Federated Capacity Planning
for Distributed Computing Infrastructures

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Alexander Papaspyrou

Dortmund

2013

Tag der mündlichen Prüfung: **21.06.2013**

Dekanin: Prof. Dr. Gabriele Kern-Isberner

Gutachter: Prof. Dr.-Ing. Uwe Schwiegelshohn (Technische Universität Dortmund)
Prof. Dr.-Ing. Ramin Yahyapour (Georg-August-Universität Göttingen)

To Nami, who escorted me
and Spyro, who passed amidst.

Preface

"The time has come," the Walrus said,
"to talk of many things."
—Lewis Carroll

Computing power has become the fifth utility for almost every application area in science and industry, and its ubiquitous availability is a key factor for advances in biotechnology, climate research, and product design. To cater this need, various concepts for the comprehensive, distributed provisioning of processing capacity have been developed over the last decade, and the most well-known representatives arguably are academic Grid computing and commercial Cloud computing. For that matter, both approaches share a certain tension between consumers and providers: While the former—researchers, engineers—express their desire for on-the-spot, reliable availability of computing power, the latter—academic computing centres, commercial data centres—care about economic and manageable provisioning.

In this work, we explore this tension with respect to capacity planning. To this end, we develop an appropriate model that reflects the needs of modern e-Infrastructures and embraces the results of three decades of distributed computing research. Using that model, we analyse whether—and if so, in how far—the building of federated infrastructures is reasonable from the stakeholders' (consumers and providers) perspective without disregarding their individual interests. In that context, we develop scheduling strategies for different architectures and approximate their limits. Finally, we evaluate the prequisites that need to be fulfilled in the technical architecture in order to transfer the afore discussed algorithms to real-world scenarios. To this end, we elicit the requirements from two production environments and develop a generalised interaction model that allows the application of the analysed strategies. There, we show that, using two newly designed protocols, not only the methods can be successfully transferred, but also—due to the extensibility of both the protocols and the architecture—envisage a manifold of other application scenarios.

Acknowledgement. Six years of intense research on a dedicated topic are—besides being a necessary precondition for a university degree—first and foremost a lesson in humility, as the trip to the edge of human knowledge turns out to be physically and mentally demanding, even beyond the personal limits. This is mainly due to the fact that the path that leads to the own, novel contribution, goes beyond this edge's boundaries into uncharted terrain and has many junctions, detours, and blind alleys. In order to master the lack of direction when exploring foreign land, the arguably most important thing is an environment that gives support, provides guidance and grants freedom.

I would therefore like to thank my advisors, Prof. Dr.-Ing. Uwe Schwiegelshohn and Prof. Dr.-Ing. Ramin Yahyapour, for giving me the opportunity to journey. Uwe Schwiegelshohn allowed me to freely pursue my ideas and, at the same time, provided the context for my research through real-world projects that required to build down-to-earth solutions rather than staying in the ivory tower. Ramin Yahyapour piqued my curiosity for distributed computing as an undergraduate already, opened the doors for priceless opportunities to collaborate with the most excellent researchers around the world, and introduced me to the majestic beauty of standardisation.

Traveling alone in modern research is all but possible. As such, I would like to thank my colleagues at the Robotics Research Institute for being so enjoyable throughout the whole time. Special thanks go to my close fellow Christian Grimme and my peers Joachim Lepping and Alexander Fölling—the honest, positive, and inspiring collaboration with such brilliant minds was most certainly one of the cornerstones of successful research. Further, I would like to thank my students Tim Dauer, Christian Fisseler, Christian Friem, Cesare Foltin, and Alexander Gumprich for helping with the implementation and verification of the presented algorithms and protocols. Andy Edmonds, Andre Merzky, Thijs Metsch, and Jochen Tösmann deserve my thanks for embarrassing me with the countless ambiguities and vaguenesses they revealed in the written text during their review.

Finally, I would like to thank my family for their caring support and endless patience. My deepest gratitude is due to my beloved wife Natalie Nami, who endured my presence, tolerated my absence, and continuously backed my work with her positive spirit and radiant mind.

Contents

List of Figures

List of Tables

1

Introduction

"Begin at the beginning," the King said gravely,
"and go on till you come to the end."
—Alice in Wonderland

The availability of information technology on the large scale has radically changed the way research and development is pursued. In fact, the ubiquitous availability of computing power for science and industry is nowadays considered the fifth utility: More and more application areas rely on complex computations, and e-Infrastructure[1] has become a key factor for advances in biotechnology, climate prediction, product design, and manufacturing. This naturally necessitates the operation of large-scale resource centres, and in many universities, commercial research centres, and even medium-to-large enterprises they can be considered a commodity service for users. But although hardware performance has significantly increased, the demand for more computing power is ever growing.

Additional processing capacity in such systems is typically acquired with respect to the demand of the local user communities; that is, if the resources are found to be constantly over-utilised, an investment into extending the system is made. Naturally, this demand is subject to constant change: the usage of a High Performance Computing system in research is typically bound to fixed publication dates, and industrial data centres depend on the amount of orders or certain—internal and external—projects. Hence, the load of such systems fluctuates over time, which leads to two undesirable situations: Either the available capacity is under-utilised, which harms the expected return on investment of the centre or, in case of over-utilisation, the users are forced into unacceptable delays due to a large backlog of work. With the cumulative amount of operational expenditure that is independent from the utilisation on the one hand, and the increasing expectations from important stakeholders regarding on-the-spot availability of computing power on the other, the provisioning of e-Infrastructure is always accompanied with the wish of satisfying conflicting goals.

[1] See Hüsing (2010) for an attempt to define the concept; here, we limit ourselves to the computational aspect of IT infrastructure for solving problems from different research domains.

1.1 Motivation

From an operators' point of view, the natural way to cope with this tension would be the dynamic reconfiguration of their system in an on-demand fashion: For example, if user-generated workload grows due to a conference deadline or a project deliverable, the operator would add additional resources to his local system until the backlog shrinks again. Likewise, he could offer idle parts of the system to other parts of his organisation, such as different departments in an enterprise or cooperating institutes within a network of universities. While such an approach ensures that the system is well-utilised—a fundamental performance benchmark for most operators and their institutions—over time, it also delivers a higher level of service quality to the users due to the adaptiveness to their workload.

The technical foundation for Distributed Computing Infrastructures (DCIs) that are capable of providing such service has been laid during the late 1990s with the emergence of Grid computing. In this area, a plethora of research has been conducted with respect to sensible workload distribution. Due to the architectural nature of Grid computing, much effort has been put into mechanisms for the delegation of workload between participating compute centres. In the industrial domain, a similar concept for moving away from on-premises infrastructure arose at the beginning of the new millennium: Pioneered by Amazon, a large, multinational electronic commerce company, the focus shifted towards the business model of selling computing capacity in an "as-a-service" fashion, and idea of Cloud computing as a different flavour of Distributed Computing Infrastructure was born.

Over the last five years, both technical foundations have been largely simplified and commoditised: With the advent of large production Grid infrastructures and the widespread offering of Cloud services, institutions' CIOs can now provision additional resources, e.g. compute, storage, and even networking, on demand without having to make a permanent investment into extending the local facilities. This is especially interesting in as far as the adaptation of capacity to the currently induced load can be handled without capital expenditure: An elastic resource space could grow and shrink on demand by reshaping available infrastructure.

With the availability of the technological foundation for doing so[2], the next logical step is to consider how these advances can be leveraged. Two natural approaches to the demand based idea discussed above can be derived very easily: When local resources are over-utilised, either load can be moved away[3], or external resources can be added to the local ones. This idea of course presumes a general agreement between different resource centres to (at the very minimum) collaborate in some way. If such a collaboration can be achieved between a number of participants, a *federation* is born – resource

[2] That is, through the commoditisation of virtualisation technology.
[3] To under-utilised resources at a different resource centre, that is.

centres agree to, within the limits of their own objectives[4], distribute workload among the participants and temporarily provide their own resources to others currently in need of more capacity.

Naturally, such federations open a multitude of interesting scientific and technical challenges regarding Scheduling and Resource Management (SRM). In order to make this general research problem of *Federated Capacity Planning for Distributed Computing Infrastructures* tangible, we narrow it down to answering the following research questions:

Question 1. What does a suitable federation model that fits to the architecture of current production infrastructures look like, and how can the technological gaps between the model and real environments be addressed?

Question 2. How large is the infrastructural effort to build a federation of resource centres, and how resilient are such systems towards change in the participant structure, amount, and behaviour?

Question 3. Since participants of the federation are organisationally (and thus financially) independent from each other, can we retain their autonomy in the federation and still achieve a good overall performance, thus satisfy each participant's requirements?

Question 4. How much insight into the operational model of each participant is necessary for being able to design and implement a well-performing federation, given that the disclosure of information such as system load, work backlog, etc. is not necessarily desirable?

Question 5. Is it possible to determine lower and upper bounds for the gain of a resource centre participating in a federation, and—if so—how close do current SRM strategies approximate them?

1.2 Goals

In this work, we aim to find solutions to the specific problem of SRM in DCIs. Our main goal is to provide a solid background for the decision whether it is beneficial to participate in a loosely coupled federation of resource providers for large-scale problem solving. Although necessarily making several simplifications on the model, we strive for lifelike assumptions on the benefits of such, and at the same time search for practical solutions to technical problems. To this end, we pursue three objectives.

[4] Often, these limits are expressed by means of cost, which enables the participants to mutually compensate the effort. Buyya et al (2005) has adopted this approach for both Grid and Cloud computing; for this work, however, the notion of monetary cost is considered to be out of scope.

First, we want to compile a realistic, yet manageable model of federated DCIs for the evaluation of such environments, and build the tools and metrics to make an assessment. This is necessary as it provides the foundation for the further analysis of the possible benefits in such a system.

Second, we want to explore whether it is at all reasonable for a provider to participate in such a federation. As each resource centre first and foremost cares about its very own users, the gain resulting from a federated model must be sufficiently large to justify the additional effort. As the gain is likely depending on the characteristics of the collaboration, we assess different kinds of algorithms for capacity planning with varying surrounding conditions. Also, we approach the boundaries of the feasible to gain insight of how large the gain can theoretically be.

Third, we want to identify and overcome the technological gaps in real world implementations of production DCIs that hinder the building of the afore discussed federations and propose a number of architectural blueprints that address the discovered challenges. To this end, we discuss example environments regarding their architecture and technicalities, formulate the two major challenges, namely negotiation and agreement, and provisioning, that need to be addressed for being able to bridge the gaps, and develop a generalised interaction model, which is implemented using two independent protocols.

Summarising, we want to provide a solid architectural, algorithmic, and technical basis for constructing federations of Distributed Computing Infrastructures on the basis of currently available technology with a potential for real world adoption.

1.3 Structure

The remainder of this work is subdivided into three parts.

Part I focuses on the modelling and evaluation of DCI environments. In Chapter 2, we classify our work in the context of the state of the art and discuss the overall model used for the further analysis. Next, in Chapter 3, we introduce the methodology of evaluation used to assess the performance gain reachable through SRM and present reference results for the non-federated case as a basis of comparison with further algorithmic studies.

Part II introduces algorithms for capacity planning problems. In Chapter 4, we make general assumptions on architecture and policy of federated DCI environments. Then, we analyse centralised architectures and focus on aspects of active and passive interaction as well as behaviour in hierarchical setups in Chapter 5. Next, in Chapter 6, we evaluate distributed architectures and review our algorithms with respect to migration of workload, robustness under changing conditions, and leasing of capacity. Finally, in Chapter 7, we compare the different approaches and explore the limits of federation for SRM algorithms in the given model.

Part III introduces blueprints for the implementation of SRM in federated DCIs. In the light of the model and the gained results, we then discuss the lessons learned from real-world implementations for climate research and plasma physics in Chapter 8, identify the obstacles towards the implementation of the reviewed algorithms in production DCIs, and formalise an architecture and technology stack on the basis of electronic contracts that allows to build loosely coupled federations in a standardised way. In Chapter 9, we address the problem of authoritative agreements between participants in the federation and develop a standardised means of negotiating electronic contracts between them. Finally, in Chapter 10 we propose an interface for the dynamic provisioning of arbitrary resources that exposes standardised mechanisms to manage the resources while at the same time having the potential of differentiation.

Ultimately, we conclude our findings in Chapter 11, giving a short synopsis of our work, subsuming our results, and providing an outlook on further research topics with respect to open questions.

Part I

Modelling and Evaluation of DCI Environments

In the following chapters, we address our first objective and build the foundation for our analysis of SRM in DCI environments. As a starting point, we discuss the state of the art in the area of distributed computing and introduce the two main architectural paradigms currently used for the structuring of DCI environments. Based on the taxonomies in literature, we narrow our problem space and set the stage for the model of federated capacity planning in DCIs which we use for our research.

Naturally, modelling without evaluation does not give any insight into the performance of SRM and the potential benefits that can be achieved through the federated approach. We therefore review potential methods of evaluation regarding their applicability to the problem at hand. Based on the selected approach of simulation, we discuss the input data used for further analysis and its properties. For the measurement of gain and loss of federated SRM, we introduce a number of assessment metrics and review their meaning with respect to the tension of system responsiveness and efficiency. We then use them to generate reference results with the selected input data for the non-federated case, that is without interaction between the resource centres participating in the federation.

2

Classification of SRM in Modern DCIs

If you ever need anything please don't hesitate
to ask someone else first.
—Kurt Cobain

Any attempt to approaching a topic as comprehensive as SRM for DCIs in its entire scope is obviously a futile act. As such, the very first step for research considering to broaden the knowledge in this area is to classify the environment in order to narrow the questions to a manageable extent.

In the light of the worded goals, we first review the current state of the art in Section 2.1. There, we revisit the history of distributed computing and discuss today's main flavours of the fundamental paradigm, namely Grid computing and Cloud computing, that have emerged over the decades in research and business. Against that background, we scope our efforts regarding SRM for DCIs by classifying them within a widely accepted taxonomy of properties, see Section 2.2. In addition, we discuss the infrastructure types under review with respect to their classes of deployment, see Section 2.3.

On this ground, we formulate the model for federated DCIs that will serve as the basis for all following findings. To this end, we introduce the formal notation for describing the environment, separate two distinct layers we assume for all participants in the federation, and discuss several detail aspects that influence SRM in such infrastructures.

2.1 State of the Art

Resource management as a challenge of distributed computing engages research for almost four decades now: Sullivan and Bashkow formulate the need for large scale distributed computing already in 1977, as

> "[...] there are certain classes of problems which are entirely beyond the capabilities of the largest present day computers [including] certain gaming problems, which arise in the evaluation of weapon systems, [...] the modelling of complex [...] physical phenomena, for example, [...] global weather prediction, [...] and other artificial intelligence problems."

Since then, a number of architectural paradigms for the general notion of distributed computing have been developed. The prevalent one in the academic context stems from the concept of "metacomputing" as formulated by Smarr and Catlett (1992), who then proposed a "[...] network of heterogeneous, computational resources linked by software in such a way that they can be used as easily as a personal computer." This was eventually refined by Foster and Kesselman (1998) towards "Grid Computing" as "[...] a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." Especially in the academic context, this is still used as the main paradigm for distributed computing.

The other important strain emerged during the first decade of the new millennium due to a schism in perception of e-Infrastructure: While the scientific community still focuses on Grids (albeit now from a production point of view), industry-oriented providers of Distributed Computing Infrastructure coined the term "Cloud Computing" for a new, demand-oriented and highly dynamic business model of resource provisioning. There have been many attempts toward a definition[1], but the most comprehensive one has been made by the National Institute of Standards and Technology (Mell and Grance, 2011), which defines cloud computing as "[...] a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

On both topics, many surveys have been published which formulate requirements, develop taxonomies, and discuss realisations of the two. For Grid computing, Kertész and Kacsuk (2007) provide a comprehensive overview of then current implementations and introduce several generic aspects of SRM in such environments; Venugopal et al (2006) approach the same topic from the perspective of data management. For Cloud computing, Zhang et al (2010) survey the current market of products, describe architectural and business models, and identify automated service provisioning and energy management as the major research challenges in the context of SRM; Rimal et al (2009) attempt a taxonomy by categorising implementations along the "Everything-as-a-Service" pattern and highlight load balancing and interoperability as major SRM problems.

Overall, a comparison of the two reveals that, especially in the context of SRM, both concepts are—besides a strong notion on virtualisation in the latter—facing mostly the same problems, see Foster et al (2008). As such, we will classify this work along a more generalised model, namely distributed computing and discuss two fundamental aspects here: properties of the SRM decision strategies and structure of the participating resources.

[1] See Vaquero et al (2008) for a survey.

2.2 Taxonomy of Properties

Casavant and Kuhl give a comprehensive survey on approaches to SRM in such environments for, as they state,

> "[...] ideas and work in the area of resource management generated in the last 10 to 15 years."

as early as 1988.

In the course of their work, they propose a general model for SRM systems as being mediators between consumers and resources on the basis of a policy framework and state that, for the evaluation of an SRM system, two properties must be considered: performance (consumer-centric) and efficiency (provider-centric[2]). Besides this, they introduce a comprehensive classification scheme for SRM characteristics which is—including most of the examples—surprisingly up-to-date and can be found in many subsequent publications in this area of research.

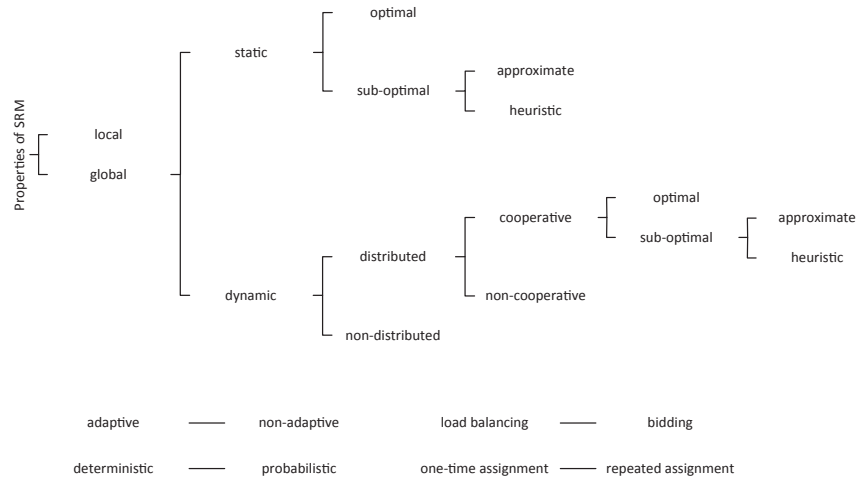


Fig. 2.1 *Taxonomy of SRM properties as proposed by Casavant and Kuhl (1988). The free-floating ones on the bottom are considered ancillary and mixable with the fundamental ones above.*

More specifically, they develop a classification scheme for SRM systems comprising of a hierarchical system of characteristics, see Figure 2.1, that is accompanied by a small set of ancillary properties. In this scheme, they distinguish several aspects that we will apply to the problem at hand:

[2] Unfortunately, they err with their notion of consumer centricity: The efficiency viewpoint is geared towards *usage* of the SRM system instead of considering the question of *utilisation*; here, we rectify this point.

LOCAL VS. GLOBAL
: At the highest level, one needs to differentiate between the assignment of indivisible workload to a single resource and the finding of a globally optimised allocation of divisible workload to a (sub-)set of resources.

STATIC VS. DYNAMIC
: The next level in the hierarchy draws the distinction based on the time at which workload is assigned to the resource. In static systems, the assignment is done in an offline fashion: All scheduling decisions are made before the system is started. In dynamic systems, workload is injected over time into the system, and the assignment is done in an online fashion: The scheduling decisions are made repeatedly[3] while the system is already running.

OPTIMAL VS. SUBOPTIMAL
: This property simply separates strategies that yield optimal solutions from those that do not.

APPROXIMATE VS. HEURISTIC
: The former assumes approaches which re-generate solutions to the SRM problem until a certain level of quality is reached, while the latter assumes approaches that rely on a priori knowledge concerning the system characteristics and parameters that affect this system in an indirect way.

DISTRIBUTED VS. NON-DISTRIBUTED
: This aspect introduces the question of the decision making location into the dynamic SRM strategy: The latter case assumes that this process is done on a single installation, while the former case assumes a physically distributed approach.

COOPERATIVE VS. NON-COOPERATIVE
: Here, one distinguishes between individual decision makers in the SRM process that make either dependent or independent decisions. This aspect can also be seen as the degree of autonomy a certain decision maker in the overall system has.

ADAPTIVE VS. NON-ADAPTIVE
: Adaptive solutions utilise strategies that can dynamically change algorithms and parameters depending on the perceived behaviour of the overall system; nonadaptive ones cannot do this.

LOAD BALANCING VS. BIDDING
: In load balancing, the basic idea is to spread the load among a number of resources in a system with the goal to achieve an equilibrium with respect to a certain indicator. Bidding, in turn, assumes that each resource in the system is responsible for offering to and/or claiming workload from other participants in the system on its own.

[3] That is, in a periodic fashion or on certain events.

PROBABILISTIC VS. DETERMINISTIC
: This aspect denotes whether the SRM strategies have a randomised component or not.

ONE-TIME ASSIGNMENT VS. REPEATED REASSIGNMENT
: In this classification, it is differentiated whether workload in the system is scheduled once, and then never touched again, or multiple times, using a reassignment mechanism that may supersede previous decisions.

In this work, we will focus on SRM strategies that are **global**; although we rely on local strategies for making decisions on a single installation, these are not subject to our findings, as the whole topic has been extensively investigated already, see Feitelson et al (2005). Further, we obviously rely on **dynamic** strategies, as the workload induced into the DCI is not known in advance: Even for very large High Performance Computing (HPC) centres, which periodically construct "quasi-offline" schedules using the workload submissions known *so far*, a pure static model would require the operators to plan all allocations from the machine's commissioning until removal from service, *before starting operations*. Due to the fact that workload management in DCIs is a derivative of multiprocessor scheduling, which is known to be NP-complete, see for example Ullman (1975), we discuss **suboptimal** solutions only. Mainly because of the online character of the problem, we follow the **heuristic** approach in this work.

As different application scenarios require different infrastructures, see Hey and Trefethen (2005), and build on different architectures, see Tonellotto et al (2007), we investigate both distributed and non-distributed SRM systems here. This also applies to the remaining properties listed above.

2.3 Classes of Deployment

The second major concern is the way the infrastructure is deployed: The design, implementation, and evaluation of SRM strategies highly depends on the structure of the DCI it is applied to, and in fact, newly developed SRM strategies are usually tailored for a certain deployment model. Again, several taxonomies considering such deployment classes have been developed. Braun et al (1998) introduce a detailed platform model tailored to the structure of HPC systems and focus on application and toolkit related issues rather than on the design of SRM systems. Pugliese et al (2008) discuss a level-based model for SRM systems and develop an architecture that addresses interoperability issues in DCIs.

For this work, we will follow the classification proposed by Krauter et al (2002), see Figure 2.2, which puts the design of SRM systems into focus and splits the model into different aspects regarding the deployment model:

MACHINE ORGANISATION
: The organisation of machines defines the pattern of communication be-

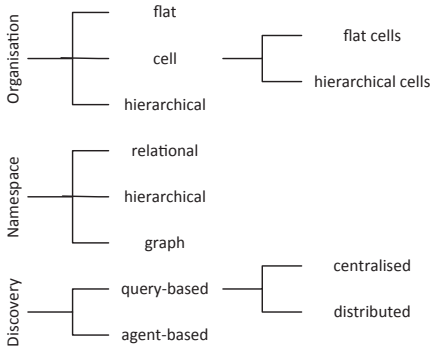


Fig. 2.2 *Taxonomy of SRM deployment classes as proposed by Krauter et al (2002).*

tween the different SRM systems, dictates the flow of information, and therefore influences the decisions being made by the SRM strategies. Usually, machine organisation and resource management are characterised as a single entity; here, these are independently defined. While the flat and hierarchical organisation is obvious, the cell organisation acts as an intermediary: There, resources within a cell communicate in a flat manner, with certain boundary machines acting as gateways to other cells.

Namespace Organisation
: Namespace organisation dictates the pattern for finding potential partners during the computation of an SRM decision and thus defines the decision space. In relational namespaces, concepts from Relational Data Base Management System (RDBMS) are reused; hierarchical namespaces are usually organised around the infrastructure itself. The combination of the two yields a hybrid namespace model which is common in network directory-based environments. The most general model assumes a graph structure, where each namespace element points to others, thus comprising the global namespace.

Resource Discovery
: The discovery of appropriate resources is the first step in any SRM strategy in a DCI. Here, two possible approaches are introduced: query-based and agent-based. While the latter assumes sending active code through the infrastructure to autonomously discover the resource landscape's shape, the former relies on queries to already known partners in order to gather information on the candidate resources.

In this work, we assume a **hierarchical cell** model: On each level of modelled entities, communication between resources belonging to the same cell on the next level is permitted, while other communication is forbidden. This allows us to provide a strict separation of cells and thus model organisa-

tional autonomy of resource centres in the whole DCI. Regarding namespace organisation, we use a **graph**-based approach, as it is the natural model for the afore discussed aspect of repeated reassignment, see Section 2.2. For resource discovery, we follow the **query** model, as the agent model induces a much higher complexity and raises additional questions around security and trust (see Greenberg et al, 1998), which are not subject to the research questions covered in this work.

2.4 Considerations on a New Model

With the context regarding existing models being set, and in order to be able to evaluate the performance of different SRM strategies in DCIs environments, it is necessary to formulate our own, original model by further concretising the properties discussed in the previous sections. On the highest level, we therefore start with machine organisation and assume two general cell levels: local system and federation. The former describes the structure within a resource centre that is participating in a DCI, and the latter describes the relationships between the different resource centres. For the formal introduction of the different components, we use the scheduling notation introduced by Graham et al (1979) and extended by Pinedo (2012), and additionally assume that a federation $\mathfrak{F}$ comprises a set of $|\mathcal{K}|$ local systems $k \in \mathcal{K}$, where $\mathcal{K}$ is an index set of all local systems.

2.4.1 Local System Level

While many different architectural models for the implementation of a resource centre in the context of DCIs are known, the most prevalent ones are Massively Parallel Processing (MPP) systems and clusters.

MPP systems comprise a large number of identical machines that connect to each other via a specialised high-speed network, see Grama et al (2003); often, such systems are referred to as "supercomputers". They provide a very tight integration between the machines with low latency and high bandwidth, effectively allowing inter process communication via message passing with minimal delay. MPP systems involve very high capital and operational expenditure: Among the 500 fastest computers in the world, the "Kei (京) Computer" as the leading system[4] required a ¥ 112 bn investment and consumes another ¥ 10 bn/yr for maintenance. With such high cost and the involved need for specialised applications that are precisely tailored to the system to deliver the highest possible performance, it is not surprising that only about 18 % of these machines are MPP systems; still, they are found in modern production DCIs such as DEISA and PRACE, see Gentzsch et al (2011).

[4] As indicated by the TOP500 list in November 2011, see `http://www.top500.org/`.

Clusters, in turn, can be classified as "networks of workstations," realising communication between the machines via commoditised interconnects, such as 10GE or InfiniBand, see Pfister (2001). Although they are not capable of delivering the same high-bandwidth/low-latency performance as found in MPP systems, they still offer a low-cost alternative for delivering parallel computing. Clusters largely build on high-end, off-the-shelf components, and usually cater standard engineering applications such as Computational Fluid Dynamics (CFD) simulations. In the aforementioned TOP500 list, about 80 % of all systems are clusters; systems with this architecture also are the backbone of most production DCIs, see Ferrari (2011).

