegelshohn, Uwe ; Streit, Achim ; Yahyapour, Ramin: Evaluation of Job-Scheduling Strategies for Grid Computing. In: *GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. London, UK : Springer-Verlag, 2000, S. 191–202

[Iamnitchi et al. 2002] Iamnitchi, A. ; Foster, I. ; Nurmi, D.: A Peer-to-Peer Approach to Resource Location in Grid Environments. In: *Symposium on High Performance Distributed Computing*, 2002

[Liang et al. 2004] Liang, J. ; Kumar, R. ; Ross, K.: *Understanding KaZaA*. 2004

[Lifka 1995] Lifka, David A.: The ANL/IBM SP Scheduling System. In: *IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. London, UK : Springer-Verlag, 1995, S. 295–303

[Natrajan et al. 2001] Natrajan, Anand ; Nguyen-Tuong, Anh ; Humphrey, Marty A. ; Grimshaw, Andrew S.: The Legion Grid Portal / Department of Computer Science, University of Virginia. 2001. – Forschungsbericht

[Reinefeld et al. 2004] Reinefeld, Alexander ; Schintke, Florian ; Schütt, Thorsten: Scalable and Self-Optimizing Data Grids. In: *Annual Review of Scalable Computing* 6 (2004)

[Romberg 2002] Romberg, Mathilde: The UNICORE Grid infrastructure. In: *Sci. Program.* 10 (2002), Nr. 2, S. 149–157

[Schwiegelshohn 2004] Schwiegelshohn, Uwe: Preemptive Weighted Completion Time Scheduling of Parallel Jobs. In: *SIAM J. Comput.* 33 (2004), Nr. 6, S. 1280–1308

[Schwiegelshohn u. Yahyapour 2000] Schwiegelshohn, Uwe ; Yahyapour, Ramin: Fairness in parallel job scheduling. In: *Journal of Scheduling* 3 (2000), Nr. 5, S. 297–320

[Streit et al. 2005] Streit, A. ; Wäldrich, Oliver ; Wiedera, Ph. ; Ziegler, W.: On Scheduling in UNICORE - Extending the Web Services Agreement based Resource Management Framework. 2005. – Forschungsbericht

[Subramani et al. 2002] Subramani, Vijay ; Kettimuthu, Rajkumar ; Srinivasan, Srividya ; Sadayappan, P.: Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests. In: *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA : IEEE Computer Society, 2002, S. 359

[Takefusa et al. 1999] Takefusa, Atsuko ; Matsuoka, Satoshi ; Aida, Kento ; Nakada, Hidemoto ; Nagashima, Umpei: Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms. In: *hpdc* 00 (1999), S. 11

[Thain et al. 2004] Thain, Douglas ; Tannenbaum, Todd ; ; Livny, Miron: Distributed Computing in Practice: The Condor Experience / Computer Sciences Department, University of Wisconsin-Madison. 2004. – Forschungsbericht

[Trunfio et al. 2006] Trunfio, Paolo ; Talia, Domenico ; Fragopoulou, Paraskevi ; Mordacchini, Charis Papadakis M. ; Pennanen, Mika ; Popov, Konstantin ; Vlassov, Vladimir ; Haridi, Seif: Peer-to-Peer Models for Resource Discovery on Grids. Institute on System Architecture : CoreGRID Technical Report, 2006 (TR-0028)

[Varga 2001] Varga, Andras: The OMNeT++ discrete event simulation system. In: *Proceedings of the European Simulation Multiconference (ESM'2001)*, 2001

[Weil 1998] Weil, A.: Utilization and Predictability in Scheduling the IBM SP2 with Backfilling. In: *IPPS '98: Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium*. Washington, DC, USA : IEEE Computer Society, 1998, S. 542