Considering the architectural level again, whether looking at Grid or Cloud environments, we can assume that the vast majority of infrastructure providers offer a more or less homogeneous environment. This is mainly due to the fact that resources dedicated to a DCI environment are usually commissioned in a single step. In HPC environments, this approach is crucial as the system is highly specialised and precisely tailored towards providing maximal performance as a whole. For commoditised resources such as clusters, economic considerations and the complexity of the acquisition process leads to few, large investments rather than many small. In Cloud environments, it is the business model that requires homogeneity: The offers made to potential customers need to be sufficiently simple and well-understood to be successful.

Machine Model

Although business considerations already indicate a homogeneous infrastructure, we can also infer this from current statistics. Looking at the TOP500 list, it shows that not only the system architecture is dominated by clusters, but also the machines' processors are very homogeneous: About 70 % of all systems use one of the current Intel Xeon processor families as their sole processor choice.As such, we can safely assume that within a single systems, all resources are in the same order of magnitude regarding their performance.

Hence, we make our first simplification: All resources on the local system level are considered to be identical. We therefore assume that a local system k is built using m_k identical resources. This means that any unit of indivisible workload can be allocated on any subset of resources of the resource centre. Moreover (and in contrast to many real-world installations), we do not allow further partitioning of the resources within a single participant; since we are striving for general-purpose algorithms in this work, we do not want to favour certain machine configurations.

This overall setup is known as a P_m model, see Garey and Graham (1975).

Job Model

In any DCI environment, users induce workload into the system that is then executed on a subset of resources in the infrastructure. As such, any indivisible unit of workload, further denoted to as a *job*, at the very least comprises a

runtime and the number of resources used. Since we assumed a homogeneous resource space on the local level, each job $j \in \mathcal{J}$ (with $\mathcal{J}$ being an index set of all jobs) can run on any subset of local resources. While it is technically possible to allow either the job or the system to change its resource requirements on submission (being "moldable"), see Cirne and Berman (2001), only a very small fraction of jobs use this feature. Changes during runtime (being "malleable") are even more seldom, see Kalé et al (2002). We therefore assume all jobs to be rigid, that is the number of required machines $1 \leq m_j \leq m_k$ is known at release time $r_j \geq 0$.

Every job requires exclusive access to the machines it has been assigned to; the system is used in a space-shared manner, and each job's processing time p_j is only known after completion. Note that this approach does not necessarily hinder a cloud-based approach: For Clouds building on physical hardware, the usage paradigm stays the same since at the end of any scheduling process, some space-exclusive assignment per unit of workload is achieved, typically on the core level. In virtualised systems, the virtual-to-physical assignment is part of the hypervisor layer (which is out of scope of this thesis), and the workload-to-resource assignment on the virtual level is the same as for the physical-only case[5].

In order to prevent faulty jobs from endlessly blocking assigned machines, users are required to provide a runtime estimate $\bar{p}_j$ after which the job is forcefully terminated. In order to prevent premature abortion, users tend to grossly overestimate this value, see Bailey Lee et al (2005); a side effect of this is that this factor almost neglects the deviations from real-world systems that are induced through the uniform resource model.

While most applications utilising such infrastructure are somewhat tailored to the environment, especially in the context of precious compute time on HPC systems, they do usually not allow interruptions during execution[6]. With the advent of virtual machine technology, which is often used as the foundation of Cloud computing infrastructures, the restriction of being non-preemptible was somewhat relaxed; however, even nowadays, the concerted preemption of a group of virtual machines, which is essentially equivalent to a parallel job, is still an open research problem, see Anedda et al (2010). We therefore do not allow preemptions at all, following Feitelson et al (1997).

Although the "job" concept originally stems from batch computing systems, it still is the prevalent unit of work in HPC/High Throughput Computing (HTC) environments; interactive jobs are seldom as they encounter many technical obstacles, see Talwar et al (2003). In Cloud infrastructures, the term "job" has been ousted in favour of "virtual machine," but the basic property of occupation with respect to time and space on the infrastructure

[5] When assuming a layer architecture with strict separation.

[6] With the notable exception of some long-running HPC applications that support checkpointing for recovery purposes, see Holl et al (2009).

remains the same and, from the SRM perspective, again we do not have any interactiveness.

The afore discussed model is widely accepted for modelling the local system level in our use case, see for example Hotovy (1996) or Song et al (2005b).

Scheduling System

At each resource centre, a software component known as the Local Resource Manager (LRM) is responsible for the assignment of jobs to machines. Such systems date back to the mainframe era in the late 1960s, where the manual operation of such machines was replaced by automatic batch manager software such as Operation Planning and Control (OPC)[7]. Nowadays, several software solutions are available on the market; the most popular ones in the HPC/HTC domain are Platform LSF, Univa Grid Engine and Altair PBS, while OpenStack, Citrix CloudStack, and VMware vCloud are the prevalent products in Cloud computing.

Although over the years, LRM software has grown very powerful, addressing a great variety of use cases, the fundamental working principle has stayed the same: Workload is submitted to the LRM by the user, enqueued until a scheduling decision can be made, and eventually allocated to a subset of machines within the underlying infrastructure. Technically, incoming jobs are collected in one or more waiting queues, partly with different prioritisation, and scheduling algorithms decide on the execution order for the jobs.

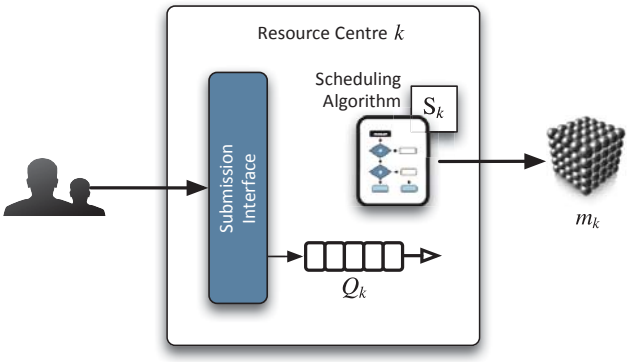


Fig. 2.3 *Schematic depiction of the LRM model assumed in this work. For a given resource centre k, it comprises a m_k identical machines or processors, a single queue Q_k and schedule S_k, a scheduling algorithm, and the submission interface for user workload.*

[7] Now known as IBM Tivoli Workload Manager.

While the model of different queues within the LRM is very valid for practical purposes and greatly simplifies the administration of such systems, it is not strictly necessary to allow this explicit separation, as it often impairs the overall utilisation (see Franke et al, 2010). We therefore make our second simplification here by assuming a single queue Q_k that contains all released but not yet scheduled jobs and a single schedule S_k for the whole resource centre k that reflects the temporal occupation of its machines. This structure is depicted in Figure 2.3; additionally, our model comprises a submission interface through which users induce their workload into the system, a scheduling algorithm that takes care of assigning workload to resources (denoted by the icon), and the bare resources m_k that are exclusively managed by the LRM.

The scheduling algorithm is invoked periodically and allocates jobs from the waiting queue to the schedule. The completion time of a job j in schedule S_k is denoted by $C_j(S_k)$. Note that, for the allocation itself, the scheduling algorithm only has access to $\bar{p}_j$; the real processing time p_j is only known after the job has completed. The wait time of j which defines how long the allocation took can then be derived as $C_j(S_k) - p_j - r_j$. Formally, we try to solve a non-clairvoyant online job scheduling problem, see Tchernykh et al (2009).

So far, our model closely matches a typical environment that can be found in most of the currently implemented real-world systems, see Feitelson et al (2005).

2.4.2 Federation Level

As soon as different resource centres, whether in the same or different administrative domains, decide to cooperate regarding the management and execution of workload, a federated environment emerges. Such federations have already been successfully put into production for many times; Andronico et al (2011) give a comprehensive overview on the world-wide "e-Infrastructure" implementation efforts over the last ten years. From these and other deployments, we can derive a number of commonalities that will be used for shaping our model on the federation level.

Sources of Workload

The first observation regards to the origin of the resources that are exposed within the federation. In almost all cases, we see existing resource centres offering machines that are part of their local infrastructure to users outside their own community. This leads to two important assumptions: First, the workload induced to the overall federation has a local source; that is, users located at the resource centres are submitting jobs to the infrastructure. Second, this workload is then dispatched to either the local machines or, if covered by the federation policies, delegated to other participants in the federation.

Both assumptions also comply with the concept of Virtual Organisations (VOs) that is found in Grid-related environments or "tenants" in the

domain of Cloud computing. In both models, users sharing a common goal (such as working at the same company or investigating the same research problem) are grouped under a common administrative domain. One purpose of this approach is to allow users to use shared resources independent from their affiliation, that is being able to collaborate without having to share the same identity realm. For infrastructure providers, it enables domain-based access control: With this model in place, they can permit resource usage to all members of a VO, without having to know each member of it[8].

Because of the grouping of consumers into ephemeral domains that serve a community until the pursued goal is eventually reached[9], we assume that each resource centre k has its own source of workload comprising jobs $j \in \tau_k$, essentially catering the local[10] user community. Naturally, this approach neglects the fact that usually, users within a single VO or tenancy are not necessarily located at the same geographical place. However, since our focus lies on SRM, we allow this simplification, as characteristics stay the same and the channelling of all workload from a certain domain through a single submission point is a common approach in many portal-based DCI environments, see Oleksiak and Nabrzyski (2004).

Properties of Infrastructure

Our second observation concerns the infrastructure itself: All resource centres within such a federation are independent from each other and only agree on a loose integration. This is all natural: Since every proprietor of expensive IT infrastructure has invested enormous amounts of money into its set-up, he will mainly pursue his very own interests. As such, it is unrealistic that the operator of a resource centre is going to cease control over the system by allowing other participants to interfere with the LRM decisions, see Franke et al (2006b). On the contrary, every participant will be interested in delivering good performance to the catered user communities while reaching a high operational efficiency. From this fundamental idea, again we can deduce a number of characteristics.

Resource Heterogeneity

Most importantly, we have to consider that the resource space is potentially heterogeneous. Under the assumption of a loose federation, providers will purchase their systems independent from each other, it is likely that, beyond the local boundaries, the machines are similar, but not entirely

[8] In fact, the decoupling enables both consumers and providers to delegate the creation of credible trust relationships to the domain manager.

[9] For example, the production of data for the Assessment Reports of the Intergovernmental Panel of Climate Change (IPCC).

[10] More specifically, we refer to those users that submit jobs to the system through a certain resource centre's LRM system.

equal. This would however necessitate to consider different job runtimes $p_j^k \neq p_j^l$ for two different systems k, l, see Sabin et al (2003).

Unfortunately, the formulation of a model that correctly maps one to the other is still an open problem with an area of research on its own. In fact, even for processors of the same family, performance does not necessarily scale in a linear fashion. For concrete production systems with well-known applications, Elmroth and Tordsson (2008) solve this problem by benchmarking. Kishimoto and Ichikawa (2004) make a more general attempt to estimate runtimes in heterogeneous environments and combine offline benchmarking and analytic modelling with a binning approach to generalise the estimations. A benchmarking-free prediction is proposed by Iverson et al (1996), who build upon an online-adaptive learning algorithm that uses non-parametric regression to provide approximated execution times. All approaches fail however to provide a reliable model for the general case: Either, they are limited to certain environments in the benchmarking case, or they require extensive parametrisation of the prediction algorithm, resulting in a meta-optimisation problem.

In the light of the similarity of performance per machine in the TOP500 list as discussed in Section 2.4.1 and the fact that the problem of runtime estimation in distributed computing is still an open issue deserving its very own regard, we choose to accept this deficiency of the model and assume all machines in the federation equal, that is $p_j^k = p_j^l$.

Network Connectivity

Next, while having sufficiently fast interconnects within a resource centre, this does not necessarily hold between resource centres in the federation, as every participant will be willing to reuse the infrastructure he is offering to its customers already, but not make additional investments for the sake of the federation. Even in publicly funded research networks like GÉANT, see Uhlig et al (2006), which provide bandwidths of up to 10 Gbit/s per link, one cannot expect the same interconnection speed as within a local centre. This fact presents an obstacle for federation usage.

From the computing perspective, parallel jobs cannot be executed in such a way that they overlap resource centre boundaries. The reasons for this are twofold: First, latency impairs communication between the parallel processes, see Kielmann et al (2000), and second, technical issues such as firewall configuration complicate the setup, see Humphrey and Thompson (2002). As such, a model where the whole federation behaves like a very large local system (formally $\sum_{k \in \mathcal{K}} m_k$) must be considered infeasible, and execution of a job is only allowed on the resources of one centre.

From the data perspective, two additional aspects must be considered. Due to the federation character, the application is executed on a remote system. This means that the linked object code needs to be either delivered on submission or pre-installed on the remote system. For the model at hand, we assume the latter: Even with the advent of ubiquitous provisioning techniques

through virtualisation, the proper adaptation to the local system environment is still a challenge, see Dimitrakos et al (2003), as many aspects[11] need to be considered for achieving good performance. The other issue relates to input and output data of the application. Although such data needs to be available only after the final allocation of a job has been made, it still may require a significant amount of time to conduct the necessary transfers. However, this can often be hidden by pre- and post-fetching before and after the execution, see Schley (2003). In that case, the overhead is not necessarily part of the scheduling process. Together with the fact that data scheduling is a separate topic not addressed in this work, we therefore assume an internetworking model that, from the SRM perspective, is sufficiently fast for performing data staging in negligible time, but not fast enough to allow cross-centre execution.

Information Policy

Finally, it is necessary to review the availability of information regarding the dynamic state of each resource centre to the federation, as this data (or the lack thereof) significantly influences the SRM algorithms' decisions. For enterprise environments, i.e. federations that deliver infrastructure within a company, an extensive availability of information (e.g. current load, queue lengths, response times) is usually given: The participating resource centres have an established trust relationship and are part of a single administrative domain.

In the loose integration scenario, however, this cannot be expected. Especially metrics such as system utilisation are often considered sensitive, as they can be important for future infrastructure funding: Future investment decisions often base their reasoning on how well-used a local system is and whether the distribution of applications, users and groups, or aspects like the ratio between internal and external usage are along the lines of the centre's overall strategy, see Franke et al (2006b). Another important reason is secrecy: In the context of businesses using e-Infrastructure, information on submitted workload as it can be extracted from a publicly-visible queue may allow competitors to recognise dedicated research and development efforts. Hence, data is not easily disclosed, and—even within academic federations—kept strictly confidential.

For our model, we therefore assume a strict information policy regarding exchange of dynamic data about the infrastructure. That is, local metrics such as the queue length or current utilisation are not disclosed. Instead, only static data such as a resource centre's size (m_k) is freely available. Besides this, only information strictly necessary for distributing workload is exchanged. Specifically, this implies the characteristics of a job provided by the user at submission time.

[11] Starting from "hole punching" in the firewall and ending with file system mapping for temporary data, that is.

Organisation of Participants

The manifold of DCIs implementations in both the scientific and enterprise domain naturally exposes a great diversity in use cases, requirements, and the underlying middleware. Regarding the organisation of these deployments, however, it is possible to classify them into two coarse-grained SRM system types: *centralised* and *distributed* architectures, see Lepping (2011).

From the users' point of view, this seemingly requires them to adapt their submission behaviour depending on the federation's organisation. In order to keep the model discussed in Section 2.4.2 as is and still allow the flexibility of assessing both architectural paradigms, we make the organisation completely transparent to the users. To this end, we assume that all workload requests are channelled through an intermediate service that forwards the jobs to the appropriate SRM component. In order to leave the users thinking that they interact with their "local" provider (be it a research VO, or the company's tenant), we let this service offer the same interfaces as a traditional LRM would expose for a resource centre. Strictly spoken, the latter violates the separation of the local and the federation level at the resource centres. However, since workload is either passed to the LRM level untouched or delegated to a remote participant, we just channel through workload without any modification. Technically, this can be achieved by providing tailored client Application Programming Interfaces (APIs) implementations of user interface libraries such as Distributed Resource Management Application API (DRMAA), see Tröger et al (2007), the POSIX batch management family of commands[12], or a proxy tool like Apache `libcloud`, see Petcu et al (2012).

Centralised SRM

In this model, a centralised SRM system—often referred to as a "meta-scheduler"—accepts the submission of all workload within the federated environment regardless of its origin. It then decides on the distribution among those resource centres that fit the job's requirements, and delegates the execution to the SRM system of the selected participant. The latter in turn performs the job-to-machine assignment.

This model is very common in industrial setups, where usually all resource centres are part of the same administrative domain, but also applies to academic DCIs (Jones, 2005; Barkstrom et al, 2003) and middlewares (Marco et al, 2009; Llorente et al, 2006), as it is easy to implement. An obvious specialisation includes multiple levels, comprising a hierarchy of schedulers. There, schedulers on a higher level assign work to schedulers on a lower level, typically using mechanisms such as direct resubmission or decision negotiation.

The main advantage of this concept is the ability to incorporate different scheduling strategies and policies on the different levels as well as a global view of the top-level SRM layer regarding submitted workload. However, to

[12] As described in Part 6 "Shell and Utilities" of DIN/EN/ISO/IEC 9945.

facilitate a globally effective and coordinated allocation, local resource centres have to delegate administrative control to the federation layer; otherwise, only best-effort Quality of Service (QoS) can be guaranteed for the DCI.

Distributed SRM

Contrary to the previous organisational type, each participating resource centre runs its own SRM component responsible for federation-level decision making. Incoming workload, regardless whether originating from local or remote, is either assigned to local machines or, by interacting with the other participants' SRM systems, delegated to other centres.

This approach is especially popular in the context of Peer-To-Peer (P2P)-based environments, where the environment changes rapidly, see Cirne et al (2006), and in very large-scale installations that need to cater millions of participants, see Kacsuk et al (2009). In both cases, decision making cannot be left to a central component for obvious reasons, and it is necessary to employ distributed SRM techniques.

The absence of any single point of failure is the main advantage of this approach, making its implementations in theory more reliable: Possible breakdowns of one participant do not necessarily impair the performance of the whole system. This strong point at the same time turns out to be the major deficiency, as the absence of a global system view admittedly requires more sophisticated SRM concepts.

2.5 Model Realisation

We define a DCI federation model for further use in our analysis as depicted in Figure 2.4. Each resource centre caters to its own user community and features separate components for local and federated resource management. Users submit their workload to the federation component (albeit using the same interface as for local submission, thus not changing their behaviour) which in turn communicates with federation components of other resource centres to coordinate workload distribution.

The inner workings of the federation model are left undefined for the moment, as they are specific to the architecture of the underlying DCI and the implementation of the scheduling strategy. Regarding our modeling considerations from Section 2.4, we however can make a number of assumptions already:

- Local user communities at each participant submit a certain fraction of the overall federation workload ($j \in \tau_k$), see Section 2.4.2;
- Even beyond participant boundaries, the federation homogeneous resources among the resource centres ($p_j^k = p_j^l$), see Section 2.4.2;

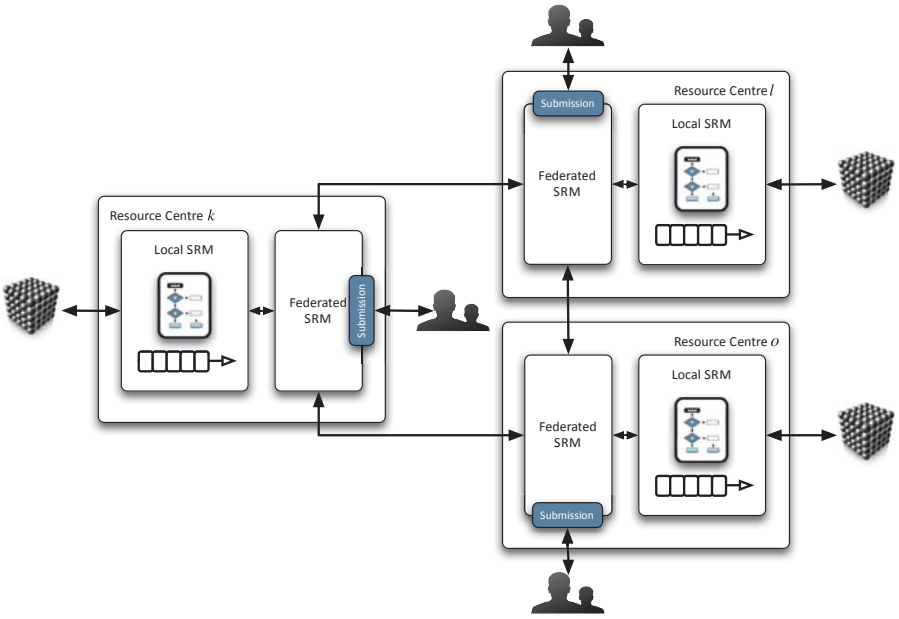


Fig. 2.4 *Schematic depiction of the federation model assumed in this work. Each resource centre comprises the local structure as depicted in Figure 2.3, and additionally offers interfaces for SRM system interaction between the brokers at each centre. In the following, we will use a simplified version that omits the inner structure of the resource centres.*

- Network traffic that relates to data staging is not penalised with respect to SRM decision making (effectively making data transfers "free"), but cross-participant executions is forbidden, see Section 2.4.2; and
- The disclosure of information about the participants' performance indicators is strictly controlled, especially regarding dynamic data, see Section 2.4.2

The afore depicted federation model is deliberately designed to be agnostic of the participant organisation, see Section 2.4.2. That is, it makes no assumptions on the architecture of a consuming DCI. This way, it fits to the architecture of modern production DCIs, and we have a framework at our disposal which can be refined for the concrete analysis later in this work. It also answers Question 1 of our original research questions.

3

Methodology for Evaluation of SRM in DCIs

Profile. Don't speculate.
—Daniel J. Bernstein

Assessing the possible performance gain achievable through federated capacity planning in DCI environments is a complex issue. Before starting to investigate the various shapes of the introduced model and their performance, it is necessary to define several aspects of evaluation to be able to generate expressive results.

Most importantly, the *method of evaluation* needs to be selected, as this is the predominant factor in the overall assessment process. In order to make an informed decision on this, the benefits and drawbacks of different methods are discussed in Section 3.1.

Another fundamental aspect lies in the *selection of input data* for the evaluation made: The quality of traces used for assessing different SRM techniques for the different types of infrastructures discussed in Chapter 2 highly influences the quality of results. The background on this is discussed in Section 3.2.

The assessment of any system requires the definition of proper *assessment metrics*. Depending on the environment in which these are being applied, it is important to understand what they express for the given scenario and in which context they can be compared. Since they depend on the evaluation method and input data selection, the metrics used for the analysis in this work are discussed in Section 3.3.

As a base for comparison, it is necessary to have *reference results* for comparison. For this, a number of non-federated systems are evaluated with regard to their performance when not collaborating in a DCI in Section 3.4.

3.1 Methods of Evaluation

For the assessment of the gain of SRM in the different architectural styles of DCI environments discussed in Chapter 2, we consider three possible approaches for assessment and discuss their assets and drawbacks for the problem at hand.

3.1.1 Theoretical Analysis

The theoretical analysis of job scheduling for parallel machines has been thoroughly researched for at least three decades, and has yielded a manifold of results for various kinds of machine models, workload characteristics, and algorithms; Feitelson et al (1997) give a structured overview on the different results.

When it comes to job scheduling in DCI environments, the vast majority of publications has been made on Grid Scheduling; however, only very few of those disclose results on theoretical issues: Schwiegelshohn et al (2008) have shown that List Scheduling performs significantly worse in Grid environments than on traditional multiprocessor machines and propose an algorithm on the basis of "job stealing" with a competitive factor of 5. Tchernykh et al (2006) use a similar model to the previous work and evaluate the performance of two-stage algorithms with respect to makespan.

The extensive area of Divisible Load Theory (DLT), as surveyed by Bharadwaj et al (2003), has been transported to Grid scheduling by Robertazzi and Yu (2006), who outline the use of flow formulations for Divisible Load Scheduling with multiple sources of (work)load. Other work has been conducted by Fujimoto and Hagihara (2003), who discussed the problem of varying processor speeds—e.g. in Desktop Grid systems such as SETI@home[1]—for sequential independent jobs and propose a new criterion instead of traditional makespan.

Overall, it must be stated that, although provable performance bounds would be the most favourable (and certainly most precise) kind of result in terms of analysis, they have two drawbacks. First, the analysis is very complex even for simple setups, and extremely hard or impossible to conduct for the discussed federation scenarios in DCI environments. Second, performance bounds address a fringe case from the practical point of view: Algorithms with unacceptable worst case behaviour perform well[2] in real world installations. As such, it seems necessary to reach out for other means of evaluation to approach the problem.

[1] Which, in turn, is based on the Berkeley Open Infrastructure for Network Computing (BOINC) architecture, see Anderson (2004).

[2] First-Come-First-Serve (FCFS) is a good example.

3.1.2 Real-World Assessment

Examining the behaviour of a given system in the real world is, from a technical point of view, among the simpler means: Especially software systems used in modern DCI environments offer—by definition of their production character and the accompanying needs of usage audit—a plethora of trace data on their usage.

Again, the prevalent infrastructure for assessment are Grid systems, with a strong notion on compute-centric systems. For example, Iosup and Epema (2006) introduced a framework for system analysis that allows for the automatic generation and submission of workload[3] to a given DCI environment and the evaluation thereof. A similar tool set has been proposed by Frumkin and Van der Wijngaart (2001), who created a tool for Grid space exploration on the basis of data flow graphs. With a slant towards benchmarking, Tsouloupas and Dikaiakos (2005) introduced an operation-centric approach that aims to complement already available data from traditional Grid information services[4]. Chun et al (2004) developed so-called "probes" that exercise basic operations on a Grid in order to measure performance and variability.

In the field of production infrastructures—see Paisios et al (2011) for an extensive list of use cases and underlying DCIs—significant assessment has been conducted on the Worldwide LHC Computing Grid (WLCG) infrastructure, as described by Bonacorsi and Ferrari (2007), where both infrastructure (mainly focusing on data bandwidth and latency of inter-tier transfers) and service reliability and performance are measured. On the Grid'5000 testbed in France, Netto and Buyya (2011) evaluated the performance of a newly developed (re-)scheduling algorithm for "Bag of Task"-style applications. Earlier work by Berman et al (2003) featured the AppLeS project, using applications with a self-scheduling approach[5], and evaluated the performance on real-world systems.

In the area of Cloud computing for scientific applications, most evaluation has been conducted on the Amazon Web Services (AWS) infrastructure. Here, Ostermann et al (2010) provided a performance analysis based on the HPC Challenge benchmark for the 2009 Amazon Elastic Compute Cloud (EC2) services, showing that the overall performance and reliability is still below expectations of traditional HPC centre users. A more middleware-centric approach has been taken by Ostermann et al (2009) with the ASKALON workflow engine by orchestrating the submission of workload to the EC2 services. Again, benchmarking applications such as LINPACK and Message Passing Interface (MPI) Chameleon were used, and certain benefits for very large workflows were discovered.

[3] The data used is, however, limited to synthetically generated workload traces.

[4] See for example Gombás and Balaton (2004); Czajkowski et al (2001); Cook et al (2003).

[5] That is, a master dispatches work in fixed-size units to workers using a greedy approach; see also Hagerup (1997).

Summarising, it shows that most real-world assessments focus on benchmarking and exploit "what-if" scenarios for job execution, but usually not regarding SRM. The few that have been conducted on production infrastructures in order to evaluate the overall workload distribution performance either use very simple strategies (e.g. random assignment or round-robin distribution) or stay on the local level of traditional LRM systems, ignoring the federation aspect. Both approaches (with the only exception of the Cloud computing case) have in common that, by nature of the real-world assessment, they are limited in their evaluation period and do not exceed days of analysis. This, however, comprises a period that is too short[6] for the evaluation of SRM behaviour of a given algorithm. Against this background, the great benefit of a production environment as evaluation scenario is outweighed by the drawbacks of impracticability.

3.1.3 Simulative Evaluation

Simulations are a good compromise for scientific evaluation where real-world observations on the system to be assessed are impractical, while theoretical analysis is too complex. Facing the problems described in the previous paragraphs, it is therefore reasonable to focus on this method of evaluation. That said, it is not surprising that a plethora of scientific research with respect to questions of SRM in DCI environments,has been done already; Dong and Akl (2006) give a comprehensive overview on the results with approximately 100 publications until 2005 alone.

Also in the area of simulation environments—specifically for the evaluation of SRM in DCI environments—many systems have been proposed so far. The most well-known one, GridSim, has been introduced by Buyya and Murshed (2002) and is used in a wide area of simulation experiments focusing on cost: For example, Deelman et al (2008) use GridSim for a cost analysis of applications on the Amazon EC2 services; Kurowski et al (2007) build an extended framework for Grid scheduling on the basis of GridSim; and Singh et al (2007) do cost optimisation on a seismic hazard application using best-effort resource availability. A variant of GridSim with a stronger focus on Cloud computing is CloudSim, see Calheiros et al (2011), which allows for a more data centre-related view on infrastructure. Adoption of CloudSim, however, is little at the moment, as its maturity has not been proven independently from the original authors.

Other research groups developed and released similar tools. For example, Casanova et al (2008) proposed SimGrid, a framework focusing on cluster, Grid, and P2P system algorithms and heuristics. Dumitrescu and Foster (2005) introduced GangSim, a simulator allowing to study the behaviour of VO schedulers regarding scheduling and resource usage policies. Guim et al

[6] See Feitelson (2002a) for details on long-term patterns in recorded workload traces.

(2007) used the Alvio simulator to model the impact of resource sharing in backfilling models, using different resource selection policies. Bell et al (2003) took a slightly different approach with OptorSim, providing a simulator that focuses on data replication issues. With all of the aforementioned simulation tools, a significant number of results has been produced and published over the last ten years.

While simulation is the method of choice for many researchers, it has several drawbacks. First, preciseness and practicability of a model are conflicting aspects, and finding a good compromise between them in order to make simulations feasible and results interesting at the same time is a challenging task which requires great care in the design of the model. Second, simulations rely on input data that induce the system behaviour, and although such data is often available, its use needs additional consideration regarding its impact on the simulation, as certain aspects in the model might not be part of the input recording, and vice-versa.

Considering the different methods of evaluation, the simulative approach still seems to be the most promising one for the assessment of SRM in DCIs. Albeit incomplete by definition, the model described in Chapter 2 has been sufficiently simplified to be tangible, but is yet interesting enough to produce valid and useful results.

Naturally, a simulative approach to the evaluation of SRM in DCIs requires making a choice regarding the evaluation toolkit to be used. With the afore discussed availability of a number of simulation environments, it is necessary to review their suitability to the problem space. In this work, we target a specific scenario: independent participants that form a loosely coupled federated DCI environment, see Section 2.4.2, with each of the participants providing a local resource with a number of processors, see Section 2.4.1. This introduces the following requirements:

Loose coupling of scheduling layers
: We assume a loose coupling of the different scheduling layers on each participant's system as well as between the federated systems. This is important for the strict information policies outlined in Section 2.4.2.

Precise modelling of the decision workflow
: Due to the strong focus on SRM algorithm performance, it is very important to fully understand the decision making process of the SRM algorithms not only on the LRM level, but also regarding the interaction along the different layers and between participants within the federation.

Long-term simulation capabilities
: The proper assessment of SRM algorithms is possible only if a long-term analysis can be conducted. While days or weeks of scheduling decisions may be interesting from the perspective of decision throughput, the efficiency of an SRM algorithm regarding consumer and provider satisfaction only shows after weeks or months of evaluation, as the systems discussed

in this work have starting periods during which users adapt their applications to the new environment.

Rationale for a Dedicated Toolkit

Out of the various simulation environments mentioned above, two require a more detailed discussion because of their general fitting to the afore described problem space.

GridSim

The first simulator is GridSim, which is a very widespread tool used in many research approaches. It allows for modelling Grid environments down to the single machine and supports both time-sharing and space-sharing configurations for their usage. Each participant (including the consumers) runs a so-called *Resource Broker* which takes care of matching resource and workload sets on the basis of cost minimisation for each participant independently, thus working towards a compromise solution for the allocation problem. However, all decisions are broken down to the comparison of cost, and it is required that all participants are able to formulate a specific cost function for their needs. Given the fact that recorded workload traces are used for the assessment, this is a requirement not feasible to properly fulfil: In fact, it is not possible to extract potential cost functions, even for groups of users, from the data available. On the contrary, any approach to model these functions (e.g. through standard statistical distributions) would interfere with the implicit behaviour depicted by the recorded trace and thus is not a proper fix to the problem. Besides this, it shows that the GridSim architecture is closely coupled to the cost approach and thus hard to extend towards non-economic models as assumed in this thesis.

SimGrid

The second simulator is SimGrid, which is similarly accepted as GridSim. Its design is much closer related to the evaluation approach used in this context and explicitly allows to introduce new SRM algorithms for assessment both on synthetic and real workload traces. Unfortunately, SimGrid provides almost no support for other DCI architectures than centralised ones, and the extension of SimGrid towards a model as described in Section 2.4 would require significant changes to the simulation framework.

The teikoku Grid Scheduling Framework

The need for a lightweight, yet extensible toolkit for simulating the federation of DCIs regarding the quality of SRM algorithms led to the development of the teikoku Grid Scheduling Framework (tGSF). tGSF is a research environment that has been designed with the following goals in mind:

Modular architecture
: Because of the manifold of experiment configurations possible even within the limited scope of this work, modularity was one of the main considerations during the design of tGSF, as it allows reuse of components for different scenarios.

High performance
: Due to the large workload traces used for the analysis in the work at hand, it was of utmost importance to have a research environment capable of handling such data sets in reasonable time: Many experiments require parameter sweeps, such that performance was crucial.

Easy transferability
: Successful experiments and their results need to be transferred from the simulation environment to physical systems. To ease this process was an additional concern during design: It should be simple to test the same algorithms in a real test-bed after verifying their general performance in the simulator.

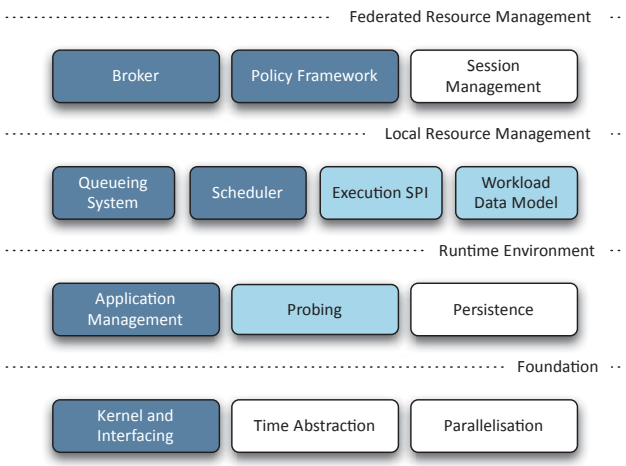


Fig. 3.1 *Architectural overview of tGSF.* Foundation, Runtime Environment, Local Resource Management, *and* Federated Resource Management *stack one on the other, where lower layers provide more general components. Cross-domain functionality has been outsourced to external libraries not displayed here. The dashed lines indicate that the layers above and below are completely separate from each other and can work independently. The coloured boxes indicate primary (dark) and secondary (light) components; the other boxes comprise components that are reusable independently of tGSF.*

Figure 3.1 shows the overall architecture of the framework. Starting from the bottom, the first layer handles aspects of the simulation system and is

agnostic of any SRM-related questions. Only the handling of discrete events for the simulation itself, the decoupling of real time and simulation time, and parallelisation of the engine for large-scale simulations is done here. In the next layer, application management (starting and stopping simulations, binding workload sources to the simulated environment, etc.), probing the system's performance (both on the framework and the business level), and persistency handling (flushing simulation trace data to a file or database) is done. The third layer provides abstractions for the first business layer in the system, modeling a single resource centre with its properties, the LRM system it runs, and the data model for describing workload down to the properties of a single job. In the topmost layer, the federation aspect is implemented through a broker component that encapsulates the communication regarding other participants in the federation, a policy framework that abstracts decision making, and session management for distributed, concurrent brokering with many participants.

For each layer, it has been ensured that independent usage (i.e. without the other layers) is possible; the only dependency is being introduced by externalised, independent common component libraries which provide cross-concern functionality. An in-depth discussion of the architecture of tGSF can be found in the appendix, Appendix A.

3.2 Selection of Input Data

The choice of simulative evaluation immediately raises the question on input data, which—being the representation of workload run in a given DCI environment—significantly influences the behaviour and perceived performance of SRM algorithms and methodologies.

The two options at hand for such input data are *synthetic* and *recorded* workload traces, and both of them have their assets and drawbacks. Regardless of which option is being followed, the evaluation of SRM algorithms needs to be done carefully[7], as Frachtenberg and Feitelson (2005) fortify with a number of potential pitfalls.

3.2.1 Synthetic Workload Traces

Attempts to correct statistical modelling of workload traces for DCIs date back almost 30 years, and the finding of good models for parallel machine environments is by far the predominant area of research. In most cases, the models attempt to identify and parametrise statistical distributions for distinct properties of a workload (e.g. the degree of parallelism or the user-estimated runtime). Sampling from such distributions then essentially produces a new, synthetic workload trace, see for example Feitelson (2002b). However, it shows that this approach is too simplistic to create data sufficiently close to real-world recordings.

A main issue here is that the properties are often modelled independently from each other, although there exist certain correlations between the different properties in recorded traces. For example, parallelism and runtime are often related to each other, and Lo et al (1998) have shown that this correlation has significant effects on the quality of SRM algorithms, proposing a coupled modelling of them. Later work by Li et al (2007) revealed importance of integrating statistical distributions with correlation models to properly satisfy the interdependencies of different properties. Although not a directly visible property of the workload, the inter-arrival time[8] is another important aspect of the workload that needs to be properly considered: Jann et al (1997) have discussed its importance and show that its correct modelling requires different distributions for different degrees of parallelism. Dependencies also can be identified between a job's runtime, its runtime estimate and the degree of parallelism: Lublin and Feitelson (2003) have shown a connection between the modelling of runtime and the "power-of-two" clustering in parallelism[9]; Song et al (2004) extended this to runtime estimates as well. Tsafrir et al (2005),

[7] Although the cited work focuses on parallel machine scheduling, the results are still valid and important for the scenarios discussed in the context of this work, since many DCIs have their roots in the domain of HPC.

[8] That is, the time between the release of two consecutive jobs within the workload.

[9] See Downey and Feitelson (1999) for a comprehensive analysis of this effect.

who use a slightly different approach to modelling the runtime estimates, confirmed the observation that there is a strong correlation between estimated and real runtimes.

However, even approaches that correlate certain properties with each other are not able to delineate the full scope of workload behaviour: Feitelson (1996) has shown that it is possible to identify groups of almost identical jobs in the workload. This behaviour can be directly traced back to the user habit of submitting a number of instances of the same application more or less at the same time[10]. As such, the correlation space needs to be extended to sequences of submissions, and leads to the use of Markov chains. Song et al (2005a) made a first attempt of improving workload modelling with this technique, and Krampe et al (2010) extended the use of Markov chains towards developing one of the first comprehensive models.

While the field of workload modelling for parallel machine environments is well-covered, research specifically for DCIs is still rare. Medernach (2005) did some pioneering work here, again using Markov Chains for modelling inter-arrival times, but Jagerman et al (1997) have shown that this approach leaves room for improvement regarding the correctness of time autocorrelations; Li and Muskulus (2007) further refined the model towards eliminating this deficiency. Still, all approaches are limited to the submission time property and do not properly honour other aspects.

Overall, the current state of the art does not seem to be sufficiently sound to use them as input for the evaluation of SRM: Even for parallel machine environments without integration into a DCI, no model allows for expressing all relevant parameters. As such, they are clearly unfit for even more complex, federated scenarios.

3.2.2 Recorded Workload Traces

Recorded workload traces do not suffer from the problems mentioned above: they contain all information that was available on the system on which the logging has been made, include all possible properties the LRM is able to expose, and implicitly have all hidden characteristics in place, provided that all relevant data has been recorded in the first place.

However, the acquisition of recorded workload traces from real-world systems is a challenge on its own: The accounting data formats of well-established LRM implementations vary greatly, and although considerable effort has been put into the standardisation of these, see for example Mach et al (2007), market adoption is still very limited. Another issue is privacy: Most LRM operators do not feel comfortable giving out these data. The obvious reason is the protection of users; another one is that full disclosure of the original accounting records of any given system provides very deep insight into organisational structures and operation flaws.

[10] A typical use case for this are parametric sweep experiments.

For that matter, only very few publicly available collections of workload data are available: the Parallel Workloads Archive[11] and the Grid Workload Archive[12].

The Parallel Workload Archive

The Parallel Workloads Archive offers 30 traces that have been recorded on parallel machines in different locations since 1993, and has become the de-facto standard for research on this kind of data (more than a hundred publications build upon the workloads published there). For each workload trace, the archive offers an anonymised version of the original trace data, and one version converted to the Standard Workload Format (SWF) as defined by Chapin et al (1999) for greater convenience. For some workloads, a third version (again using the SWF format) is available, which is a "cleaned" copy of the original SWF conversion. While the non-cleaned SWF trace is sanitised only with respect to syntactically broken records in the original workload trace, the cleaned version also purges anomalies due to administrative intervention or abnormal user behaviour. Feitelson and Tsafrir (2006) explain the method used for cleaning, and—backed by comprehensive analysis of the behaviour of the cleaned versions—recommend the so revised traces for exclusive use.

The Grid Workload Archive

The Grid Workloads Archive focuses on traces that have been recorded on academic Grid environments. It offers 11 traces, and each of them spans a number of participating HTC centres. It provides the trace data in a different format, called the "Grid Workload Format," which allows for additional fields regarding job interdependencies, data and network requirements, VO and project information, and job provenance[13]. The format is partly based on SWF, JSDL, and the Open Grid Forum's Usage Record (UR) format, but defines no community- or user-driven standard and poses an isolated effort.

Archive Selection Rationale

Here, we prefer the former of the two over the latter, for several reasons. First, the federated character of the researched infrastructure paradigm is much better fitted to independent parallel machine setups (as given in the Parallel Workloads Archive) with loose interaction, rather than a fixed Grid paradigm as assumed in the Grid Workloads Archive. Second, the traces in the Parallel Workloads Archive are much better understood and well-researched in terms of anomalies and side effects to SRM analysis than the ones in the

[11] `http://www.cs.huji.ac.il/labs/parallel/workload/`.
[12] `http://gwa.ewi.tudelft.nl/`.
[13] That is, information on the submission and execution system, and intermediaries.

Grid Workload Archive. Third, the Grid Workloads Archive traces contain a large amount of information unneeded for the research within this work; especially, data and network usage information is not considered at all. Finally, it must be noted that the traces from the Grid Workloads Archive lack interestingness regarding SRM algorithm design, since most of them have an overall resource usage below 50 %. In such cases, scheduling degenerates to simple batch queueing with (almost) no delay, and it does not make sense to introduce sophisticated heuristics for better performance in the federated case, since each resource centre obviously provides sufficient resources to its users anyway.

3.2.3 Utilised Workload Traces

For the work at hand, we use a selection of traces to conduct the SRM algorithm assessments. As reference, we discuss these traces here regarding their origin, a number of key properties, and the sanitisation that has been conducted before usage.

All data sets are taken from the previously discussed Parallel Workload Archive, in their sanitised versions (if applicable). More specifically, we used the following traces (see Table 3.1 for detailed information):

KTH96
: A trace from an IBM RS/6000 SP with 100 processors at the Swedish Royal Institute of Technology, Stockholm. The data has been collected between September 1996 and August 1997 over a period of eleven months.

CTC96
: Another recording from an IBM RS/6000 SP, this time with 430 processors, made at the Cornell Theory Center, Ithaca, NY. The data has been collected between June 1996 and May 1997 over a period of eleven months.

LANL94
: A trace from a Thinking Machine CM-5 with 1,024 processors at the Los Alamos National Laboratory, Los Alamos, NM. The data has been collected between October 1994 and September 1996 over a period of 24 months. This workload exposes two specialties that have been part of the LRM configuration: Only jobs with a parallelism of 32 or greater were accepted for submission ($m_j \geq 32$), and the parallelism was required to be a power of two ($m_j \in \{32, 64, \dots\}$). These restrictions led to a close-to-optimal allocation, resulting in very short turnaround times.

SDSC00
: A recording from an IBM RS/6000 SP with 128 processors at the San Diego Supercomputer Center, La Jolla, CA. The data has been collected between April 1998 and April 2000 over a period of 24 months.

SDSC03
: A trace from the “Blue Horizon” instalment with 1,152 processors, again

at the San Diego Supercomputer Center. The data has been collected between April 2000 and January 2003 over period of 32 months.

SDSC05
: A recording from the "DataStar" instalment with 1,664 processors, yet again at the San Diego Supercomputer Center. The data has been collected between March 2004 and April 2005 over a period of 13 months.

Table 3.1 *Recorded workload traces used in this work, with their properties.*

Identifier	m_k	Period (mos.)	Users	Number of Jobs	
				cleaned	sanitised
KTH96	100	11	214	28,489	28,479
CTC96	430	11	679	77,222	77,199
LANL94	1,024	24	213	122,060	110,769
SDSC00	128	24	437	59,725	54,006
SDSC03	1,152	32	468	234,314	232,281
SDSC05	1,664	13	460	96,089	84,876

Although the recordings are rather old, their age is no hindrance to the use in the work at hand. This is mainly because the SRM algorithms analysed in this context are influenced by the characteristics of the workload, and not so much by the performance of the hardware or the structure of the applications. In addition, the implicit behavioural patterns of local user communities are a very good fit for the federated scenario described in Chapter 2, since they properly model independent groups of users that participate in a federation of infrastructure providers.

The Need for Shortening

The fact that the workloads span different periods of time needs additional attention. While this does not matter in a non-interacting setup where each resource centre is agnostic of others, all federating setups where participants interact on the SRM level need to be aligned with respect to the simulated period. The reason for this is quite obvious: If differing time periods were used, results would be greatly distorted from the moment one resource centre is running idle. Actually, this would allow all other participants to deploy workload to the idle resource centre, effectively improving their performance (since they have less workload to handle). Naturally, such a setup is unrealistic: Infrastructure installations do not run idle until being decommissioned, due to the large financial investment for their acquisition, but instead are in full service until the end of their use period.

To overcome this obstacle, we created shortened versions of the original traces, which we then use for different mix-and-match setups. Table 3.2 pro-

vides a list of these modified traces; the added index on the name indicates the length of the workload in months, for later reference.

Table 3.2 *Shortened versions of the traces described in Table 3.1, with their properties.*

Identifier	m_k	Period (mos.)	Number of Jobs
$KTH96_{11}$	100	11	28,479
$CTC96_{11}$	430	11	77,199
$LANL96_{24}$	1,024	24	110,769
$SDSC00_{11}$	128	11	29,810
$SDSC00_{24}$	128	24	54,006
$SDSC03_{11}$	128	11	65,584
$SDSC03_{24}$	1,152	24	176,268
$SDSC05_{11}$	1,664	11	74,903

The Need for Sanitisation

Even although the Parallel Workloads Archive traces have been cleaned regarding irregular entries (see Section 3.2.2), a closer analysis of the data reveals that, for proper simulation, additional sanitisation steps need to be done.

First, it shows that not all submitted jobs are eligible for execution: A number of them have a requested number of processors value higher than the total number of processors theoretically available on the machine, that is $m_j > m_k$. Obviously, such requests cannot be fulfilled at all and most certainly are the result of misconfiguration in the LRM[14]. Lines in the trace that fall under this category were removed.

Second, some jobs have different values for their requested and allotted number of processors. This discrepancy is not invalid by itself; for workloads that contain moldable or malleable jobs, such configurations are very common. However, in case of the systems under review, it is unclear whether such jobs were allowed, and if so, which of them were actually exploiting this option rather than just having illicit values in the trace for other reasons. Actually, three cases need to be considered here:

a. Lines in the trace that have a number of processors requested, *none* allotted, and run successfully. Here, the latter was set to the value of the former; and
b. Lines in the trace that feature more requested than allotted processors, or vice-versa. Here, the former was set to the value of the latter.

[14] In fact, all modern LRMs can be configured such that jobs with resource requests not eligible for execution are discarded directly, before being accepted for submission.

c. Lines in the trace with invalid values for both fields were discarded.

Third, there exist entries in the workloads with a processing time $p_j \leq 0$. Since such jobs have no influence on the schedule (they obviously do not contribute to the overall resource usage), we removed them from the trace.

Fourth, the workloads in use for this work expose so-called "runtime estimates" provided by the user. A number of jobs have longer actual runtimes than initially requested ($\bar{p}_j < p_j$). The soundness of such jobs largely depends on the configuration of the underlying LRM: The so-called "overuse" of resources is allowed only in few cases; usually, jobs that have runtime estimates lower than their real runtime are forcefully terminated by the LRM. Again, it is difficult to assess whether such lines in the trace are the result of overuse allowances in the system, or just wrong recordings. As such, we discarded lines in the trace exposing such values.

Fifth, there exist jobs with no runtime estimate at all, although the system requires one to be set. Since such workloads are supposed to be usable with scheduling heuristics that built upon estimated runtimes, jobs without this value set correctly can be considered invalid. Therefore, we filtered out lines in the trace lacking this value as well.

Applying the afore listed rules to the original ("cleaned") workloads from the Parallel Workloads Archive results in modified versions of those workloads; see Table 3.1 for information on the number of jobs discarded due to the additional sanitisation.

3.3 Assessment Metrics

The last precondition for being able to conduct an assessment of SRM in DCI environments is the definition of quality measures for the performance of different capacity planning heuristics.

From a practical perspective, two important viewpoints need to be considered:

The consumer-centric viewpoint
: A user in a DCI environment as assumed in Section 2.4 is mainly interested in having his personal workload executed on the system as fast as possible.

The provider-centric viewpoint
: In contrast to this, a provider within the DCI most importantly wishes to show a good efficiency for the system he provides.

Naturally, the search for proper metrics starts in the area of parallel machine scheduling, as this is the basic model for the infrastructure in the here discussed federation scenario. Classical measures from scheduling theory such as Total Completion Time $\sum C_j$ or Maximum Lateness $L_{\max}$ as discussed by Brucker (2007) obviously do not fully address the aforementioned viewpoints; Feitelson (2001) in fact shows that it is necessary to consider properties of the environment to ensure sufficiently expressive objectives.

Following, we introduce a number of metrics that we use in this work in detail. The stems in parallel machine scheduling require special attention: Since there is a difference between the workload *submitted* and the workload *executed* on a participating system in the federation, it is necessary to differentiate the two. To this end, the former is addressed as τ_k, while the latter is addressed by π_k for participant k.

3.3.1 Makespan

For any given workload, the amount of time necessary to run it on a given set of machines is an important indicator for the quality of the SRM system. This quality level is being expressed by the *makespan* of a schedule and is defined as the completion time of the last job. Especially theoretical considerations make use of the makespan, usually to prove upper bounds for the worst case of an algorithm, see for example Garey and Graham (1975) or Naroska and Schwiegelshohn (2002).

For the here discussed problem family, it is hard to qualify the "last" job, as we are considering an online problem with a continuous stream of workload[15] being submitted to the SRM system. As such, we narrow our view to the latest completion time of a certain set of jobs for a given system, see Equation 3.1.

[15] Strictly speaking, the makespan for a given system can be calculated only after its decommissioning.

$$C_{\max,k} = \max_{j \in \pi_k} \{C_j\} \tag{3.1}$$

Another issue is that makespan is location-bound[16]. While this is not a problem by itself, it must be considered when comparing results with respect to a certain user community.

3.3.2 Squashed Area

Another important measure is the amount of resources used, in absolute numbers. This characteristic is defined as *squashed area* (SA as introduced by Ernemann et al (2003)), being denoted as the product of a job's parallelism and processing time as shown in Equation 3.2.

$$\mathrm{SA}_j = m_j \cdot p_j \tag{3.2}$$

Considering again the set of all jobs executed on a given participants machine k, we define the overall squashed area as shown in Equation 3.3

$$\mathrm{SA}_k = \sum_{j \in \pi_k} m_j \cdot p_j \tag{3.3}$$

In fact, the squashed area can be used to express an interesting metric for the federated case as well: The *relative change of squashed area*, see Equation 3.4, indicates resource usage shifts due to delegation, because it describes the ratio between τ_k and π_k independent from the makespan.

$$\Delta\mathrm{SA}_k = \frac{\mathrm{SA}_k}{\sum\limits_{j \in \tau_k} m_j \cdot p_j} \tag{3.4}$$

Grimme et al (2007) show that the relative change of squashed area is a good companion metric for assessing the change from non-federated to federated scenarios. The local usage, however, is still interesting and thus will be discussed in the following section.

3.3.3 Utilisation

The *utilisation* (U) of a resource is an important indicator for the efficiency of a given participant's system. At any given time t, the utilisation is the ratio between used and available resources, see Equation 3.5.

$$\mathrm{U}_k(t) = \frac{1}{m_k} \cdot \sum_{j \in \pi_k^t} m_j \quad \text{with } \pi_k^t = \{j \in \pi_k | C_j(S_k) - p_j \leq t \leq C_j(S_k)\} \tag{3.5}$$

[16] That is, it refers to a certain participant in the federation, and not to a certain workload.

Again, this very metric can be formulated towards including all jobs for a given participant k after all jobs $j \in \pi_k$ have been executed, resulting in Equation 3.6.

$$\mathrm{U}_k = \frac{\mathrm{SA}_k}{m_k \cdot \left(C_{\max,k} - \min_{j \in \pi_k} \{ C_j(S_k) - p_j \} \right)} \tag{3.6}$$

Reusing the qualified makespan in the equation obviously limits the metric to a certain timeframe. Given the online character of the scheduling problem, this seems to be an obstacle here. However, the denominator ensures meaningful bounds for the timeframe by limiting it to the earliest start time and the latest *known* end time for participant k.

For the whole federation, the makespan can be extended even further by embracing all participating systems, see Equation 3.7.

$$\mathrm{U}_o = \frac{\sum_{k \in K} \mathrm{SA}_k}{\sum_{k \in K} m_k \cdot \left(\max_{k \in K} \{ C_{\max,k} \} - \min_{k \in K} \{ \min_{j \in \pi_k} \{ C_j(S_k) - p_j \} \} \right)} \tag{3.7}$$

3.3.4 Average Weighted Response Time

So far, we solely focused on resources with respect to performance measurement of SRM algorithms: Squashed area and utilisation directly show whether the resource was efficiently used. Makespan has a more general view on the system, since it considers how fast a given workload could be handled from the scheduling perspective.

A metric which is independent from the resource characteristics is the *average, resource-consumption weighted response time* (AWRT) as defined by Schwiegelshohn and Yahyapour (2000), see Equation 3.8.

$$\mathrm{AWRT}_k = \frac{\sum_{j \in \tau_k} \mathrm{SA}_j \cdot (C_j - r_j)}{\sum_{j \in \tau_k} \mathrm{SA}_j} \tag{3.8}$$

It should be noted that calculation is done with respect to the whole federation, since the metric operates on the set of jobs *submitted* to participant k (indicated by τ_k in Equation 3.8) rather than the set jobs *executed* at participant k. This way, we analyse the SRM performance improvements of each participant with respect to its original workload and user base.

Seemingly, Average Weighted Response Time (AWRT) gives a more user-centric view on the performance of SRM in the federation. Careful consideration, however, reveals that this is not entirely true: In a system that assumes a uniform resource space (as being done in this work, see Section 2.4.1), the only factor that actually affects the overall performance is the *wait time* as a

direct indicator of how efficiently the SRM system is able to schedule[17]. Still, the resulting average is just an indicator for the overall response time of the system, and cannot be interpreted as an estimate how long a specific user has to wait for a specific job.

Uptaking the views discussed at the beginning of the metrics discussion, see Section 3.3, it shows that the metrics cannot be directly assigned to either of the views. Rather than that, they show a certain bias in their expressiveness towards one view or the other, but—due to the connectedness via the wait time—cannot be clearly separated.

[17] Processing time is meaningless here—in the local case, there is no choice to run the job on a system with higher processing power, and in the federated case, all systems are assumed to be equitable in their processing power, see Section 2.4.2.

3.4 Reference Results for the Non-federated Scenario

In order to determine the performance gain of federated capacity planning mechanisms, we compare their behaviour regarding the assessment metrics described in Section 3.3 with the non-federated case, as this is the lower bound for the performance of SRM systems in a federation as discussed in Question 5. To this end, we assume a scenario where users only submit workload to their local resource centres, without the provider having any collaboration agreement with others in place.

To have a proper basis for comparison, this non-federated scenario should be as realistic as possible. From an SRM perspective, the main influences are the configuration of the LRM system regarding the decision strategy of the internal scheduler, see Section 2.4.1.

3.4.1 Classic LRM Algorithms

For the reference results, we use three popular[18] standard heuristics as the basis for further comparison:

First-Come-First-Serve

FCFS selects the first job in the waiting queue of the LRM system and, as soon as sufficient processors are available, schedules it for execution.

Table 3.3 *Reference results for AWRT, U, and $C_{max,k}$ for the input data described in Section 3.2.3 using the FCFS algorithm.*

Trace	AWRT (s)	U (%)	$C_{\max,k}$ (s)
$KTH96_{11}$	418,171.98	68.67	29,381,344
$CTC96_{11}$	58,592.52	65.70	29,306,682
$LANL96_{24}$	12,265.92	55.54	62,292,428
$SDSC00_{11}$	266,618.82	71.53	30,361,813
$SDSC00_{24}$	2,224,462.43	75.39	68,978,708
$SDSC03_{11}$	72,417.87	68.58	29,537,543
$SDSC03_{24}$	166,189.54	73.22	64,299,555
$SDSC05_{11}$	70,579.08	60.12	29,380,453

While this rather simplistic strategy yields surprisingly good results in many cases, see Schwiegelshohn and Yahyapour (2000) or Grimme et al (2008c), its worst-case behaviour can result in unreasonably low utilisation,

[18] See Ernemann et al (2005) for a survey on SRM environments for the world's 500 then fastest supercomputers. Although other algorithms are used in practice as well, they mostly focus on prioritisation of certain user groups, which is out of scope for this work.

see Schwiegelshohn and Yahyapour (1998a). Still, FCFS is of practical importance because of its popularity among system administrators, which stems from the predictable behaviour, the ease of deployment (as it lacks any configuration), and its implicit fairness (Schwiegelshohn and Yahyapour, 1998b). Reference results for the utilised input data using FCFS are shown in Table 3.3.

Extensible Argonne Scheduling sYstem

Extensible Argonne Scheduling sYstem (EASY) is a widespread variant of *Backfilling*-based algorithms. It essentially allows to schedule any (obeying the order of the queue) but the first job for immediate execution, provided that the first job will not be delayed.

Table 3.4 *Reference results for average weighted response time, utilisation, and makespan for the input data described in Section 3.2.3 using the EASY algorithm.*

Trace	AWRT (s)	U (%)	$C_{\max,k}$ (s)
$KTH96_{11}$	75,157.63	68.72	29,363,626
$CTC96_{11}$	52,937.96	65.70	29,306,682
$LANL96_{24}$	11,105.15	55.54	62,292,428
$SDSC00_{11}$	73,605.86	73.94	29,374,554
$SDSC00_{24}$	112,040.42	81.78	63,591,452
$SDSC03_{11}$	50,772.48	68.74	29,471,588
$SDSC03_{24}$	70,774.86	73.91	63,696,120
$SDSC05_{11}$	54,953.84	60.17	29,357,277

More formally, let t be the current moment in time, j the job at the head of the queue, and T_j the earliest possible start time of j, calculated from the runtime estimate $\bar{p}_k$ of the already scheduled jobs. If j cannot be started immediately, that is $T_j > t$, the first job i from the waiting queue which satisfies $t+\bar{p}_i \leq T_j$ is scheduled for execution, provided that sufficient resources are available. This translates to two criteria for jobs that can be backfilled, that is (1) short jobs which fit into the free space until the occupation of the first job in the queue is expected to start and (2) narrow jobs which do not affect the first job because their parallelism is sufficiently small to fit next to it.

The rationale behind is the fact that jobs with a high degree of parallelism tend to congest the queue with respect to jobs with a low degree of parallelism, and EASY tries to bring the latter forward without delaying the former. Lifka (1995) provides a comprehensive description of EASY with performance results. Reference results for the utilised input data using EASY are shown in Table 3.4.

List Scheduling

List Scheduling (LIST) as introduced by Graham (1969) denotes a generalised variant of a greedy scheduling algorithm: The LIST algorithm iterates over the waiting queue and, in the order of assigned weight[19] for each job from head to tail, schedules every job for execution as soon as sufficient resources are available. As such, the algorithm's behaviour is identical to the *First Fit* strategy, see Aida (2000).

Table 3.5 *Reference results for average weighted response time, utilisation, and makespan for the input data described in Section 3.2.3 using the* LIST *algorithm.*

Trace	AWRT (s)	U (%)	$C_{\max,k}$ (s)
$KTH96_{11}$	80,459.27	68.72	29,363,626
$CTC96_{11}$	53,282.02	65.70	29,306,682
$LANL96_{24}$	11,652.44	55.54	62,292,428
$SDSC00_{11}$	74,527.98	73.96	29,364,791
$SDSC00_{24}$	116,260.72	82.26	63,618,662
$SDSC03_{11}$	48,015.99	68.87	29,413,625
$SDSC03_{24}$	72,220.28	73.91	63,695,942
$SDSC05_{11}$	53,403.51	60.20	29,339,408

In contrast to EASY, it does not enforce a non-delay policy for the first job, potentially causing starvation for it. Reference results for the utilised input data using LIST are shown in Table 3.5.

3.4.2 Discussion of the Results

It shows that EASY outperforms the FCFS heuristic in all cases with respect to AWRT, and in most cases with respect to Utilisation (U); in fact, this confirms the observations made by Feitelson and Weil (1998). Where our experiments yield the same results for FCFS and EASY with respect to U, this stems from the structure of the workload: Their makespan is influenced only by the last job in the original trace (which, in turn, has a very late release time), and thus utilisation cannot improve through better scheduling, but is determined by the submission only.

In some cases, LIST outperforms the other heuristics, but overall EASY turns out to be the best-performing LRM algorithm; however, it requires additional information (the runtime estimates) to make its decision. Mu'alem and Feitelson (2001) back these observations in their work as well.

It should be noted that, in the following chapters, an important effect that other research work usually takes into account is not considered here: the notion of timezones. Lublin and Feitelson (2003) show that certain patterns can

[19] As weight, this work uses the job arrival time (or release date).

be observed for the day cycle. For example, around lunch time the submission density is much higher than during night.

For the non-federated case, this pattern has no effect on the results. However, considering the upcoming federated scenarios, Ernemann et al (2004) show that it is possible to exploit the time difference between the different participants (locations in Europe and the United States of America would be a perfect match), thus getting better results. On the other hand, ever-advancing tools on the user side let this effect fade away, see Li and Muskulus (2007).

In this work, it was decided to not consider timezones, regardless of the afore described positions: Not leveraging the beneficial factor of shifts tends to make the case for algorithm performance rather more difficult than easier.

Part II

Algorithms for Capacity Planning Problems

With the modeling and evaluation background set, we address our second goal by focusing on algorithmic questions regarding SRM for DCIs. To this end, we evaluate the performance of algorithms in the context of different DCI structures derived from real-world installations.

With the insight gained through the definition of our model for federated capacity planning in DCIs as discussed in Section 2.5, we make some general assumptions on the architecture of SRM systems, especially on the decision making process and its layer, policy and interaction model. With that, we set the stage for the analysis of the two most widespread models of SRM: centralised and distributed systems.

First, we review centralised SRM systems, which are popular for production environments because of the ability to construct them along already existing organisational structures. Here, we focus on the specific aspect of information disclosure between and autonomy among the participants of a federation, aiming to approach Question 3 and Question 4 of the original research problem. To this end, we adapt three different approaches—one passive, one active, and one hierarchical—and explore their potential.

Second, we review distributed SRM systems, which became the method of choice for DCI design over the last years because of their flexibility regarding changing user and provider communities. There, we focus on the lack of global view for the decision making strategies and explore their potential, attempting to answer the resiliency aspect of Question 2. On the one hand, we adopt the lessons learned from the analysis of centralised SRM, introduce two distributed heuristics with different information requirements and compare them to each other. On the other hand, we introduce a novel approach for dynamic, situation-aware SRM systems that are able to self-adapt to system usage pattern changes. Additionally, we relax the management model for resource centres, allowing them to lease resources to other participants instead handling their workload, and evaluate two algorithms for this.

Finally, we critically review the different approaches, attempt a qualitative comparison among them, and discuss their applicability, implementation effort, and requirements for production use, targeting the infrastructure aspect of Question 2. In addition, we explore the limits of possible gain in order to set a cornerstone for what can be achieved by modern SRM systems, tackling Question 5. To this end, we approximate those limits for small federation scenarios using evolutionary computation and identify approximate bounds for SRM in federated DCI environments.

4

General Assumptions

Architecture starts when you carefully put two bricks together.
—Ludwig Mies van der Rohe

Since the emergence of production e-Infrastructures, a plethora of DCIs in both scientific and enterprise domains have been built. While all being different in their requirements, use cases, and middlewares, there are common aspects to be identified on the architectural and business level.

Kee et al (2004) present an analysis of existing resource configurations and, specifically for the Grid computing use case, propose a platform generator that synthesises realistic configurations of both compute and network properties of such e-Infrastructures. Here we follow a much simpler model that allows us to focus on the SRM problems rather than the overall infrastructure modelling.

In general, most existing DCIs comprise a federation of partners that share a common goal, such as a specific project, and pool their resource centres. Regardless of whether users in such a project have to decide on the distribution of their jobs by themselves, or this is done in an automatic manner, the LRM layer is not adapted to the DCI federation environment. That is, the LRM schedulers themselves are usually incapable of interacting with schedulers from a different administrative domain without additional software, even within the same project. In addition, each participating centre runs its own scheduling systems which is driven by certain, provider-specific policies and rules regarding the planning scheme for incoming workload.

Furthermore, we can assume that there is a general notion of trust between the partners within a collaborating DCI. That is, pooled resources are shared in a collaborative manner and the delegation or exchange of workload is a generally desired effect. However, each resource centre acts strictly on its own: The owners decide how much control they are willing to cede over their resources to a scheduler in the federation and employ their own LRM systems with a given queue configuration and scheduling heuristics. Furthermore, participants do not necessarily expose the characteristics of their job queues to other centres, and direct access to this information is forbidden.

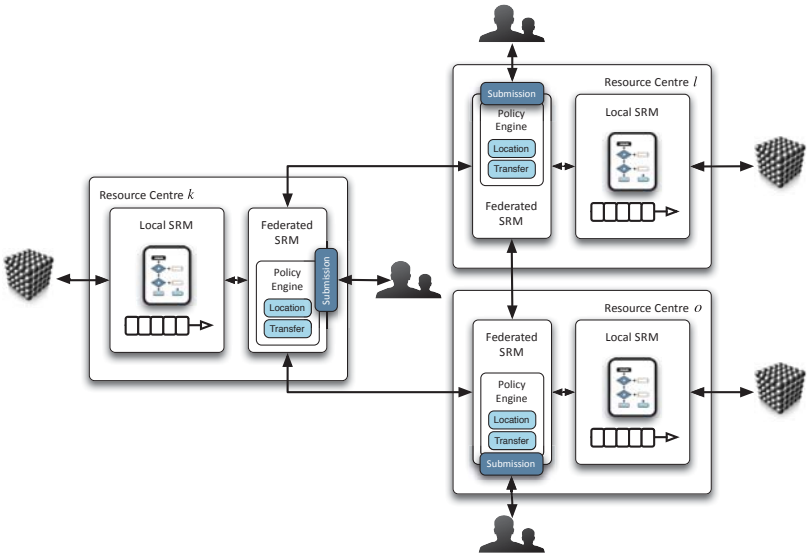


Fig. 4.1 *Structure of the DCI in the federated case. As an extended version of Figure 2.4, it now incorporates the components within the Federated SRM part, namely the Policy Engine with its Location and Transfer policy templates.*

4.1 Layer Model

This federated architecture paradigm requires that each participant exposes an interface of some kind to the other participants, see Figure 4.1. Because of the autonomy requirement, the Federated SRM layer leaves the LRM untouched, thus enforcing a clear separation between the two. For the Local SRM layer, we follow the model described in Section 2.4 such that machine model, job model, and the structure of the LRM stay the same.

The Federated SRM layer is deployed on a per-participant basis and consists of two parts: First, it features a *Policy Engine* component which acts regarding to a set of rules that regulate when to request, offer, or accept remote jobs. Second, it provides interfaces[1] for job migrations on the one hand and job submissions on the other hand. The former acts as a communication endpoint for the other participants in the federation, while the latter ensures that all workload requests, regardless whether they are of local or remote origin, are channeled through the broker component.

In order to make the integration as seamless as possible, we go further than just restricting the information flow between partners in the federation as discussed in Section 2.4.2. To this end, we consequently limit the amount of information about the LRM visible to the policy engine, such that only

[1] For a more elaborate specification of the interfaces see Section A.4 in the appendix.

the current utilisation and the locally waiting workload are known. While the former is typically exposed[2] by most batch management systems and Grid monitoring stacks (e.g. Massie et al, 2004), the latter can be tracked by the decision maker component itself, since it acts as a bridge between the user and the local batch management system. Overall, this ensures a very loose coupling between the LRM and Federated SRM level.

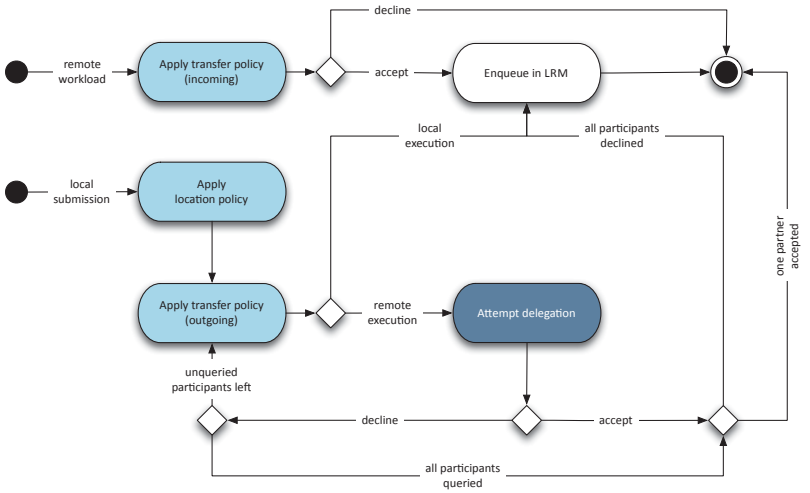


Fig. 4.2 *Decision making process for local and remote jobs. Within the exchange of workload, the location and transfer policies as part of the policy engine are applied.*

4.2 Policy Model

In order to further abstract the policy management within the Federated SRM layer, we split the decision making process into two independent parts. The whole process is depicted in Figure 4.2, and differentiates between workload originating from remote and workload coming from the local domain (i.e. being directly submitted by the resource centre's own users).

In the former case, the decision process is simple: The transfer policy is applied to the set of jobs, and depending whether being accepted or declined, the jobs are enqueued with the LRM or discarded. In the latter case, the decision process needs additional steps: First, the location policy is applied to determine candidates for delegation, and for all candidates, the transfer policy is applied to decide whether the delegation should occur. Depending on the result, a delegation is attempted or the job is directly enqueued with the LRM.

[2] At least within the local system domain, that is.

If the attempt succeeds, the job is handed over; otherwise, the next candidate is queried. Only if all candidates decline, the job is eventually enqueued with the LRM such that users have their workload successfully submitted regardless of the current status of the federation.

4.2.1 Location Policy

This policy becomes relevant if more than two resource centres participate in the federation. In this case, more than one target for workload delegation becomes available, and the location policy determines as a first step the sorted subset of possible delegation targets.

4.2.2 Transfer Policy

After the location policy has been applied a decision must be made whether a job should be delegated to the selected candidate. For this purpose we apply the transfer policy separately to each participant following the result from the location policy. The transfer policy then decides whether the job should be executed locally or delegated to the selected resource centre. In the first case, the job is sent to the own LRM. In the second case the job is offered to the selected partner for delegation. This request can result in two states:

1. The job is accepted by the remote resource centre. In this case, the job is delegated to this partner and no other further delegation attempts have to be made.
2. If the acceptance is declined the transfer policy is applied to the next partner in the list of candidates determined by the location policy. This iterative procedure is continued until all partners in the federation have been requested. If no candidate is willing to accept the job, the offering site must execute it locally.

Further, the transfer policy has to decide about jobs that are offered from remote participants and can choose between accepting or declining a job offer. In the former case, the job is immediately forwarded to the LRM.

On this basis, the policy engine may offer and accept subsets of jobs from the current waiting queue to other decision makers from other sites. Note, however, that the selection which jobs to expose to alien decision makers remains in the domain of the local site.

4.3 Architectural Considerations

With the layer and policy model being set, we now discuss their application regarding different DCI architectures. In Section 2.4.2, two general architectural models have been introduced already: centralised and distributed.

From the layer model standpoint, the two are indistinguishable, since the separation of concerns is happening within each federation partner's resource centre, and the pattern of communication between participants does not matter. From the policy model standpoint, however, this difference is important: In fact, it directly affects the way the location and transfer policies are constructed, as the flow of communication may be organised in a hierarchical or equitable fashion.

4.3.1 Interaction in Centralised Flavors

For the centralised flavor, we distinguish two interaction models. In the active case, the system features a global SRM (the "coordinator"), which decides on the distribution of workload and assigns jobs to the local SRMs for execution (the "workers"), see Figure 4.3. Looking at the policy layer, we find that, with respect to the coordinator/worker relationship, only the former applies a location policy: Workers do not decide on the delegation target unless they act as coordinators towards the next tier[3]. Transfer policies, however, can be in place for both: For the coordinator, they are mandatory in order to decide on whether to delegate to a certain worker or not, and for the workers whether to accept, if such freedom is applicable.

Another interesting feature is due to the flexibility of our layer model: With respect to the origin of workload, the submission of jobs can stay either with the participants' local SRM system (following the grey arrows in the figure)or go directly to the global SRM (following the black arrows in the figure). In the former case, the centralised scheduler *never* receives jobs directly, but instead they are delegated to it by the local systems. Both approaches are fair, as capacity management should be seamless to the users anyway, and the stream of incoming workload does not change its characteristic, as the jobs are passed to the coordinator untouched. Here, we leverage this advantage such that the local SRM is capable of forwarding only those jobs relevant for federated scheduling, instead of sending everything upstream.

In the passive case, the decision making is left to the local SRM systems, but the participants do not interact directly. Instead, they utilise a central exchange platform for offering own and accepting foreign workload, see Figure 4.4. This interaction model is very similar to the whiteboard pattern introduced by Tavares and Valente (2008).

Again, workload submission occurs at the local SRM. Due to the passive nature of the central component, no policies are needed here. Instead, each resource centre features its own policy layer comprising one or more transfer policies and no location policy (since the only delegation target is the exchange platform).

[3] E.g. in a hierarchically organised scheduling architecture.

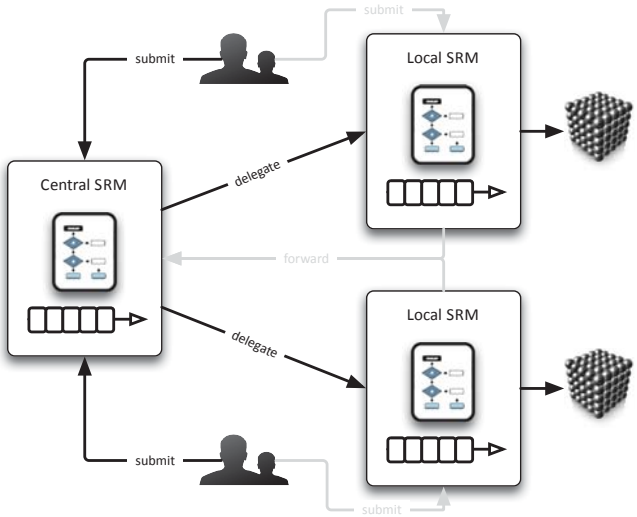


Fig. 4.3 *Structure of the federation for centralised SRM with active interaction. In the hierarchical case, the Local SRMs on the nth tier act as intermediaries towards the n+1th tier by assuming the coordinator role, but stay in the worker role towards the n − 1th tier.*

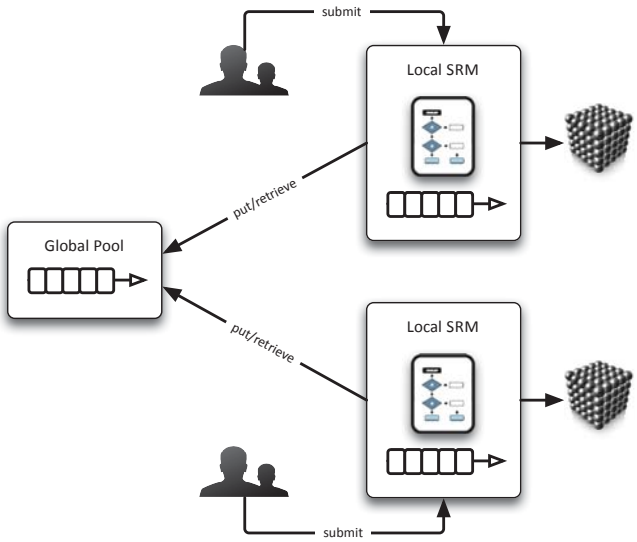


Fig. 4.4 *Structure of the federation for centralised SRM with passive interaction. The Global Pool component only serves as a centralised, ubiquitously accessible data structure that is utilised by the active Local SRM systems.*

4.3.2 Interaction in Distributed Flavors

For the distributed flavor, we only have a single interaction model. Each par-

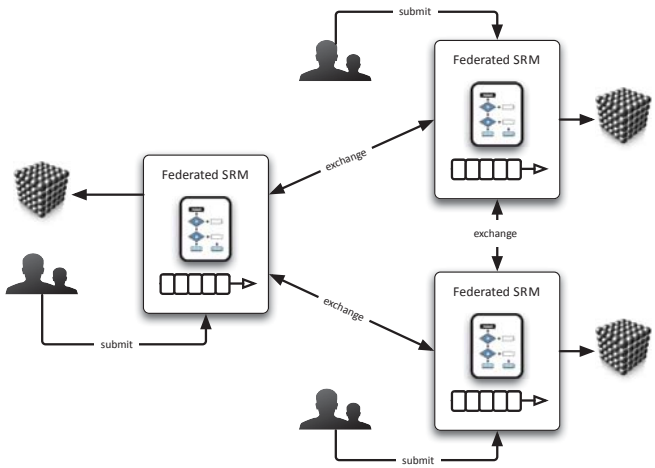


Fig. 4.5 *Structure of the federation for distributed SRM. Here, all participants are actively employed in the SRM process and exchange workload delegation offers and requests via their public interfaces.*

ticipant in the distributed case acts as an autonomous entity within the federation. For our purposes, we assume that every participating resource centre caters to its own, local user community which submits workload to the system. In addition, the SRM system is capable of exchanging workload with other SRM systems in the federation, see Figure 4.5.

Every SRM system acts independently of the others, making its own decisions. For incoming workload, it decides whether it is executed on the local resources, or delegated to other participants, and delegation attempts can be declined by the target system. Each SRM features both location and transfer policies, making decisions on workload that is incoming from local or remote (delegated, that is) submissions.

5

Selected Methods for Centralised Architectures

Uneasy lies the head that wears a crown.
—William Shakespeare

Many production environments consider centralised architectures the model of choice for structuring resources in a DCI. In fact, well-known examples from High Energy Physics (Andreeva et al, 2008; Altunay et al, 2011), Drug Discovery (Chen et al, 2004, 2010), or Computational Chemistry (Dooley et al, 2006), build on this paradigm for many years now. From an SRM perspective, it is interesting to explore the performance of the centralised architecture regarding the interaction models discussed in Section 4.3. With respect to the research questions, we focus on the autonomy and disclosure aspects, and introduce three approaches.

We start with discussing a centralised passive interaction model that allows resource providers to easily connect to a federation while keeping their local scheduling configuration almost untouched, see Section 5.1. To this end, we introduce new variants of popular LRM strategies which communicate via a central exchange.

Next, we compare the performance of the centralised active interaction model under different autonomy and disclosure properties. To this end, we evaluate in how far the federations' performance is impaired if the workers are allowed to decline workload and the master's global view is limited. The resulting implication of restrictive information policies and the amount of autonomy on the local system level is analysed in Section 5.2.

Finally, we review the hierarchical model with changing masters and evaluate the performance of a propagation-based approach in Section 5.3. On the basis of a working algorithm for media streaming in P2P networks, we develop a novel next-neighbour SRM strategy for scheduling in large networks with hierarchical workload forwarding.

5.1 On the Prospects of Passive Workload Exchange†

As a first step towards the design of federated SRM algorithms for capacity planning in DCIs with a centralised architecture, we assume a simple, ad-hoc collaboration use case, where different resource providers which to exchange workload within a federation without the need for an active scheduling component that needs to be maintained.

More specifically, we discuss the passive interaction model, see Section 4.3.1, and its impact on the federation by introducing a global job pool the resource centres in the federation use to cooperate. While workload is being fed into the system via each local LRM system, the local scheduler can now decide whether to execute the job locally or offer it to a pooled queue that all participants use as the collaboration medium for workload distribution (this kind of setup is supported by most modern LRM systems for so-called multi-cluster systems). In the same manner, every participant can pick a job from the pool if there is no local job waiting that fits into the current schedule and can use idle resources. In this analysis we evaluate the performance of three job sharing algorithms that are based on those introduced in Section 3.4.1.

Using slightly modified variants of these popular LRM scheduling heuristics for capacity planning, we show that—compared to the non-federated case—in case of an unchanged local scheduling algorithm many configurations benefit from the ability to additionally use a global job pool, especially in the cases of machines with a high local load.

5.1.1 Pooling-aware Standard Heuristics

With respect to the architectural considerations made in Section 4.3, we assume a *centralised passive* interaction model. The central exchange platform is realised through a global job pool which local schedulers can use to receive jobs from and transfer jobs to. We adapt FCFS, EASY, and LIST to using this platform, and implement the pool as as a global job queue where all jobs are ordered according to their local submission time.

The interaction with the pool requires a policy that decides, whether a job is offered to the job pool or is kept in the local queue. Informally, this policy enforces that whenever a job from the local queue cannot be executed immediately on the local processors, it is offered to the global pool and removed from the local queue. This can be translated to an algorithmic template as described in Algorithm 1, which we then use as the basis for all algorithm variants.

In the beginning, FCFS, EASY, or LIST is applied locally (see Line 2). Following (see Line 4), all jobs that have been considered for the current schedule

† Parts of this section have been presented at the 13th Workshop on Job Scheduling Strategies for Parallel Processing in Seattle, WA on June 17, 2007. The corresponding publication, see Grimme et al (2008c), is available at `http://www.springerlink.com`.

Algorithm 1 General template for the job-sharing procedure

Require: a local job queue l, a global job pool g
1: *algorithm* $\in$ {FCFS, EASY, LIST}
2: apply *algorithm* to l
3: **for all** considered, but locally non-schedulable jobs **do**
4: offer job to g
5: **end for**
6: **if** no job could be scheduled locally **then**
7: apply *algorithm* to g
8: **end if**

during the previous step —including all potential backfilling candidates—but could not be scheduled immediately are moved to the pool. If no job from the local queue could be scheduled, the algorithm is executed again, using the global job pool (see Line 7).

For the modified FCFS policy, only the first job of each queue is considered respectively. In the case of adapted EASY policy, the application of the algorithm is completely separate for each queue. As such, backfilling is repeated on the global job pool if and only if local scheduling failed. Note that only the currently treated queue is used for finding backfilling candidates, and not the union of locally and globally available jobs. The changed LIST policy also iterates over the queue stopping as soon as an assignable job has been found, and also resorts to the global job pool if no local job could be fit into the schedule.

In the remainder of this section we will denote the FCFS based job-sharing algorithm as *P-FCFS*, the EASY based as *P-EASY*, and the LIST based algorithm as *P-LIST* respectively.

5.1.2 Experimental Setup

Before presenting detailed evaluation results on the described scenario, an overview on the setup of the experiments is given, the performance objectives are shortly discussed, and used input data is listed.

In order to measure the schedule quality, we use the metrics defined in Section 3.3 as objectives, namely Squashed Area (SA) and the change of it, and AWRT.

For the performance assessment, we use input data from KTH96, CTC96, LANL96, SDSC00, and SDSC03, spanning machine sizes from 100 to 1,024 nodes. With that, we cover two small, two large, and one medium-sized configuration in order to reflect the federation of heterogeneous resource centres. The setups and their corresponding combinations of workload traces are listed in Table 5.1.

Table 5.1 *Setups f_1 to f_4 used for the analysis of the pooling approach, using the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup.*

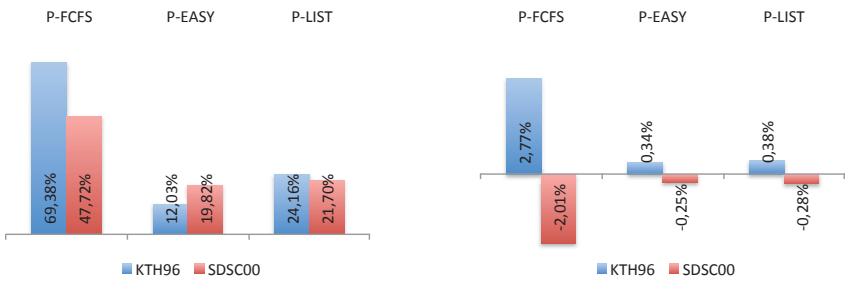
Setup	KTH96 11 mos.	CTC96 11 mos.	LANL96 24 mos.	SDSC00 11 mos.	SDSC00 24 mos.	SDSC03 24 mos.
f_1	✓			✓		
f_2			✓		✓	
f_3	✓	✓		✓		
f_4			✓		✓	✓

5.1.3 Evaluation

Based on the results from the reference data in Section 3.4, we evaluate the performance of the different federation setups in the following. The discussion of results is split into two-participant and three-participant sections.

Results for Two Participants

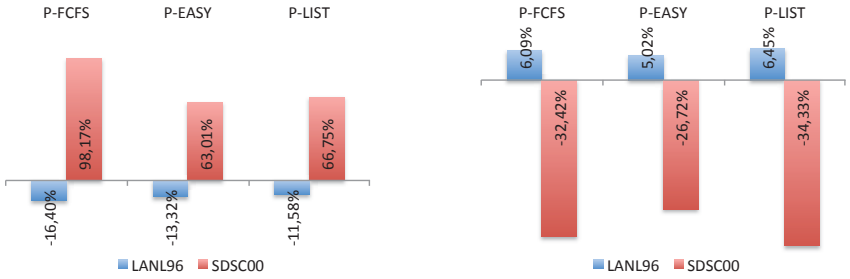
For the constellation of two equally-sized participants, represented by KTH96 and SDSC00 workloads, a significant improvement in AWRT with respect to the corresponding local setup can be observed, see Figure 5.1a. This is achieved even without affecting the utilisation or squashed area significantly, see Figure 5.1b.



(a) *AWRT.* **(b)** ΔSA_b.

Fig. 5.1 *Results for the two-participant setup f_1, as percental change wrt. to the non-federated case.*

A completely different behaviour can be observed when a small participant (SDSC00) cooperates with a large participant (LANL96), see Figure 5.2. Here, the small participant migrates a higher percentage of its jobs to the



(a) *AWRT.* **(b)** ΔSA_b.

Fig. 5.2 *Results for the two-participant setup f_2, as percental change wrt. to the non-federated case.*

large participant than vice versa. Obviously, the AWRT of the small participant improves strongly while the utilisation decreases. Concerning the large participant we observe that due to the large amount of jobs migrated from the small to the large resource centre the AWRT becomes, and the utilisation increases. However, the impact of migration is small which may result from the special usage policy of LANL96, see Section 3.2.3, so that small jobs from SDSC00 can be used to fill gaps.

The comparison of the job-sharing algorithms with the results for EASY as the best non-federated algorithm yields structurally similar results. However, for two small resource centres, the P-EASY and P-LIST approaches outperform EASY with respect to AWRT. Summarising, the following conclusions can be drawn from the experiments:

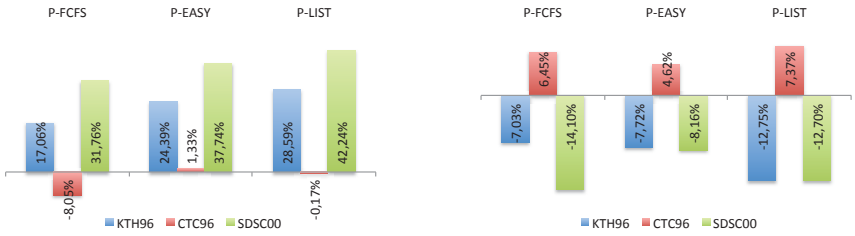
1. Adapting the local algorithms to job-sharing can lead to an overall AWRT improvement for resource centres with high utilisation and rather equal size.
2. Combining a resource centre with low utilisation and short AWRT with a more loaded one resembles a load balancing behaviour which may involve a significant deterioration concerning AWRT for the less loaded participant.
3. For disparate sized resources a significant higher percentage of job migrates from the small to the large participant than vice versa. While many jobs from the larger resource cannot be executed on the smaller resource due to oversized resource demands, jobs from the smaller may fill gaps in the schedule of the larger.

Results for Three Participants

For the setups with three collaborating participants the job-sharing algorithms again yield shorter AWRT values for all participating resource centres. In f_3

however, the utilisation at the largest resource centre (CTC96) increases in a load balancing behaviour, see Figure 5.3, but this does not affect the AWRT for P-EASY and P-LIST.

Since the sharing of jobs results in lower utilisation for two and higher utilisation for one participant while preserving or even improving the AWRT, we can identify an additional benefit of pooling: On the now less utilised system the providers have freed previously occupied resources, effectively allowing them to serve additional customers.



(a) *AWRT.* **(b)** ΔSA_b.

Fig. 5.3 *Results for the three-participant setup f_3, as percental change wrt. to the non-federated case.*

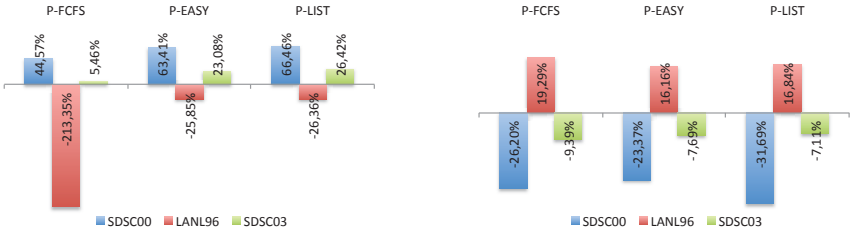
In f_4, which comprises two large participants and one small participant, the constellation leads to a strong deterioration in AWRT for P-FCFS for LANL96, as this participant takes most of the shared jobs. This effect is probably due to the characteristics of the LANL96 system and is intensified by jobs moving towards LANL96 from the small site, an effect that was also observed in the two-participant setup f_2. For the other algorithms the situation is alike while the AWRT deterioration is less strong.

Job Pool Size Variation during Interchange

In order to complement our results, we assessed the size of the job pool during the simulation. This provides additional insight to the behaviour and influence of the three algorithms regarding the usefulness of the pool. Large pool sizes are more likely to indicate that the workload is not handled well, thus leaving many jobs in the lower prioritised global queue.

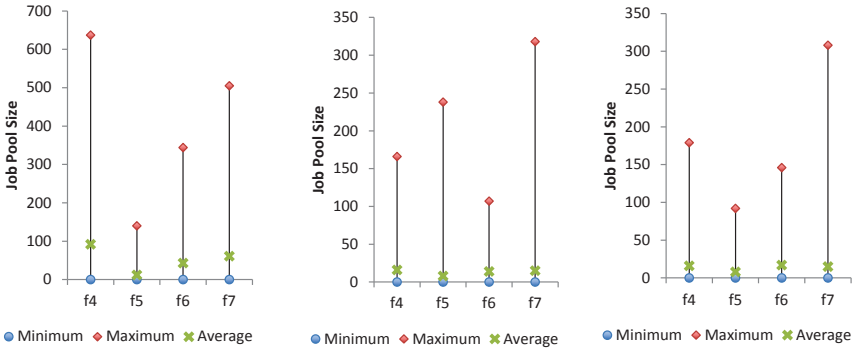
The size of the pool was determined every time a job is added to or removed from it for every participant, and then averaged.

Figure 5.5 gives an overview of minimal, maximal and average pool sizes for the discussed setups. It shows that P-FCFS has significantly larger pool sizes, both for maximum and average values, than the two other variants. For P-LIST, this result was to be expected: As the algorithm iterates over the global



(a) *AWRT.* **(b)** ΔSA_b.

Fig. 5.4 *Results for the three-participant setup f_4, as percental change wrt. to the non-federated case.*



(a) *P-FCFS.* **(b)** *P-EASY.* **(c)** *P-LIST.*

Fig. 5.5 *Pool sizes for setups f_1, f_2, f_3 and f_4.*

queue repeatedly unless local resources are fully occupied, the likelihood for keeping the pool size small is obviously higher. For P-EASY, the results are varying: While for the smaller setup the maximum pool sizes are considerably smaller than for the other algorithms, the mixed setup shows the highest maximum pool size. On the average, however, both backfilling algorithms clearly outperform P-FCFS; it is obvious that backfilling opportunities highly influence the size of the pool.

We generally observe that the smaller the setups in terms of participants' sizes are, the smaller the pool sizes get for the backfilling algorithms. Again, this stems from the likelihood of backfilling opportunities of the workloads: Smaller systems feature more jobs with low resource requirements (as not so much space is available anyway), thus providing more potential backfilling candidates, while larger systems tend to have bigger jobs in the workload which are hard or impossible to backfill. This is also reflected by the results

for both the two- and three-participant setups and allows us to conclude that in comparison with the total number of jobs processed the pool is not overcrowded: As the average values were comparably small, jobs did not reside in the pool for a long time and for practical use scenarios, job starvation does not seem to occur.

5.1.4 Related Work

Regarding the indirect communication model in the centralised case, a first study by Blumofe and Park (1994) shows potential advantages that result from work stealing in a network of workstations. However, they assume a two-layer approach with "micro"- and "macro"-scheduling and focus on FIFO and LIFO strategies on the macro level. Also, their evaluation does not consider real workloads, but uses four applications (two of them real ones) and assess speedup and slowdown. Ernemann et al (2002b) identify improvements for the average weighted response time objective assuming hierarchical centralised scheduling structures in multi-site computing.

Among the large variety of studies on load balancing Lu et al (2006) developed algorithms for distributed load balancing in a Grid environment assuming an additional communication delay. Their approach requires that the whole local installation is controlled by a central scheduler. Also, they built the evaluation solely upon statistical workload models.

5.2 On the Influence of Active Decision Making

Although the passive interaction model discussed in Section 5.1 provides a simple means to setup an ad-hoc federation and, for that use case, performs well, it is not suitable for larger setups: With more resource centres being added to the federation, the global job pool will undoubtedly emerge as the bottleneck of the whole system, due to the increasing number of requests, the growing pool size, and the need for every participant to traverse the whole queue on every request. These scalability issues are a necessary byproduct of the model itself, and cannot be easily remedied.

For larger-scale federations it is therefore reasonable to look at the active interaction model instead. WLCG is a good example here: With almost 150 participants and about 1,000,000 jobs per day, it is certainly not suitable for the central exchange platform model discussed above. Instead, it uses a centralised scheduler called *Workload Management System* (Marco et al, 2009), which distributes the incoming workload among the available resources along a set of policies. In this operational model, the coordinator makes all scheduling decisions and all workers have to execute them.

However, under policies other than those imposed by CERN for the Large Hadron Collider (LHC) data processing, this amount of influence is not necessarily desirable from the providers' point of view: In a system with a central component for decision making, each participant on the second hierarchy level must cease a certain amount of autonomy to let the first hierarchy level make reasonable schedules for the whole federation. In addition, the decision making on the first level may require the second-level systems to disclose information about their current state; this information is usually not publicly available, which leads to a conflicting situation: On the one hand, being part of a federated capacity planning environment may be beneficial in terms of better service to the own user communities and system utilisation; on the other hand, fully ceasing control and providing system data may not be compatible with the participants' policies.

To better understand the influence of this tension on federated SRM for the centralised case, we explore two scenarios: A full-control case, where the global SRM system conducts all planning, and attached resource centres have to submit to its decisions; and a no-control case where the local systems retain full autonomy.

Surprisingly, we find that less autonomy on the local (and thus more control on the federation) level does not necessarily lead to better results regarding response time of the participating centres. On the contrary, the results indicate that—within the limits of the configuration evaluated here—the no-control case yields good results even though the centralised scheduler only plays an consultatory role. Clearly, the participation in the federation is beneficial even for the largest centres, improving AWRT by approximately 7 % at the very least. With the shifts in utilisation, this shows that loose federa-

tions of resource providers have ample room for improvement with respect to utilisation of their systems without impairing user satisfaction.

5.2.1 An Autonomy-centric Viewpoint on the Federation

In the sense of the architectural considerations made in Section 4.3, we assume a *centralised active* interaction model. For the different levels of autonomy, we use the classification proposed by Gehring and Preiss (1999), who formulate their model along the *level of information exchange* and propose two system classes.

Scenario 1: Scheduling with Full Control

With full control, the centralised scheduler has access to the complete workload of each participant, and the participants' LRMs are not allowed to decline jobs from the centralised scheduler. In the real world, enterprises using federated LRM installations within a single administrative domain often run such setups.

Due to the added knowledge gained from the cooperative behaviour of the participants, the centralised scheduler can incorporate additional data into its dispatching decisions. In order to leverage this, it uses a *BestFit* strategy, see Jackson et al (2001), which analyses the current queue and schedule of each participant and schedules the job under decision to the system where it would leave the least[1] amount of resources unoccupied.

In the full-control case, no queue is used by the centralised scheduler: Since every job is dispatched directly after its arrival, and keeping a backlog of waiting jobs is not necessary.

Scenario 2: Scheduling with no Control

Here, the centralised scheduler does not have any knowledge about locally submitted jobs. In addition, it does not know about the current utilisation of the participants and cannot query this information. The participants' LRMs are free to delegate workload to the centralised scheduler and accept or decline offers from the centralised scheduler. Loose, project-based federations in the research space often follow this model.

Since nothing is known about the resource centre and the workload[2], the algorithmic capabilities of the centralised scheduler are rather limited. A simple approach to the problem would be to use a Largest Resource First (LRF) strategy: Because larger participants are more likely to have non-occupied

[1] Naturally, this decision might not be optimal.

[2] Except for the maximum number of processors for each participant, which is usually public.

resources than smaller ones, backfilling on each participant using workload coming from the centralised scheduler is more likely to happen.

Here, we use the average wait time as criterion for the participant selection. Due to the restricted information policy in this scenario, we introduce a variant that limits itself to the workload channeled through the centralised scheduler, see Equation 5.1.

$$\mathrm{AWWT}^{d}_{k,l} = \frac{\sum\limits_{j \in \tau_k \cap \pi_l} \mathrm{SA}_j \cdot (C_j(S) - p_j - r_j)}{\sum\limits_{j \in \tau_k \cap \pi_l} \mathrm{SA}_j} \tag{5.1}$$

As for AWRT, the wait time is weighted with the resource consumption of each job. By keeping the list of participants sorted along the Average Weighted Wait Time (AWWT) of delegated jobs, in ascending order, a Smallest Expected Response First (SERF) strategy is realised.

For both approaches, the centralised scheduler needs to keep an additional local queue to buffer jobs that have been delegated by one of the participants, but cannot be scheduled directly. In contrast to the previous section, this queue cannot be accessed by the participants, because the centralised scheduler also follows a strict information policy.

5.2.2 Experimental Setup

For the evaluation of the centralised setups, the metrics U and AWRT as defined in Section 3.3 are used as objectives. The setups and their corresponding combinations of workload traces are listed in Table 5.2. Note that, because of the unusual configuration and behaviour of LANL96, we replaced this workload by SDSC05.

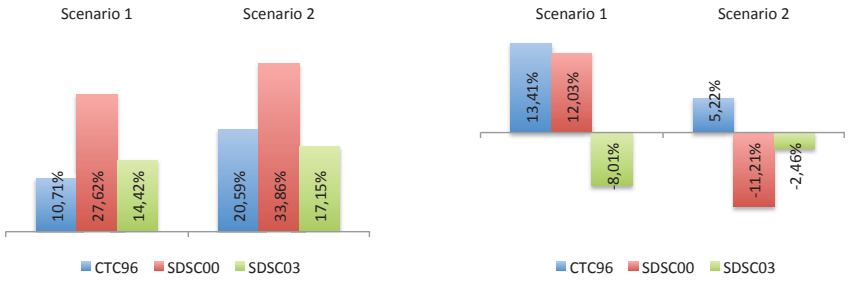
Table 5.2 *Setups f_5 to f_7 used for the analysis of the autonomy approach, using the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup.*

Setup	KTH96 11 mos.	CTC96 11 mos.	SDSC00 11 mos.	SDSC03 11 mos.	SDSC05 11 mos.
f_5		✓	✓	✓	
f_6	✓	✓		✓	✓
f_7	✓	✓	✓	✓	✓

As the heuristic in the LRM systems of each participant, we use EASY for all setups. This is because the analysis from the reference results in Section 3.4 indicates that it usually outperforms FCFS, and LIST as a local heuristic is not very commonly used.

5.2.3 Evaluation

Following, the afore described setups are evaluated on their performance regarding AWRT and U as described in Section 3.3 with respect to the non-federated case, see Section 3.4. To be able to directly compare the impact of full-control and no-control scheduling scenarios, three setups are being assessed for both cases regarding the effect of federation in centralised architectures.



(a) *AWRT.* **(b)** *U.*

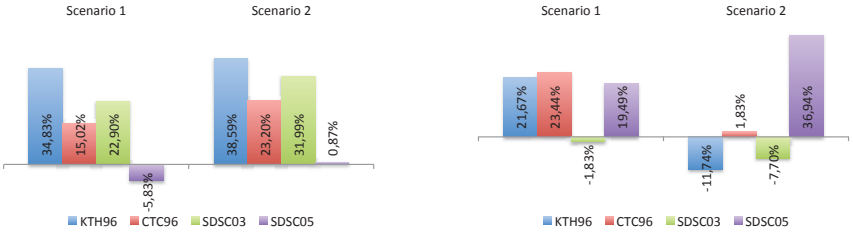
Fig. 5.6 *Results for the three-participant setup f_5 in scenarios 1 and 2, as percental change wrt. to the non-federated case.*

In the smallest setup f_5, improvements in AWRT can be realised for all three participants in both scenarios, even if it clearly shows that the smallest centre, SDSC00 benefits most of the federation, see Figure 5.6a. This effect can also be seen for the systems' utilisation, see Figure 5.6b: While the change for the larger systems CTC96 and SDSC03 is almost negligible, the smaller centre shows a clear deviation for both scenarios, albeit in different directions.

For setup f_6, similar observations can be made. Again, the AWRT increases for all participants in both scenarios, see Figure 5.7a, even though the benefit for the largest centre, SDSC05, is not significant in the full-control scenario. This does not, however, hold for U, see Figure 5.7b, where SDSC03, a large centre with more than 1,000 processors, has less utilisation. For the smaller participants, again considerable deviations can be observed.

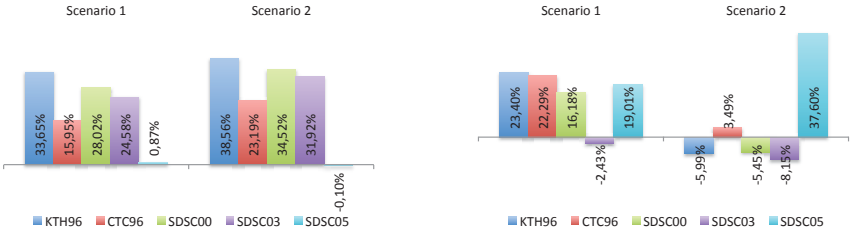
The largest setup in this analysis, f_7, shows the same behaviour, see Figure 5.8. AWRT improvements can be taken by all participants where again the largest centre has the least benefit, and utilisation also behaves similarly to the previous cases.

Surprisingly, it seems that—given this specific experimental setup—more knowledge on the federation level and less autonomy on the provider level does not necessarily yield better scheduling results. In fact, it shows that for



(a) *AWRT.* **(b)** *U.*

Fig. 5.7 *Results for the four-participant setup f_6 in scenarios 1 and 2, as percental change wrt. to the non-federated case.*



(a) *AWRT.* **(b)** *U.*

Fig. 5.8 *Results for the five-participant setup f_7 in scenarios 1 and 2, as percental change wrt. to the non-federated case.*

all three setups, the no-control scenario leads to better results for AWRT in all cases.

The systems' utilisation, U, cannot be interpreted as direct gain or loss for the centres, but rather as an indicator for the shift of workload between them. Here it shows that the full-control scenario leads to a better utilisation of the smaller centres while moving away workload from the mid-size systems (such as CTC96 and SDSC03, depending on the setup). In the no-control case, the situation is opposite: Utilisation generally moves towards the largest participant in the setup, see Figure 5.7b and Figure 5.8b, or at least moves away from the smallest centres, see Figure 5.6b.

5.2.4 Related Work

Regarding the algorithmic setup of the centralised system, Ernemann et al (2002a) show early results on similar strategies (namely *BestFit*), but limit themselves to the CTC96 workload and a synthetic trace. In addition, their focus lies on cross-domain execution, which is not discussed here. A rather

general overview on the topic is given by Hamscher et al (2000): Here, a hierarchical setup is studied (among others), using a utilisation-based distribution strategy for the centralised scheduler.

On the theoretical side, Tchernykh et al (2006) introduce a hierarchical setup with a centralised scheduler and local resource centres. In this context, allocation strategies for bound analysis are used both on the federation and the local level, showing that certain approximation guarantees can be reached.

5.3 On the Application of Hierarchical Capacity Planning†

As we have seen, the centralised active approach works well even if the autonomy of the participants in the federation is high: The participants were free to decide on whether to accept or refuse the decisions made by a higher authority in a tiered environment, and the distribution of workload still produced good results. With this observation, we can extend our model to the hierarchical case: The decision making process is still centrally managed (that is, a coordinator starts and directs the process), but each worker in a given tier can further delegate workload autonomously.

This approach of continually asking the "next neighbour" to perform a task is very common in modern P2P systems (Balakrishnan et al, 2003). There, especially in the field of capacity management, a request is inserted by a user at one peer and propagates through the network until eventually another peer decides to handle it, see for example Xu et al (2006). With the trend in research towards P2P-based capacity management architectures, see Foster and Iamnitchi (2003), it is therefore reasonable to explore the transferability of solutions from the P2P domain to SRM in federated DCI environments. This is all the more interesting, because modern P2P systems are able to deliver functionality on a very large scale: Even small peer networks often cater hundreds of thousands of participants at a time, spanning large geographical areas and logical network topologies.

We investigate in an experimental study whether the adaptation of P2P-based capacity management is feasible by adapting a stream allocation strategy for video sharing in peer-based networks to the dispatching of DCI. In this relatively unexplored area of distributed capacity planning, we evaluate the performance of this strategy using three exemplary scenarios over a total time period of eleven months and considering different participant configurations regarding the number of available resources and submitted jobs.

The results show that our algorithm outperforms the well-known EASY heuristic regarding AWRT and U results, and proves yet again that, also for the hierarchical case, the participation in a federated DCI environment can be considered beneficial for all participants. Admittedly, the construction of virtual schedules is compute-resource intensive, requiring significantly more computation time[3] than the strategies discussed in the previous chapters; however, performance is still acceptable insofar that real-time usage is not limited.

† Parts of this section have been presented at the 39th International Conference on Parallel Processing in San Diego, CA on September 13, 2010. The corresponding publication, see Grimme et al (2010), is available at `http://ieeexplore.ieee.org`.

[3] Simulation length actually increased by factor 10, from a decision time of 0.3 ms/job to 3.1 ms/job.

5.3.1 Shaking as a means of Capacity Planning

As the starting point for the adaptation experiment proposed above, we use a strategy developed by Cai et al (2007), which addresses the problem of service scheduling in peer-to-peer video provisioning. This method—called *Shaking* in the original work—dynamically balances the workload throughout a network consisting of independent peers to optimise video availability for requesting clients by reorganising local queues recursively.

Original Algorithm

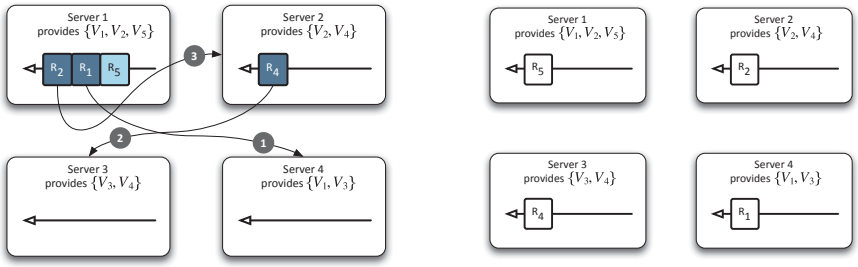
More specifically, a client submits a request to a streaming server which is selected regarding individual[4] objectives. The request is added to the end of the server's queue and has to wait for the completion of preceding requests before being processed. If no preceding requests exist, there is nothing to do for the shaking algorithm. However, in case a request cannot be handled immediately, the client tries to migrate preceding requests to distant servers in order to reduce the waiting time of its own request. To this end, a *closure set* consisting of alternative provider candidates for the queued requests is constructed: Starting with a set of servers for the initial request, the client analyses the queue of each element in the set and initiates a search in order to determine a suitable candidate server set for the pending requests. This is repeated until either a search threshold is met[5] or only idle servers are found.

The union of these sets yields the constant closure set, and on this closure set, the algorithm attempts to migrate pending requests to other servers with the goal to minimise the waiting time for the newly enqueued request. Generally spoken, all predecessors of a given request are moved—or *shaken*—to servers that are able to provide the same type of service without additional waiting time. In order to achieve this, those requests from the receiving servers that precede the now migrated requests have to be shaken again. This leads to a (recursive) chain of migrations with a specific order of execution that determines the benefit of the procedure. In order to find the best migration plan, multiple shaking orders—restricted by the fixed size of the closure set—are evaluated by the client and the best plan is selected for execution.

In Figure 5.9 we give an example of the situation before and after the application of the algorithm's working principle. Request R_5 for video V_5 is added to the queue of server 1. After constructing the closure set containing server 1 to 4, Shaking tries to remove the preceding requests R_1 and R_2 from the queue of server 1. R_1 can be easily moved to server 4 which also serves V_1, see (1), while R_2 cannot be moved to server 2, see (3), until R_4 in server 2 has been moved to server 3, see (2). The so determined shaking plan results in a benefit for R_5 at server 1 as well as in a faster execution of R_4 at server 3.

[4] Such as bandwidth, latency, or content-specific properties.

[5] It is noteworthy that, if no threshold is given, the closure set may contain all available servers in the network.



(a) *Before the Shaking process.* **(b)** *After the Shaking process.*

Fig. 5.9 *Example of the application of the original algorithm applied for the optimisation of video provision in a peer-to-peer network.*

5.3.2 Adapted Algorithm

For our novel algorithm, Shaking-G, we retain two key features from the original algorithm: its dynamic scheduling behaviour and the minimisation of wait time. The process is divided in two main phases. The first phase, called *starting phase* prepares the allocation decision. The second phase, called *shaking phase*, then reallocates the workload based on the previously outlined plan.

In contrast to the P2P video provisioning problem tackled by the original algorithm, Shaking-G considers computational jobs as entities to be shaken between distant sites. A participant can "serve" a job if it has enough free resources to start it immediately. Each participant provides a broker component which accepts user submissions and encapsulates the dispatching strategy for workload distribution; as such, the here described shaking procedure is transparent for the client, see Figure 5.10. This setup corresponds to the hierarchical centralised active interaction model described in Section 4.3.1.

The *starting phase* is initiated at the moment a user submits a job j to its local resource centre k. where it is added to the resource manager's queue. In case k provides a sufficient number of free resources to start j, it is transferred to the underlying LRM and executed immediately. Otherwise, the broker initiates a search for remote participants that meet the processing requirements of j in terms of provided processors. The returned candidates are queried sequentially whether they are able to start the job immediately. A positive answer leads to a migration of j to the respective resource centre. In this case the application of the shaking mechanism is not necessary.

If none of the candidates is able to provide the necessary resources, the job remains in the local broker's queue and the *shaking phase* is started to migrate preceding jobs of j from the local queue until the waiting time of j is reduced. This is equivalent to reducing the Estimated Completion Time (ECT) of a job, which is based on the workloads' runtime estimates $\bar{p}_j$, see Section 3.2.3,

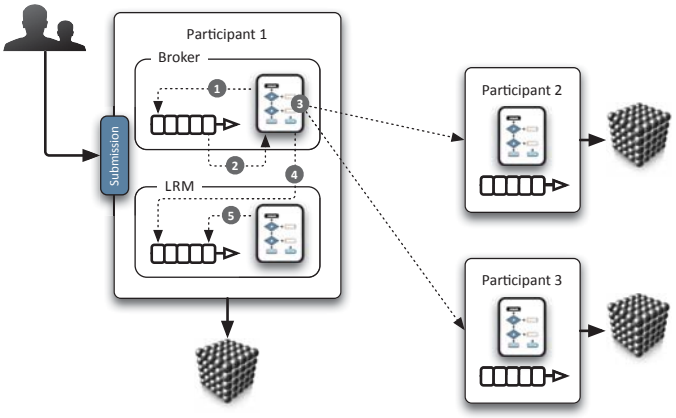


Fig. 5.10 *Schematic depiction of the creation of a shaking plan in the recursive Shaking-G algorithm. Users induce their workload to their local resource centre through their accustomed submission interface, and the* SRM *system enqueues it with the broker component. The architecture is an extended version of the centralised active case in Figure 4.3.*

available at release time. Based on a resource centre's current occupation of resources the scheduler can build a virtual schedule for the currently waiting jobs considering their runtime estimations. The completion time $C_j(\bar{S}_k)$ of job j in this predicted schedule $\bar{S}_k$ determines the ECT value of the job at participant k.

For a given job l in the queue, the algorithm creates a set of candidates which are able to fulfil the job's resource demand. For the determined candidates a set of shaking plans for job l is constructed: Each candidate is queried whether a faster execution of job l is possible. A positive answer completes the shaking plan and annotates it with the new ECT for the job. A negative answer results in a recursive application of the shaking phase for the considered candidate. The depth of recursion is limited by a threshold in order to restrict the runtime of the algorithm. In order to avoid the repeated consideration of participants during the construction of shaking plans, a shaken candidate is added to a shaking set. This set is excluded from further shaking phases. Finally, the shaking plans that offer the most beneficial reduction of waiting time for job j are executed.

In Figure 5.10 the creation of a shaking plan is shown exemplarily. After a new job has been inserted into the queue of participant 1, the broker then checks whether immediate execution is possible (1). If other jobs are preceding the submitted one in the broker's queue (2), Shaking-G attempts to shake the preceding jobs in order to improve the starting time for the new job. To this end, the broker of participant 1 attempts to delegate them to other participants in the federation (3). Recursively, the jobs from participant 1's queue are

offered to neighbouring participants 2 and 3 which then, hypothetically, try to improve the starting time of those jobs by shaking their own queues. During the whole procedure the queue modifications and job exchange activities are recorded to specific shaking plans for each traversal path in the recursion. The constructed shaking plans are compared regarding the ECT of the initial job and the most beneficial plan is selected to be executed. Note that only beneficial job exchanges lead to a beneficial recursion path and shaking plans on each shaking level. As soon as enough resources are available, the broker hands the job over to the LRM system (4), where the local strategy eventually schedules the job for execution (5).

In detail, the modified algorithm proceeds as follows: As soon as job j is enqueued (that is, not executed directly) at participant k the shaking phase can begin. This phase starts by examining all jobs that are queued before j. These jobs need to be rescheduled to other participants, eventually allowing a sooner execution of j. To this end, a closure set is created with jobs preceding j and participant k is added to the "shaking pool". The algorithm then selects a job l from the closure set in an FCFS fashion. Next, a new search for suitable servers[6] begins, and the results are queried for the ECT that job l would have if it was to be processed there. The returned ECT values from each participant are compared to the locally calculated ECT value. The calculation of ECT is done as follows:

1. Obtain the current scheduling state of the participant (local queue and schedule, that is);
2. Create a shadow copy of the actual schedule and allocate each job in the queue using FCFS, and repeat this process until the job that caused the shaking is under review;
3. Look at the makespan in the expected schedule and return it as ECT.

If at least one suitable server has a better ECT than the local system, an allocation to the system with smallest ECT is created. If no suitable server can provide a better ECT, a new shaking set is created with only those systems that are not already in the shaking pool.

With respect to the original shaking strategy, we introduced two important changes into the afore described algorithm's variant: In addition to the migration criterion of available and currently free resources, the ECT metric is used in order to evaluate whether the exchange of jobs is profitable. Additionally, we integrated the steering of the shaking process—formerly initiated and centrally steered by the submitting client—into the participants' brokers. To this end, we replaced the direct, client-side creation of shaking plans with a recursive, server-side plan construction. Along with that, the global view on the queues of distant sites was restricted and replaced by an interface for controlled interaction, following the limited information doctrine of this work.

[6] That is, resource centres that meet the processing requirements of l.

5.3.3 Experimental Setup

For the evaluation of the described scenario, a number of workloads and metrics have been used. On the metrics side, AWRT and U have been used, as defined in Section 3.3. For input data, the traces[7] described in Section 3.2.3 were used, namely KTH96, CTC96, SDSC00, and SDSC03. For this algorithm, we omitted SDSC05 because of the long simulation times. The setups and their corresponding combinations of workload traces are listed in Table 5.3.

Table 5.3 *Setups f_8 to f_{10} used for the analysis of the shaking approach, using the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup.*

Setup	KTH96 11 mos.	CTC96 11 mos.	SDSC00 11 mos.	SDSC03 11 mos.
f_8		✓	✓	
f_9	✓	✓	✓	
f_{10}	✓	✓	✓	✓

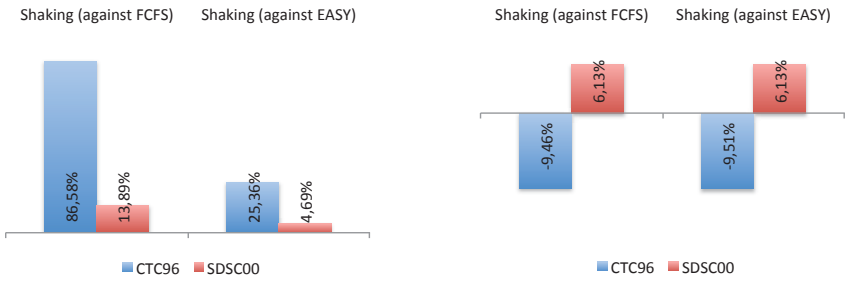
5.3.4 Evaluation

In the following, the changes of the two metrics compared to the reference results as described in Section 3.4 are discussed. It shows that it is possible to increase the benefit regarding AWRT for all resource centres in the examined cases, which results in significantly shorter response times for the corresponding users. Along with this, slight changes in the total utilisation for all participants can be observed.

Figure 5.11 reveals that the smaller resource centre benefits from the size of the larger one: The achieved AWRT is approximately 25 % shorter compared to EASY. Although AWRT also improves at the larger site, U drops by 10 %: Shaking-G obviously leverages previously unused capacities to reduce the high load on SDSC00. However, at the same time the SRM strategy creates a job allocation that does not penalise CTC96 users.

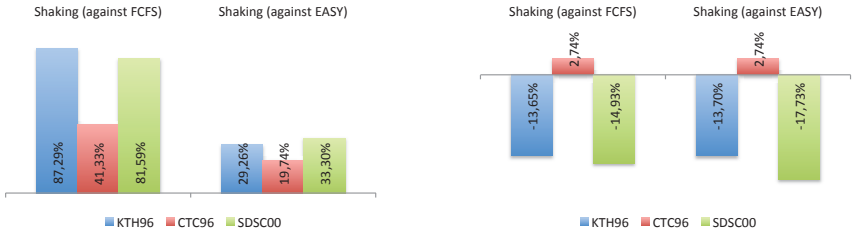
It is noteworthy that, for the analysis at hand, no threshold for the maximum number of shakes was introduced. Naturally, Shaking-G runs at risk to perform an infinite number of shakes under this parametrisation. In practice, however, jobs are only shaken once on average, and it is rarely the case that jobs are shaken more often. For the here examined real workloads and scenarios the highest amount of migrations from one participant to another was 35. As previously stated, this indicates that starvation does occur, but even if it would, this can be countered by lowering the maximum number of allowed shakes.

[7] Note that only traces exposing runtime estimates were used, as these are required for the calculation of ECT, and thus for the proper functioning of the algorithm.



(a) *AWRT.* **(b)** *U.*

Fig. 5.11 *Results for the two-participant setup f_8, as percental change wrt. to the non-federated case.*

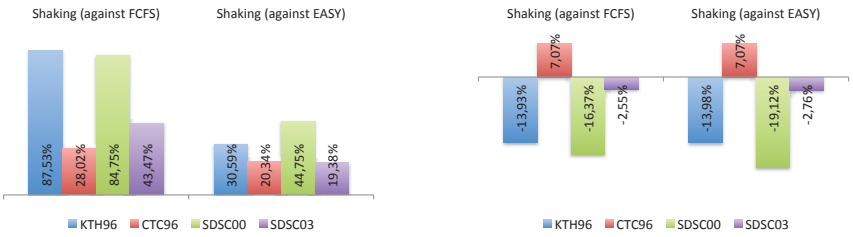


(a) *AWRT.* **(b)** *U.*

Fig. 5.12 *Results for the three-participant setup f_9, as percental change wrt. to the non-federated case.*

Qualitatively similar results are obtained for f_9, see Figure 5.12, where two smaller participants significantly benefit from the interaction with one larger resource centre. Although the utilisation increases at CTC96, the AWRT values can be improved. Load balancing is consequently not the only factor that characterises the benefit of Shaking-G: Due to the different participants, remote resources can be involved in the scheduling process and used for a more efficient job allocation.

Finally, we review the results for f_{10} where a large resource centre joins the federation, see Figure 5.13. Especially for SDSC00, even higher improvements are obtained in comparison to f_9, and a clear shift of the load is observable: CTC96 increases its utilisation while the load for KTH96 and SDSC00 is reduced. Still, a significantly shorter AWRT is achieved for all participants, and even the largest participant is able to benefit from the cooperation.



(a) *AWRT.* **(b)** *U.*

Fig. 5.13 *Results for the four-participant setup f_{10}, as percental change wrt. to the non-federated case.*

5.3.5 Related Work

Lo et al (2005) introduce a system called "Cluster Computing on the Fly", which leverages the benefits of day/night cycles (see Section 3.4.2) and allows for Point-of-Presence scheduling; however, no specialised routing of workload is conducted. Dou et al (2003) discuss a similar approach to global Grid scheduling, but base their scheduling decisions on each participating node's load factor, but not on response time or utilisation. Subramani et al (2002) propose a distributed SRM system that interacts with all participants on both the local and distributed level. Although they consider the inaccuracy of runtime estimations and also communication delays, their evaluation is based on partially rescaled workloads and limited to a subset of 5,000 jobs only. While England and Weissman (2005) assume distributed structures and analyse costs and benefits of load sharing, their work is based on statistical workload models, and they review the average slowdown objective only. Ranjan et al (2008) review existing scheduling schemes on real workload traces and propose a no-central-component concept for sharing cluster resources by allowing direct negotiation between arbitrary partners. They premise a shared federation directory that manages the local resources and collects quote and cost information used for economic scheduling. This assumption, however, is problematic in most real-world Grid scenarios, since the systems' current state, e.g. current utilisation, are—as discussed before—usually treated confidential.

6

Selected Methods for Distributed Architectures

A committee can make a decision that is dumber
than any of its members.
—David Coblitz

BUILDING UPON a centralised architecture paradigm regarding SRM admittedly is the predominant choice in large production DCIs, with the EGEE environment (Gagliardi et al, 2005), as one of the most widely deployed. Other—mostly emerging—efforts, such as the D-Grid Initiative[1], however, have chosen to build smaller, more community-tailored DCIs, see Freitag and Wieder (2011), that adhere to the distributed architecture paradigm. But although this approach enables flexible collaboration with participants attaching to and detaching from the federation easily when necessary, the decoupling naturally comes at a price: The allocation of workload on the DCI level becomes more complicated because the SRM heuristics are lacking a global view, see Section 4.3. All the more, each participant will want to protect his investment, striving to deliver the best possible performance to the own user community.

To see whether the distributed interaction model behaves equally well under similar conditions as the centralised active one we first explore the potential of workload migration and exchange without global control, but assuming similar restrictions regarding autonomy and information disclosure as discussed in Section 5.2. Herefor, we introduce a new heuristic and evaluate its performance in Section 6.1.

Next, we review the performance of more dynamic, situation-aware SRM system: Besides the obvious requirement of adaptation whenever the local usage pattern changes, a proper federated SRM system should also be able to cope with tension between local and global interests. We discuss this aspect in Section 6.2, and introduce a self-adapting system for SRM using a fuzzy controller architecture and evolutionary computation.

Last, we approach the problem in the light of the latest developments in the Cloud computing domain and investigate whether a change from a workload-centric allocation paradigm towards a resource-centric leasing paradigm is reasonable, and develop a new SRM approach in Section 6.3.

[1] `http://www.d-grid.de`.

6.1 On the Benefits of Workload Migration†

Since our approach of passive and active workload exchange in the centralised case turned out to be successful, this will be our starting point for the design of algorithms for distributed SRM. However, we now target a completely different architectural paradigm: As discussed in Section 4.3.2, we allow resource centres to interact directly, and we have no single instance in control of the whole system.

When taking a closer look at the interaction model depicted in Figure 4.5, it turns out that there are similarities. The Federated SRM component's behavior in the centralised active approach, namely pushing workload from the coordinator to a worker can be reinterpreted for the distributed case as one federated SRM system *offering workload* to another. Likewise, the pulling of jobs from the global job pool to a participant can be transformed to one federated SRM *requesting workload* from another.

Although the interaction model is similar, the nature of the federation—because of the inherent autonomy of its participants—stays different: The resource centres do not consider a potential performance gain for all users in the federation, but rather for the local community only; the same holds for the utilisation of the systems. Against this background, it is necessary to have a closer look on the question whether DCIs comprising participants with mostly egoistic motives can still benefit from a collaboration.

In order to find first directions for an answer, we make an analysis of the behaviour under this assumption in the following. By leveraging the results from the central case in Chapter 5, especially regarding the workload exchange process, we introduce two new strategies: one *request-centric* (RQF) that pulls workload from other participants, and one *offer-centric* (ACF), which pushes workload to other participants. Regarding the disclosure of information, we let the former gain insight into the participating resource centres' current load in order to "steal" workload that matches the local utilisation, and leave the latter with no information about the other participant other than its size, letting it offer non-fitting workload to the others.

The evaluation of both algorithms' AWRT and ΔSA characteristics shows that each approach yields a significantly better user experience than non-federated SRM while freeing up resources for more workload, thus providing overall more efficient workload distribution. Surprisingly, the RQF variant has no advantage over ACF, although having access to more information and doing more delegations; this validates the our results from the centralised setups described in Section 5.2 and shows the strong potential in offer-centric systems.

† Parts of this section have been presented at the 8th IEEE International Symposium on Cluster Computing and the Grid in Lyon, France on May 20, 2008. The corresponding publication, see Grimme et al (2008a), is available at `http://ieeexplore.ieee.org`.

6.1.1 One-to-one Collaboration between SRM Systems

With respect to both options described above, we apply two basic strategies on the federated SRM level: one *pull-based*, where one broker **requests** workload from another, and one *push-based*, where one broker **offers** workload to another.

The *Request Fitting* (RQF) strategy essentially monitors the local queue and, whenever the first job cannot be executed immediately, potential backfilling candidates[2] are requested from the other participants. Every candidate job that fulfils either of the two EASY backfilling criteria, see Section 3.4.1, is then delegated to the requesting system and added to the end of its local queue. The *Accept Fitting* (ACF) strategy in contrast monitors all incoming workload and, for any job that cannot be executed immediately, offers this job to the other participants. The receiver of this offer accepts it if the offered job could be executed immediately, accepts the delegation in the positive case and finally adds the job to the end of its local queue.

The main difference, although using very similar reasoning regarding choice of delegations, lies in the initiating participant: For RQF, the delegation source is passive; for ACF, the delegation target is passive.

6.1.2 Experimental Setup

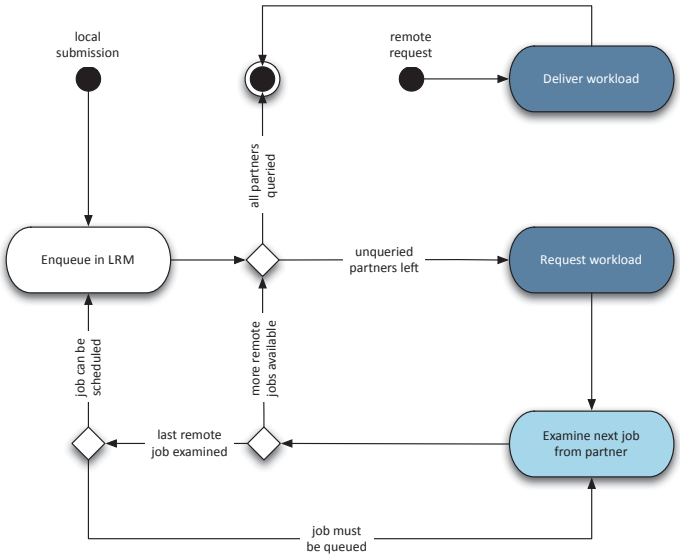
For the evaluation of the two migration heuristics, we use AWRT and the change in SA as metrics, see Section 3.3. In addition, we measure the amount of migration between the different participants. To this end, we introduce migration matrices M_n that show the ratio of job interchange, in percent, with respect to the original submission system, see Equation 6.1. Rows denote the participant where jobs have been originally submitted to while columns specify the participant the job was actual executed at.

Moreover, in order to measure the actual amount of workload that has been migrated between resource centres, we introduce the matrix M_{SA} similar[3] to M_n that relates the overall SA per workload with the migrated portions. Exemplarily, the Squashed Area for the set π_{kk} is computed as $\mathrm{SA}_{\pi_{kk}} = \sum_{j \in \pi_{kk}} p_j \cdot m_j$.

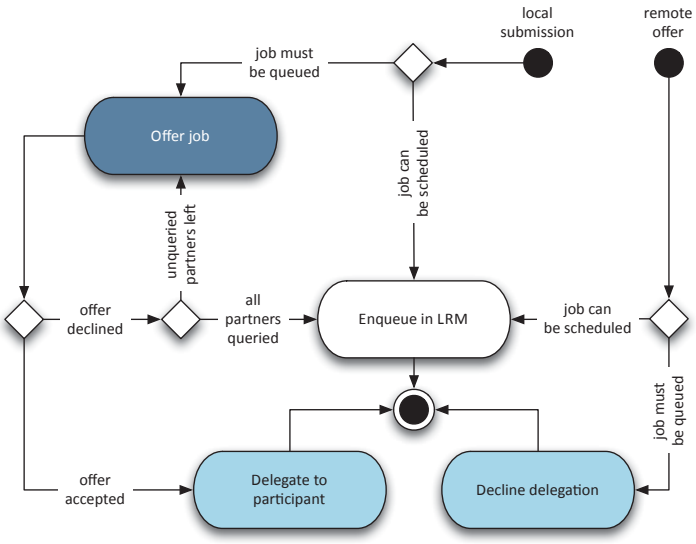
$$M_n = \begin{bmatrix} \frac{|\pi_{11}|}{|\tau_1|} & \cdots & \frac{|\pi_{1k}|}{|\tau_1|} \\ \vdots & \ddots & \vdots \\ \frac{|\pi_{l1}|}{|\tau_l|} & \cdots & \frac{|\pi_{lk}|}{|\tau_l|} \end{bmatrix}, \quad \{l,k\} \in [1 \cdots |\mathcal{K}|] \text{ and } \sum_{l=1}^{|\mathcal{K}|} \frac{\pi_{lk}}{\tau_l} = 1 \qquad (6.1)$$

[2] Naturally, these are pre-filtered with respect to their minimum resource requirements and the available resources at the requesting side.

[3] Effectively, the π from Equation 6.1 is replaced with the corresponding SA value.



(a) *RQF.*



(b) *ACF.*

Fig. 6.1 *Brokering process of the migration strategies applied in one-to-one collaborations. Both process renderings feature two starting points: One handles the submission of workload by the local user community, while the other handles incoming remote requests from other* SRM *systems.*

Finally, we record the number of migrations until final execution in order to assess the danger of starvation.

Table 6.1 *Setups f_{11} to f_{13} used for the analysis of the collaboration approach, using the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup.*

Setup	KTH96 11 mos.	CTC96 11 mos.	SDSC00 11 mos.	SDSC03 11 mos.	SDSC05 11 mos.
f_{11}	✓	✓	✓		
f_{12}	✓			✓	✓
f_{13}	✓	✓	✓	✓	✓

As input data, we use the workload traces from KTH96, CTC96, SDSC00, SDSC03, and SDSC05. The combinations of these for the different federation setups are listed in Table 6.1. As local scheduling strategy, we apply EASY, see Section 3.4.1.

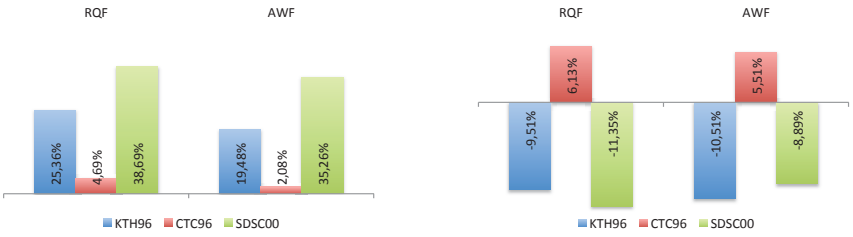
6.1.3 Evaluation

We now evaluate the performance of the different federation setups in the following, compared to the reference data as described in Section 3.4.

Performance Characteristics

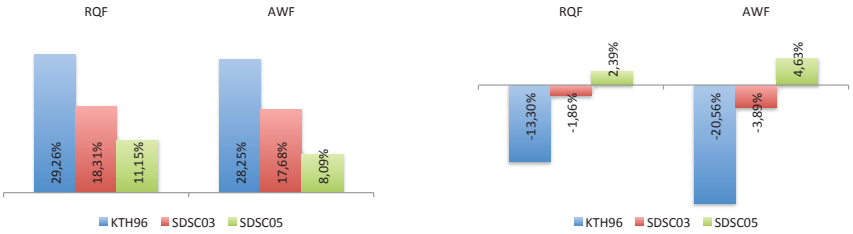
Setup f_{11}, comprising two small and one medium resource centres, shows that the users of all three sites benefit from a significantly decreased AWRT value, see Figure 6.2a. Along with this, the utilisation of the smaller sites is reduced to the account of the larger site, see Figure 6.2b. For the other three-participant setup, f_{12}, the results are similar: AWRT improvements can be reached for all sites, while workload tends to move towards the larger ones, see Figure 6.3. Generally, the migration leads to a better usage of resources which is reflected in the almost unchanged U_o value for all setups, see Table 6.2. This behaviour can be accounted to the fact that jobs originating from smaller resource centres essentially act as adequate fill-ups for vacancies in the schedule of larger resource centres. Considering the job migration characteristics, one may claim that requesting jobs from cooperating sites enables local schedulers to backfill more efficiently.

Qualitatively, the results for f_{13} show similar dynamics as perceived for the two three-participant setups, see Figure 6.4. However, the improvement of AWRT seems to increase with the number of interacting sites. An explanation for this behaviour is the greater load balancing potential that arises from a larger number of partners. Overall, all evaluated setups expose a beneficial behaviour with both migration strategies.



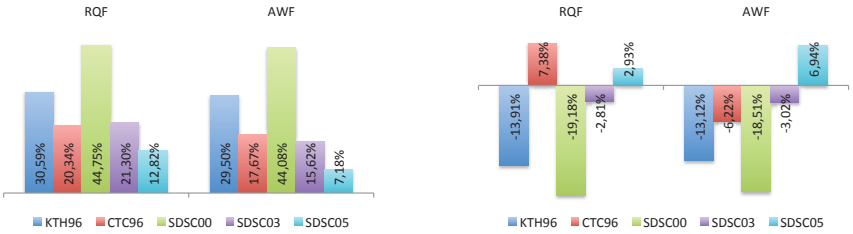
(a) *AWRT.* (b) *ΔSA.*

Fig. 6.2 *Results for the three-participant setup f_{11}, for RQF and ACF, respectively, as percental change wrt. to the non-federated case.*



(a) *AWRT.* (b) *ΔSA.*

Fig. 6.3 *Results for the three-participant setup f_{12}, for RQF and ACF, respectively, as percental change wrt. to the non-federated case.*



(a) *AWRT.* (b) *ΔSA.*

Fig. 6.4 *Results for the five-participant setup f_{13}, for RQF and ACF, respectively, as percental change wrt. to the non-federated case.*

Table 6.2 *Utilisation per resource centre and for the whole federation for the setups f_{11} and f_{12}, in percent. A delta (δ) symbol indicates a relative percental change wrt. to the non-federated case.*

Setup	Participant	U_k (%)	ΔU_o (%)
f_{11}	$KTH96_{11}$	62.18	0.0372
	$CTC96_{11}$	69.73	
	$SDSC00_{11}$	66.55	
f_{12}	$KTH96_{11}$	59.30	0.092
	$SDSC03_{11}$	67.50	
	$SDSC05_{11}$	61.67	

Migration Characteristics

For a better understanding of the delegation behaviour between the different participants, we take a closer look to job migrations. To this end, the following figures show the number of migrated jobs according to their node requirements. The values are normalised with respect to the number of occurrences in the original workload trace. While the improvements of AWRT and ΔSA are very similar for both RQF and ACF, the migration behaviour shows significant differences.

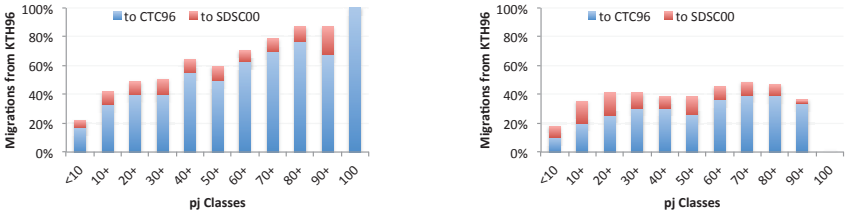
For KTH96, both algorithms show a strong tendency of migrating workload to CTC96, see Figure 6.5. This is not surprising, since in the f_{11} setup, CTC96 is by far the largest participant. However, RQF migrates a much larger amount of workload especially towards CTC96; this effect is not directly visible from the ΔSA results in Figure 6.2b.

CTC96, in turn, migrates a very small percentage of its jobs to the other partners. Obviously, this is due to its size, providing 430 processors; jobs with $p_j > 128$ cannot be migrated at all[4]. Besides this, CTC96 shows a very uniform migration behaviour: Jobs of all size classes are migrated to other participants in a more or less equal fashion, see Figure 6.6. This is in direct contrast to KTH96, where a bias towards migrating jobs with 40 to 80 requested processors can be seen.

In the case of SDSC00, we observe a similar effect as for KTH96. While migrations to CTC96 are very common and show a similar pattern as in the case of KTH96, delegation towards KTH96 is very limited. Also, RQF migrates more aggressively than ACF, see Figure 6.7, which is similar to KTH96. Again, this emphasises the aforementioned balancing effect that workload tends to move from smaller to larger resource centres.

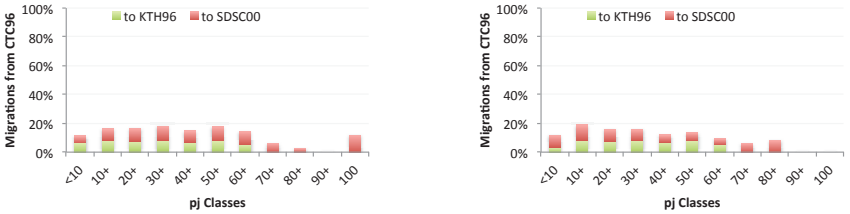
The other setup, f_{12}, shows a slightly different pattern. This is mainly because the vast difference in size of the resource centres, especially regarding KTH96, which is less than one tenth in size of each of the two others. Other than that, RQF shows a very balanced migration behaviour towards both other participants. Only the largest category seems to distort this; however,

[4] Due to the lack of sufficient resources at both KTH96 and SDSC00.



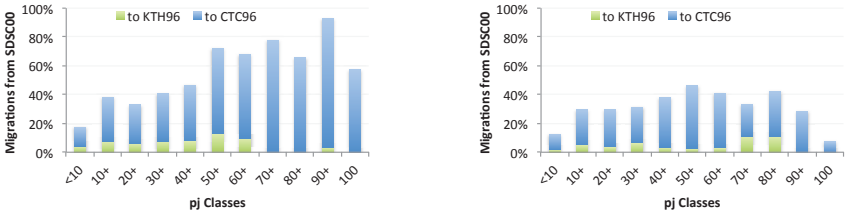
(a) *RQF.* **(b)** *ACF.*

Fig. 6.5 *Migrations (clustered along the jobs' p_j values) from KTH96 to the other partners for the three-participant setup f_{11}, for RQF and ACF, in percent of the total amount of jobs in the particular class.*



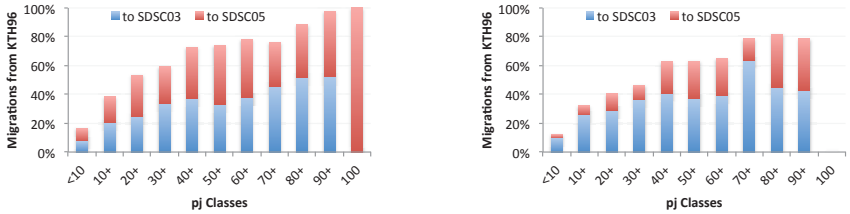
(a) *RQF.* **(b)** *ACF.*

Fig. 6.6 *Migrations (clustered along the jobs' p_j values) from CTC96 to the other partners for the three-participant setup f_{11}, for RQF and ACF, in percent of the total amount of jobs in the particular class.*



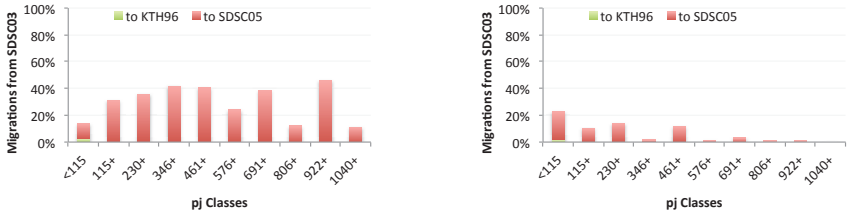
(a) *RQF.* **(b)** *ACF.*

Fig. 6.7 *Migrations (clustered along the jobs' p_j values) from SDSC00 to the other partners for the three-participant setup f_{11}, for RQF and ACF, in percent of the total amount of jobs in the particular class.*



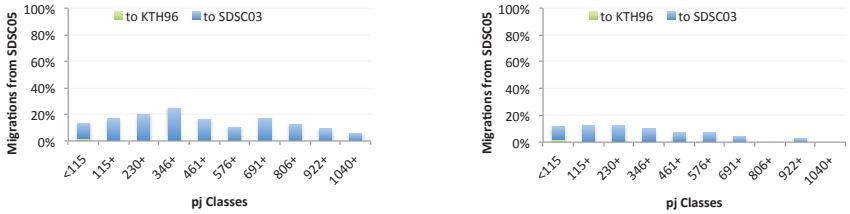
(a) *RQF.* **(b)** *ACF.*

Fig. 6.8 *Migrations (clustered along the jobs' p_j values) from KTH96 to the other partners for the three-participant setup f_{12}, for RQF and ACF, in percent of the total amount of jobs in the particular class.*



(a) *RQF.* **(b)** *ACF.*

Fig. 6.9 *Migrations (clustered along the jobs' p_j values) from SDSC03 to the other partners for the three-participant setup f_{12}, for RQF and ACF, in percent of the total amount of jobs in the particular class.*



(a) *RQF.* **(b)** *ACF.*

Fig. 6.10 *Migrations (clustered along the jobs' p_j values) from SDSC05 to the other partners for the three-participant setup f_{12}, for RQF and ACF, in percent of the total amount of jobs in the particular class.*

it must be noted that the number of jobs requiring 100 processors (i.e. the full resource centre's size) is very small[5].

For SDSC03, we observe that the amount of migrations for all p_j classes except the smallest is significantly lower with ACF, dropping to almost zero for the bigger categories. One reason for this behaviour is the amount of information available to RQF: The request-based algorithm has access to the full remote queue at all times and thus can easily find a particular slot in the local schedule at request time. ACF, in turn, offers such workload when appropriate to the local system—regardless of whether the remote system is capable to handle it at that particular moment in time. Migrations to KTH96 are obviously few due to the limited size of the resource centre.

Interestingly enough, SDSC05 shows largely the same (albeit dampened) pattern as SDSC03. This confirms the observation from f_{11} that workload tends to move towards the larger centre. Still, the larger participants show a vital exchange of workload.

Summarising, it shows that RQF has a much more aggressive exchange behaviour than ACF. Also, it shows a stronger tendency towards delegating jobs with large resource requirements while ACF favours small and medium-sized workload. From that, we conclude that ACF tends to keep more workload in the local domain the larger a resource centre's size is. While the higher exchange apparently indicates a better performance in terms of delegation, the opposite is the case: considering that the performance characteristics for AWRT and ΔSA are almost equal for both algorithms, ACF achieves comparable results while being less invasive to the local resource centres and leaving more autonomy to the LRM systems.

The migration matrices, see Equation 6.1, further reveal that the number of not migrated jobs exceeds that of migrated jobs in any direction by far. This can be perceived for the principal diagonal of the matrix M_n in both setups, where at least 70 % of all jobs are executed locally, see Table 6.3. Regarding the amount of migrated work, as denoted by M_{SA} in Table 6.4, we identify a correlation between migrated work and the difference in size of two sites. With an increasing difference in size, the larger partner tends to absorb more workload from the smaller partner; such a behaviour can be observed for f_{11}, where CTC96 takes around 35 % of the work from each of KTH96 and SDSC00. At the same time, similarly-sized sites exchange roughly the same amount of work; in f_{12}, this happens between the larger sites.

While ACF ensures that migrations happen only once, RQF gives no guarantee for this: It is possible that a job is transferred from a remote site more than once. Of course, this behaviour might lead to starvation because of continuous job circulation.

However, the quantification in Figure 6.11 shows that excessive job circulation only occurs very rarely. Even if some jobs migrate more than 100 times, the average values are approximately equal to one. This leads to the assump-

[5] In fact, only a single job in $\mathrm{KTH96}_{11}$ requires the full machine's extent.

Table 6.3 *Migration matrices M_k denoting the amount of jobs from each resource centre's original workload migrated to the other partners for both two-participant setups, for RQF and ACF, in percent.*

(a) f_{11}.

M_k	to KTH96		to CTC96		to SDSC00	
	RQF	ACF	RQF	ACF	RQF	ACF
from KTH96	72.75 %	78.24 %	21.37 %	13.08 %	5.88 %	8.68 %
from CTC96	6.73 %	4.01 %	87.36 %	87.56 %	5.91 %	8.43 %
from SDSC00	5.05 %	2.73 %	18.63 %	14.47 %	76.32 %	82.80 %

(b) f_{12}.

M_k	to KTH96		to SDSC03		to SDSC05	
	RQF	ACF	RQF	ACF	RQF	ACF
from KTH96	77.12 %	82.02 %	11.27 %	13.76 %	11.61 %	4.22 %
from SDSC03	1.48 %	1.04 %	78.18 %	82.95 %	20.34 %	16.01 %
from SDSC05	1.87 %	1.86 %	12.03 %	9.69 %	86.10 %	88.45 %

Table 6.4 *Migration matrices M_{SA} denoting the amount of work from each resource centre's original workload migrated to the other partners for both two-participant setups, for RQF and ACF, in percent.*

(a) f_{11}.

M_{SA}	to KTH96		to CTC96		to SDSC00	
	RQF	ACF	RQF	ACF	RQF	ACF
from KTH96	52.16 %	55.48 %	38.18 %	43.07 %	9.66 %	12.41 %
from CTC96	6.83 %	5.74 %	84.92 %	89.03 %	8.25 %	5.23 %
from SDSC00	7.48 %	18.49 %	35.46 %	22.96 %	57.06 %	58.55 %

(b) f_{12}.

M_{SA}	to KTH96		to SDSC03		to SDSC05	
	RQF	ACF	RQF	ACF	RQF	ACF
from KTH96	53.95 %	61.12 %	23.03 %	12.55 %	23.02 %	26.33 %
from SDSC03	1.11 %	1.21 %	69.71 %	98.67 %	29.18 %	0.12 %
from SDSC05	1.37 %	0.71 %	20.99 %	0.64 %	77.64 %	98.65 %

tion that there exists a temporal locality within the waiting queue: Even if a job that could be started immediately is added to the end of the queue, it is often backfilled directly. As such, we can assume that—with respect to the used input data sets—the migration of jobs does neither provoke artificial delays nor starvation.

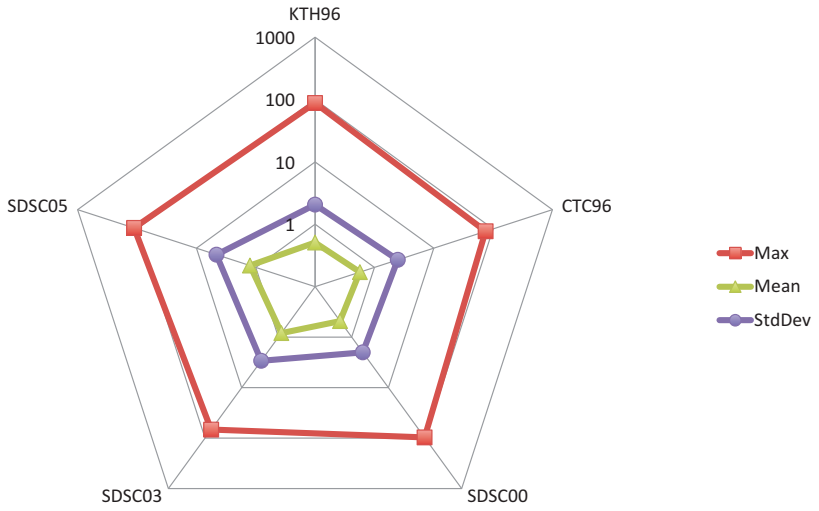


Fig. 6.11 *Number of migrations until a job is finally scheduled and executed for f_{13} with RQF, for each participant's original workload.*

6.1.4 Related Work

On the theoretical side, Hong and Prasanna (2003), assume a similar model of heterogeneous participants of a compute infrastructure federation and approach the allocation problem with a linear programming formulation, focusing on the maximisation of the steady state throughput. However, they assume equal-sized jobs only. A more production-centric investigation has been made by Hamscher et al (2000).

Chiba et al (2010) evaluate the performance of a work-stealing algorithm in an Amazon EC2- and Simple Storage Service (S3)-based ecosystem. Their approach yields good results regarding throughput, but mainly focuses on optimising the compute/data relationship between the different components. A more HPC-related view is taken by Ravichandran et al (2011), who evaluate the work-stealing approach on multi-core clusters. They find that good performance can be achieved for large-scale applications, but stay on the local level without federating with others.

6.2 On the Robustness of Capacity Planning[†]

The search for robust scheduling approaches naturally leads to the question whether heuristic approaches using purely static algorithms as discussed so far can properly cope with non-standard situations regarding resources and workload. This is especially important if considering federations with resource centres rolling their own delegation strategies and, even more importantly, if such expose (accidentally or on purpose) a malign behaviour towards the other partners.

Naturally, static algorithms cannot be expected to handle such situations in a proper way without implementing specific precautions against them, necessitating a-priori knowledge about the behaviour of the other partners. Here, situation-aware systems come into play: Algorithms that constantly analyse the current system state and adapt their decisions to the situation detected. With such, it is possible to rule out partners that do not behave in a cooperative way within the federation without knowing *how* this behaviour will look like in advance. Simple rules like ACF and RQF, as discussed in Section 6.1, however, are too inflexible to react to situations that only differ slightly, and encoding a full rule set for all possible cases (and their combinations) would be a herculean, probably infeasible task. Instead, malleable rules for decision making are required.

In the following, we explore such malleable rule sets regarding their performance in SRM for federated DCI environments. We achieve the interference of simple IF-THEN rules using fuzzy systems, and conduct the construction of proper rule sets using evolutionary computation. With the combination of these tools, we create a fuzzy-based rule system for decision making, build a novel SRM strategy on top of it, and evaluate its performance. Our pairwise learning approach lets a system evolve that realises a well-performing rule set for capacity planning in the given scenario.

With setups ranging between two and five participants, we show significant improvements in AWRT in all cases, and—at the same time—see strong robustness to unknown situations with untrained workload and alien partners. The metric-driven rule base approach proves to be stable under fluctuating conditions in the workload and, although being limited to a rather small set of simple features, showed situational awareness under circumstances of changing federation structure. Overall, we find that the rule-based approach provides good results in either case, setting the foundation for production environments with changing situations in terms of workload and environment. While the strategy provides sufficient robustness, it still is non-invasive with respect to the local management layer.

[†] Parts of this section are based on an article in the IEEE Transactions of Parallel and Distributed Systems in July 2010. The corresponding publication, see Fölling et al (2010b), is available at `http://ieeexplore.ieee.org`.

6.2.1 Evolutionary Fuzzy Systems for Distributed Brokerage

The afore described task can be condensed into two major requirements:

- *Situation-dependent, adaptive decision-making:* The current state of the system is crucial when deciding on whether to accept or decline foreign workload, e.g. allowing for additional remote jobs if the local system is already highly loaded seems to be inappropriate.
- *Robustness and stability in changing environments:* Even with respect to future, still unknown (and usually unpredictable) job submissions, it is crucial that aspects such as complete site failures or even rogue participants are handled gracefully with respect to the own *and* overall performance.

Both requirements are well-addressed by fuzzy systems. There, the controller acts depending on the current system state, and the states are modelled by fuzzy sets which in turn are represented by simple membership functions. Although the system states themselves do not contain any uncertainty the whole representation of the system is uncertain. As they have proven to be a reliable concept to tackle challenging online scheduling problems, we apply them to federated SRM in the context of this work.

For setting up proper rule sets in the fuzzy system, we use evolutionary algorithms to find working parameterisations of the aforementioned membership functions.

General Architecture of EFSs

This combination of fuzzy systems and evolutionary algorithms is commonly denoted as Evolutionary Fuzzy System (EFS), see Cordon et al (2001). Such systems build on traditional fuzzy approaches, for example the Takagi–Sugeno–Kang (TSK) model as proposed by Takagi and Sugeno (1985), but derive and optimise the expert knowledge in their rule base by using evolutionary computation. The main advantage of this approach is that the latter does not require particular knowledge about the problem structure and can be applied to various systems. On the contrary, the unknown and potentially huge search space can be explored with a minimum amount of external knowledge.

The overall EFS builds upon a set of rules, where each rule describes a certain system state, and in this specific context, we construct the rules with respect to two inputs. On the one hand, we incorporate the current state of the LRM layer, using the current utilisation of the resource centre's underlying hardware. On the other hand, we build upon the job a decision must be made for, expressing the current situation regarding workload pending for execution, see Figure 6.12.

In the fuzzy rule concept of TSK, a rule consists of a feature describing conditional part and a consequence part. For the application at hand, this

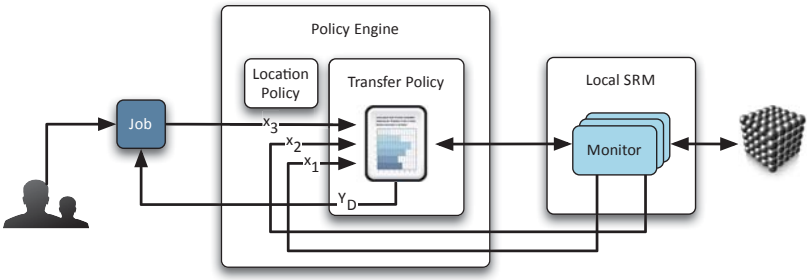


Fig. 6.12 *General architecture of the EFS controller. As the core of the transfer policy, it takes different features (x_1, x_2, x_3) as input and computes a decision on accepting or declining a given job. While the features can be derived from metrics in the LRM system (see x_1, x_2), it is also possible to use workload-related information (see x_3). The location policy is an independent part in the overall engine and has no direct influence.*

translates to a rule that decides on accepting or declining a job offered to the resource centre's SRM system through its Federated SRM layer. For example, a rule set could consist of statements made by a domain expert expressing the following:

$$\begin{array}{ccc} \text{IF } \textit{queue is long} & \text{AND } \textit{job highly parallel} & \text{THEN } \textit{decline job} \\ \vdots & \vdots & \vdots \\ \text{IF } \textit{queue is empty} & \text{AND } \textit{job not parallel} & \text{THEN } \textit{accept job} \end{array}$$

A complete rule base now constitutes the core of the rule system and essentially makes the fuzzy controller. Whenever a new job has been submitted to the local system or has been offered from remote sites, this rule system is used to make a decision.

Encoding of Rules

The general TSK model consists of N_r IF-THEN rules R_i such that

$$\begin{aligned} R_i \quad := \quad & \text{IF } x_1 \text{ is } g_i^{(1)} \text{ and } \ldots \text{ and } x_{N_f} \text{ is } g_i^{(N_f)} \\ & \text{THEN } y_i = b_{i0} + b_{i1}x_1 + \ldots + b_{iN_f}x_{N_f} \end{aligned} \tag{6.2}$$

where $x_1, x_2, \ldots, x_{N_f}$ are input variables and elements of a vector $\mathbf{x}$, and y_i are local output variables. Further, $g_i^{(h)}$ is the h-th input fuzzy set that describes the membership for a feature h. Thus, system state is described by a number of N_f features. The actual degree of membership is computed as function value of an input fuzzy set. Furthermore, b_{ih} are real valued parameters that specify the local output variable y_i as a linear combination of the input variables $\mathbf{x}$.

For our TSK instance within the EFS, we need to keep the number of parameters in the rule system small. This is mainly because of the evolutionary computation approach taken for rule optimisation. Every additional parameter increases the search space of the optimisation problem and might deteriorate the solution quality. To overcome this problem, we use Gaussian Membership Functions (GMFs) to characterise the fuzzy sets, as they two parameters (mean and variance) to specify the function shape. This makes GMFs particularly suited for the representation of rules with a minimum number of parameters.

With that at hand, we now model feature $h \in N_f$ for single rules R_i as a $(\gamma_i^{(h)},\sigma_i^{(h)})$-GMF:

$$g_i^{(h)}(x) = \exp\left\{\frac{-(x-\gamma_i^{(h)})^2}{{\sigma_i^{(h)}}^2}\right\} \tag{6.3}$$

The $\gamma_i^{(h)}$-value adjusts the centre of the feature value, while $\sigma_i^{(h)}$ models the region of influence for this rule in the feature domain. In other words, for increasing $\sigma_i^{(h)}$ values the GMF "becomes wider" while the peak value remains constant at a value of 1. With that, we are able to steer the influence of a rule for a certain feature by adjusting $\sigma_i^{(h)}$.

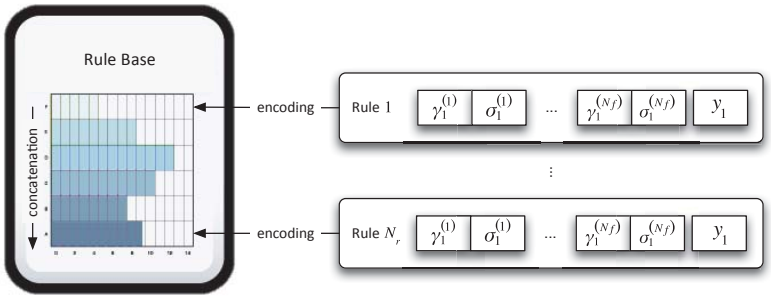


Fig. 6.13 *Encoding pattern of the rules in the EFS implementation. Each rule consists of one parameter tuple $((\gamma_i^{(h)},\sigma_i^{(h)}))$ for each feature h, and a decision variable y_i for that specific rule. Each individual for the optimisation is then constructed by concatenating the rules to a set.*

Because the GMF can be described with two parameters only, it allows to encode a feature as a pair of real values $\gamma_i^{(h)}$ and $\sigma_i^{(h)}$ as proposed by Juang et al (2000) and Jin et al (1999). With this feature description at hand, a single rule's conditional part can composed as depicted in in Figure 6.13.

For the consequence part, we further simplify the general model in Equation 6.5. Regardless of the system state, the transfer policy yields binary decisions only: Either an offered job is accept or not. Therefore, we choose the output value for a rule R_i as

$$y_i = \begin{cases} 1, & \text{if job is accepted} \\ -1, & \text{otherwise} \end{cases} \tag{6.4}$$

With this approach, all weights except b_{i0} in Equation 6.2 are set to 0 and the TSK model output becomes $y_i = b_{i0}$.

With this approach, a GMF-based TSK can be encoded with only $2 \cdot N_f + 1$ variables per rule, see Figure 6.13. This leads to $l = N_r(2 \cdot N_f + 1)$ parameters per complete rule base, given through the concatenation of the individual rules.

Decision Computation

The general TSK output function of the system $y_D(\mathbf{x})$ is computed as

$$y_D(\mathbf{x}) \quad = \quad \frac{\sum_{i=1}^{N_r} \phi_i(\mathbf{x}) y_i}{\sum_{i=1}^{N_r} \phi_i(\mathbf{x})} \tag{6.5}$$

where for a given input vector $\mathbf{x}$

$$\phi_i(\mathbf{x}) = g_i^{(1)}(x_1) \wedge g_i^{(2)}(x_2) \wedge \ldots \wedge g_i^{(N_f)}(x_{N_f}) \tag{6.6}$$

is the *degree of membership* of rule R_i. Each rule's recommendation is weighted by its degree of membership with respect to the input vector $\mathbf{x}$. The corresponding output value of the TSK system is then computed by the weighted average output recommendation over all rules.

For our implementation, we determine the actual controller output for a set of input states $\mathbf{x}$ by first computing the superposition of all degrees of memberships for a single rule R_i. For each rule R_i a degree of membership $g_i^{(h)}(x_h)$ of the h-th of all N_f features is determined for all h as the function value of the h-th GMF for the given input feature value x_h.

Considering Equation 6.6, the multiplicative superposition of all these values as "AND"-operation then leads to an overall degree of membership

$$\phi_i(\mathbf{x}) = \bigwedge_{h=1}^{N_f} g_i^{(h)}(x_h) = \prod_{h=1}^{N_f} \exp\left\{ -\frac{(x_h - \gamma_i^{(h)})^2}{\sigma_i^{(h)^2}} \right\} \tag{6.7}$$

for rule R_i, and

$$Y_D = \operatorname{sgn}(y_D(\mathbf{x})) \tag{6.8}$$

can be computed as the final controller output by considering the leading sign only, such that a positive number again represents the acceptance of the job.

6.2.2 Realisation of the Policy Model

For the implementation of the decision policies described in Section 4.2, two different approaches are taken. The location policy is handled by a simple implementation using the partner-specific AWRT; for the transfer policy, a fuzzy inference method is used.

Implementation of the Location Policy

In order to create a prioritisation of the available potential delegation targets we follow a two-step approach. As first step, we generate the subset of participants that physically provide enough machines to execute the job. That is, we rule out all resource centres with $m_k < m_j \ \forall \ k \in K$. As second step we sort remaining partners regarding their aggregated AWRT of already delegated workload. That is, so far well-performing resource centres are being favoured over those yielding higher AWRT values.

For this purpose, every participant calculates the AWRT value for every other resource centre separately, considering only those jobs that have been successfully delegated to the corresponding resource centre. More formally, we assume

$$\mathrm{AWRT}_{k,l} = \frac{\sum\limits_{j \in \tau_k \cap \pi_l} \mathrm{SA}_j \cdot (C_j(S) - r_j)}{\sum\limits_{j \in \tau_k \cap \pi_l} \mathrm{SA}_j} \tag{6.9}$$

in order to indicate how well the delegation target l was able to handle the workload delegated from k so far. We assume that a short AWRT for delegated jobs in the past is expected to yield also short AWRT values for future delegations. The outcome of this procedure is a ranking of potential delegation targets ascending in their hitherto performance.

Implementation of the Transfer Policy

The transfer policy, see Section 4.2.2, is founded on an TSK-based EFS. As described in Section 6.2.1, this model allows including an arbitrary number of features as controller input. This naturally allows rather complex state descriptions through a comprehensive number of system parameters.

However, since every additional input induces an additional variable into the state superposition, an additional (γ,σ)-pair per rule is also required. As we optimise the proposed system with an Evolution Strategy (ES), the number of optimisation variables must be kept as small as possible to keep the search space of the optimisation problem tangible. Therefore, we only use $N_f = 2$ different features to constitute the conditional part of a rule.

The first feature builds upon the jobs $j \in \nu_k$ that have been inserted into the waiting queue ν at participant k. More specifically, the *Normalised Waiting Parallelism* at participant k (NWP_k) is used as the first feature, see Equation 6.10.

$$\mathrm{NWP}_k = \frac{1}{m_k} \sum_{j \in \nu_k} m_j \tag{6.10}$$

This feature indicates how many processors are expected to be occupied by the submitted workload[6] related to the maximum number of available processors m_k. It therefore reflects the expected load of the machine in the near future.

The second feature focuses on the actual job that a decision has to be made upon. More specifically, the *Normalised Job Parallelism* denotes the ratio of a job's resource requirements m_j and the maximum number of available resources m_k at the resource centre k the job was submitted to, see Equation 6.11.

$$\mathrm{NJP}_j = \frac{m_j}{m_k} \cdot 100 \tag{6.11}$$

Translated to the afore described EFS architecture, two features (NWP and NJP) need to be covered by rules of the fuzzy controller to express the possible system states. The question that remains is how many rules are needed for the system to perform reasonably well. It turns out that there is no reliable way to determine this value from the problem formulation directly; however, Franke et al (2006a) have shown that a number between five and ten yields good results. Therefore, we pinpoint $N_r = 10$ for the work at hand.

6.2.3 Experimental Setup

In contrast to the previous experiments, this analysis requires a more detailed description regarding the experimental setup: Besides the discussion of input data and metrics, we additionally detail the configuration of the EFS and its optimisation.

Input Data and Metrics

For the evaluation of the two migration heuristics, we use AWRT and the change of SA, as defined in Section 3.3.

As input data, we use the workload traces from KTH96, CTC96, SDSC00, and SDSC05, spanning machine sizes from 100 to 1664 nodes. Again, we limit ourselves to smaller experiments because of the large computational effort necessary for optimisation. The combinations of these for the different federation setups are listed in Table 6.5. As local scheduling strategy, we apply FCFS, see Section 3.4.1. When discussing the two-participant setups, additional brackets (e.g. f_n(RC)) denote the resource centre RC under training.

As it is necessary to train the EFS, we additionally split the original eleven-month workloads used into five- and six-month sequences. We then use the

[6] Remember that the number of required processors m_j is known at release time.

Table 6.5 *Setups f_{14} to f_{19} used for the analysis of the robustness approach, based on the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup. The "5 mos./6 mos." header row indicates that, for each workload, a five-month trace for training and a corresponding six-month trace for verification was used.*

Setup	KTH96 5 mos./6 mos.	CTC96 5 mos./6 mos.	SDSC00 5 mos./6 mos.	SDSC05 5 mos./6 mos.
f_{14}	✓	✓		
f_{15}	✓			✓
f_{16}		✓		✓
f_{17}	✓	✓		✓
f_{18}		✓	✓	✓
f_{19}	✓	✓	✓	✓

former as training sequences, and the latter as data for the verification of the algorithmic performance. We exclude the SDSC00_6 trace, which we only use to investigate the behaviour of the trained system with a previously unknown participant; therefore, no five month training sequence for this workload was created.

Configuration of the Evolutionary Fuzzy System

With the aforementioned rule setup of $N_r = 10$ rules with two features respectively, the corresponding optimisation problem has $N_r \cdot (N_f \cdot 2 + 1) = 10 \cdot (2 \cdot 2 + 1) = 50$ parameters.

For the optimisation itself, we apply a $(\mu + \lambda)$-ES, see Beyer and Schwefel (2002). During the run of 150 generations, a continuous progress in fitness improvement is observable. As recommended by Schwefel (1995), we assume 1/7 for the ratio between μ and λ. Therefore, we use a parent population of $\mu = 13$ individuals, resulting in a children population of $\lambda = 91$ individuals. Hence, 91 individuals must be evaluated within each generation. As objective function we use the AWRT, see Section 3.3.4.

For the variation operators we further use the following configuration: The mutation is performed with an individual mutation step-size for each feature. As the two features vary in their domain by a ratio of 1:10, see Section 6.2.2, we use a mutation step-size [7] of 0.01 for NWP and 0.1 for NJP, as this is sufficient for the expected accuracy. A more elaborate analysis of potentially well-performing mutation step-sizes to the here discussed problem is out of scope. This mutation is applied for the conditional part of the rule as they are real values. For the binary consequence part we mutate values by flips

[7] Here, we use standard values as suggested in literature (Engelbrecht, 2007), and choose NJP as the reference, since the use of the default step size of 0.1 for NWP would imply a value of 1 for NJP, which is too high for proper operation of the EFS.

of -1 to 1 and vice versa. Further, we apply discrete recombination in each reproduction step.

The population is uniformly initialised within the ranges $[0, 10]$ for the (γ,σ)-values of NWP and $[0, 100]$ for NJP respectively. As the fitness evaluation of an individual is time consuming (from several minutes up to half an hour) we evaluated the whole population in parallel as for $|\mathcal{K}|$ participants ($|\mathcal{K}|^2 - |\mathcal{K}|$) pairings have to be simulated. During production usage[8], however, only basic mathematical operations are required that can be performed in a negligible amount of time.

6.2.4 Evaluation

With the optimised EFS at hand, we now apply the transfer policies to different scenarios, reaching for two goals. On one hand, the behaviour should be robust against unknown workload, effectively being able to handle submissions different from the original training data. On the other hand, it should be transferable to scenarios with larger federations and potentially unknown partners.

Training Phase: Unilateral Cooperation

Starting with no rule set at all, we need to bootstrap the system: A basic set of rules has to be learned. Although it is generally necessary to create rule sets for both location and transfer policy, see Section 4.2, we start with a pairwise training approach in order to reduce other partners' influences as much as possible. To this end, we limit job exchange to a single resource centre only—thus needing no location policy at all—and concentrate on the optimisation of the transfer policy. Furthermore, we evolve only one participant at a time while applying the ACF policy, see Section 6.1.1, to the other.

Pairwise training with static policies for one participant has benefits and drawbacks. The main advantage is stability: A simultaneous training of two rule bases would lead to an ever-changing environment. This makes an evolutionary-guided adaptation very difficult, as a rule base that leads to good results during one generation might fail completely in the next generation if the partner changes its behaviour completely. The main disadvantage is that the effort for training is very high, as each potential pairing of partners needs to be considered to eventually yield a comprehensive rule set.

Results for Training Sequences

As expected, the optimisation leads to significant improvements in AWRT for all examined setups[9]. Partly, this results in larger AWRT for the non-adapting

[8] That is, after the rule sets have been learned, and the participants' resource management systems apply to incoming workload.

[9] The optimised partner is listed as an additional index on the setup.

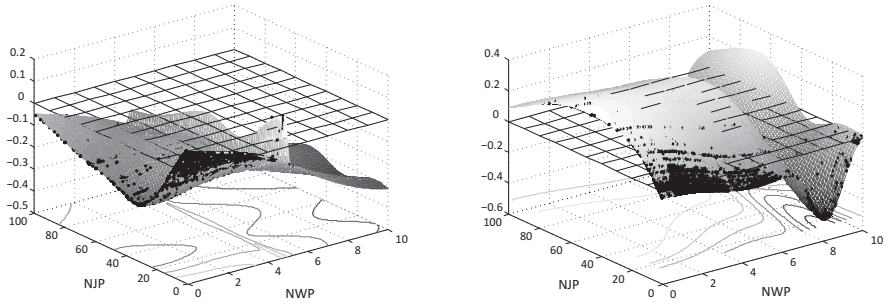
partner. For instance, in setup f_{14}, the optimisation of KTH96 leads to a worsened result for CTC96 of almost 10 %. In this specific case, this behaviour corresponds to a strong shift of work as from KTH96 to CTC96. With a change of learning focus towards CTC96, we achieve also improvements of 5.44 % for AWRT and a slight load relief for CTC96. Also, KTH96 still significantly improves, although it is not under optimisation. Mainly, this is because of its unsatisfactory performance in the non-cooperative case.

In setup f_{15}, an optimised KTH96 naturally benefits from more available resources because of the size of SDSC05. Still, the latter shows a remarkable improvement in AWRT of 3 % while at the same time SA is slightly increased. This indicates that an improvement in AWRT is not necessarily caused by smaller utilisation, but a better allocation of workload to resources due to the optimised rules. This also holds for the interaction of a medium and a large resource centre, see setup $f_{16}(CTC96)$, where CTC96 also benefits with a decrease in AWRT of more than 20 %. Likewise, SDSC05 benefits from the cooperation with the medium-sized system, see setup $f_{16}(SDSC05)$.

For a better understanding of the optimised transfer behaviour, we show in Figure 6.14 the set of characteristic curves for the different federation setups. The system states are shown on the x- and y-axes for NJP and NWP, respectively. In accordance to the EFS formulation in Section 6.2.1, the z-axis denotes the borderline between acceptance (>0) and refusal (≤ 0) of jobs.

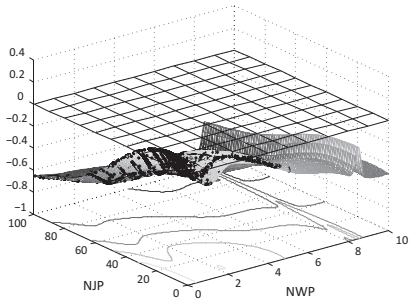
In Figure 6.14a we observe that KTH96 exposes a restrictive exchange behaviour as only small jobs are accepted and almost all remote jobs are declined. As locally submitted and remotely offered workload is treated similarly, this behaviour tends to offer all jobs to remote sites first. Also, it shows that mainly small jobs are accepted. In Figure 6.14c also larger jobs from the SDSC05 are offered as almost the whole range for NJP is activated. However, the exchange policy for the KTH96 refuses them in all cases. For CTC96 the situation is different: It does not only handle locally submitted jobs, but also accepts remote jobs that require many processors. Here, small jobs are offered to the remote site first, see Figure 6.14b. The large resource centre, SDSC05, always accepts incoming workload as long as the waiting queue is not too full, see Figure 6.14d and Figure 6.14f. Whenever the NWP value grows beyond 4, the transfer policy switches from acceptance to refusal. This is reasonable as it prevents starvation of jobs in the queue and protects the SDSC05 from overuse of resources.

Summarising, it becomes apparent that workload exchange is only beneficial if the federation participants show a minimum will for cooperation; that is, it is necessary that each participant accepts remote workload at least if suitable for him.

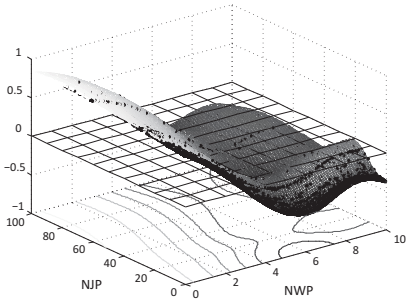


(a) f_{14}, *optimising KTH96.*

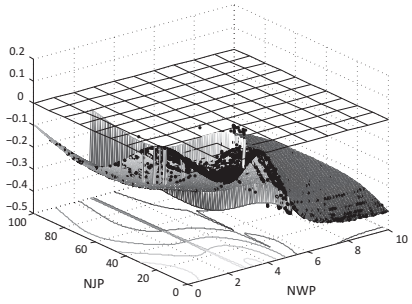
(b) f_{14}, *optimising CTC96.*



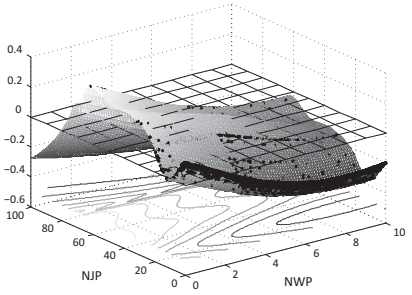
(c) f_{15}, *optimising KTH96.*



(d) f_{15}, *optimising SDSC05.*



(e) f_{16}, *optimising CTC96.*

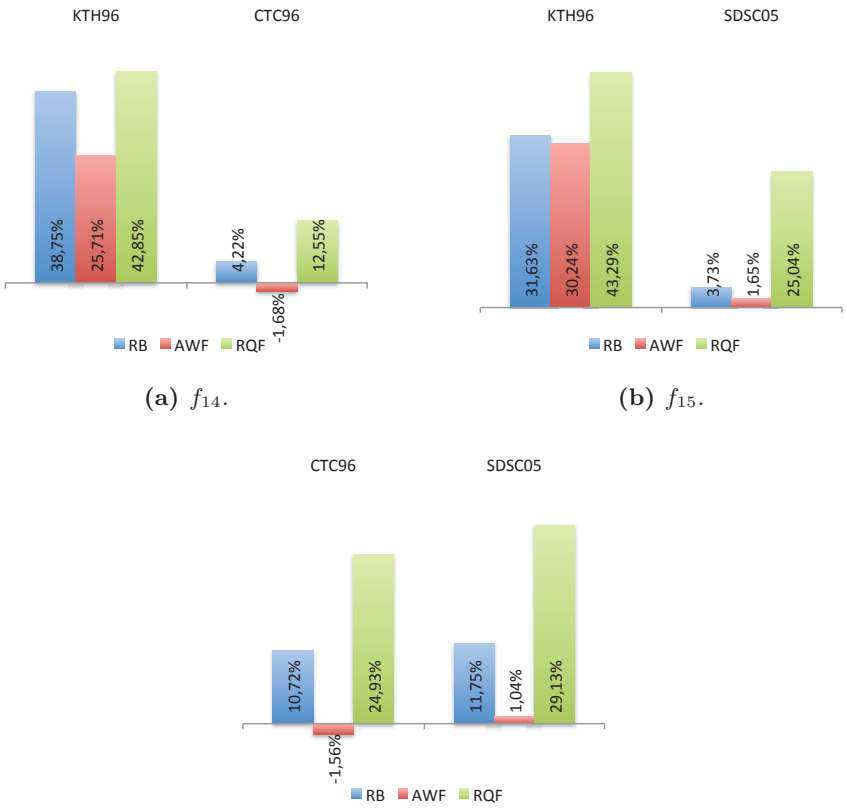


(f) f_{16}, *optimising SDSC05.*

Fig. 6.14 *Set of characteristic curves after optimisation for the different setups. The system state description is depicted on the x- and y-axis for NJP and NWP respectively. The grid at level zero on the z-axis marks the border between acceptance of jobs (>0) and refusal of jobs (≤0). Additionally, the activated areas during application on the training workloads are depicted by black points.*

Stability of Trained Rulesets

To assess the stability of the pairwise learned rule bases with respect to unknown situations[10], we apply them to the six-month versions of the workloads denoted in Table 6.5. We further assume that every resource centre applies the rule base as transfer policy that it "learned" during the interaction described in Section 6.2.4; note that the ACF policy is not used anymore.



(a) f_{14}. **(b)** f_{15}.

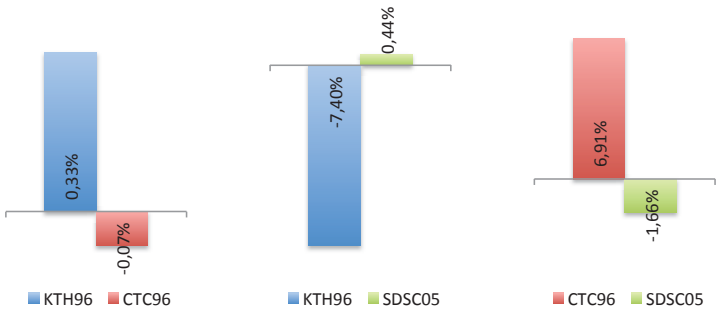
(c) f_{16}.

Fig. 6.15 *Improvements in* AWRT *for the two-participant setups using trained rule bases on untrained data sets (RB), compared to the results from the ACF and RQF strategies as described in Section 6.1.*

Obviously, the EFS approach still leads to a significant decrease of AWRT in all cases, see Figure 6.15, indicating a high robustness with respect to

[10] Specifically, regarding unknown workload, that is.

submission changes. Further, it exposes an improvement with respect to ACF as discussed in Section 6.1. This is all the more remarkable as this policy additionally benefits from the positive influence of EASY as a local heuristic. Regarding RQF, the EFS still yields an acceptable performance. Given that the static policy can select from the remote waiting queue at will, effectively exposing the other system completely to the world, the EFS produces still good results.



(a) f_{14}. **(b)** f_{15}. **(c)** f_{16}.

Fig. 6.16 *Change in squashed area (ΔSA) for the two-participant setups using trained rule bases on untrained data, as percental change wrt. to the non-federated case.*

The ΔSA values stay similar to the previous experiments in distributed architectures and do not yield any new insight, see Figure 6.16.

Drawbacks of the Training Method

A major weakness of the approach naturally lies in the complexity of the process: While it is not necessary[11] to have training set for all possible federation partners, the pairwise optimisation approach is still time-consuming and does not scale. However, it is possible to ameliorate this problem by taking a different approach on the optimisation: Fölling, Grimme, Lepping, and Papaspyrou[12] developed a learning mechanism using Competitive Co-evolutionary Learning. There, the evolutionary algorithm does not run pairwise trainings with elitist selection using one reference partner. It rather evaluates the full federation, letting the best-performing rule sets for each individual resource centre compete against the corresponding rule sets of other participants, then

[11] In fact, we assume from our experiments that it is sufficient to have a set of capacity-wise characteristic ones; still, it is an open problem how to select those.

[12] See Fölling et al (2009) for a learning-focused discussion and Fölling et al (2010a) for a fuzzy-focused discussion.

the second-best, and so forth. Naturally, the training is not adjusted to a specific partner behaviour anymore, but instead trained to well-perform in the context of a federation. Still, the approach shows a similar performance to the one discussed here.

Application Phase: Multilateral Cooperation

After setting up the basic rule sets for workload exchange in a controlled environment with only a single resource centre, we now extend this approach towards applying the rule bases in a federation scenario with more participants. Since a third partner is incorporated into the setup, two additional aspects require handling:

1. Before making a decision on whether to hand over workload from one participant to another, it must be decided *which* partner to use. This is done using the location policy as described in Section 6.2.2.
2. As each resource centre is now cooperating with several (known) partners, it needs to choose the appropriate transfer policy with respect to the corresponding partner. As such, each participant keeps a separately trained transfer policy for each resource centre it interacts with. As soon as the location policy yields the partner for the delegation under decision, the corresponding transfer policy for this very partner is selected.

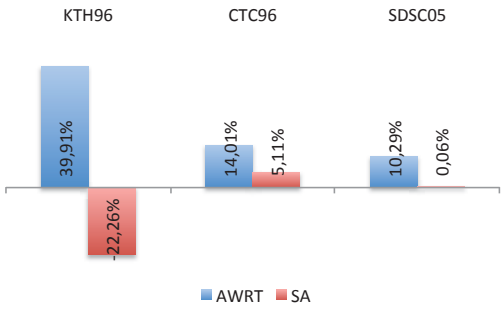


Fig. 6.17 *AWRT improvements and Δ_{SA} values for the three-participant setup f_{17}, using trained rule bases on untrained data and the location policy described in Section 6.2.2, as percental change wrt. to the non-federated case.*

The results in Figure 6.17 clearly indicate that the AWRT still improves significantly for all partners while the utilisation decreases for the smaller partner. Although CTC96 and SDSC05 are slightly more loaded, their objective values still improve.

Verification Phase: Handling Alien Participants

So far, our approach was tailored to learning a pool of rule bases for partners that are known in advance. This, however, requires knowledge about the submitted workload in order to tune the transfer policies properly. With respect to the robustness requirement, we now extend our approach to an environment with resource centres previously unknown to the federation.

As mentioned in Section 4.2, the transfer policy is applied to each participant separately. If a new resource centre enters the federation, this means for the rule base approach as discussed here that the existing participants need to select an appropriate policy from the set of available policies. Naturally, a direct mapping is not possible, as the new resource centre has not been part of the training, and a corresponding rule base does not exist.

To identify the best suitable transfer policy without having a direct match, we assume a correlation between delegation targets' maximum amount of available resources and their transfer behaviour. We conjecture that the behaviour within the federation mainly depends on a resource centre's size. Thus, we categorise the trained rule bases along the sizes of the machines they belong to. Among the pool of transfer policies the best fitting one, with respect to the number of maximum available resources, is selected to make the decision for a submitted job.

In order to assess this new situation, we investigate the performance of setup f_{15}, which features the previously unknown SDSC00 resource centre. By applying the aforementioned categorisation, the other partners fall back to the KTH96-specific rule base, because it is closest to SDSC00 in terms of machine size (100 processors compared to 128 processors). For SDSC00 itself, we use ACF as the transfer policy, see Section 6.1.1.

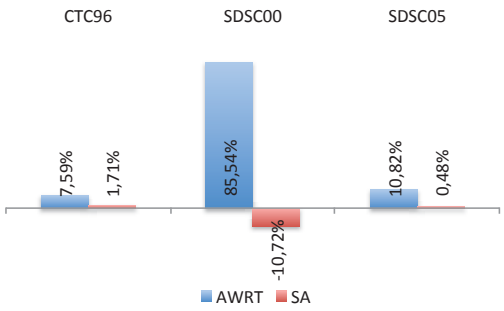


Fig. 6.18 *AWRT improvements and Δ_{SA} values for the three-participant setup f_{18}, using trained rule bases on untrained data, the location policy described in Section 6.2.2, and SDSC00 as the unknown participant, as percental change wrt. to the non-federated case.*

Figure 6.18 depicts the results for this setup. We observe strong AWRT improvements for SDSC00 due to the high potential given by the poor results for non-federated execution, see Section 3.4. However, the other participants as well improve their AWRT by at least 8 %.

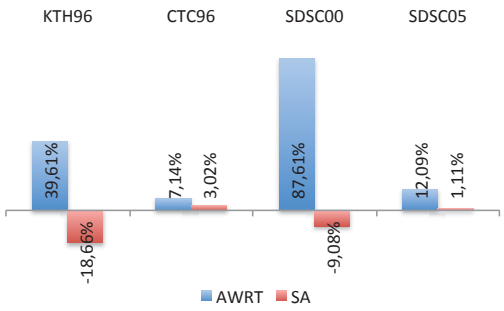


Fig. 6.19 *AWRT improvements and Δ_{SA} values for the four-participant setup f_{19}, using trained rule bases on untrained data, the location policy described in Section 6.2.2, and SDSC00 as the unknown participant, as percental change wrt. to the non-federated case.*

The same holds for a federation with four participants and one unknown partner, see Figure 6.19: Again, AWRT improvements of more than 10 % are achieved for all resource centres while the utilisation moves from the smaller participants to the larger ones.

6.2.5 Related Work

Due to the highly dynamical character of the discussed problem, several optimisation techniques have been used already. In the field of nature-inspired algorithms, Carretero et al (2007) propose a GA-driven meta scheduler and Jakob et al (2005) apply hybrid heuristics. Fuzzy systems, in turn, have only been partially applied: Huang et al (2005) determine a knowledge base for Grid computing by transforming Grid monitoring data into a performance data set. They extract the association patterns of performance data through fuzzy association rule in order to optimise the Grid scheduling. Fayad et al (2007) address a Grid scheduling problem with sequential jobs where processing times underlay uncertainty.

The capabilities of EFS have only recently been applied to scheduling: So far no endeavours have been made to apply them to federated capacity planning problems. However, results have been obtained for parallel computer scheduling with EFS: Franke et al (2006a) derived a methodology to genetically learn a fuzzy system from real workload data to establish user group prioritisation. With such a system, the owner of a parallel computer is able

to flexibly realise prioritisation for certain user groups without worsening the overall system utilisation. After a training phase, the controller enforces the desired priorities while still delivering significantly shorter response times for the favoured user groups.

6.3 On Resource-centric Capacity Planning in DCIs†

Until recently, compute resources were usually tangible "tin" in a server room that was physically accessible to the users: Albeit offerings for IT outsourcing are being marketed for many years already, many segments, especially those concerned with research and development, still followed the traditional approach of hardware being purchased, used, and—usually after its recovery period—decommissioned. With the ascend of Cloud computing in 2006, a shift of paradigm regarding the means of resource acquisition had silently occurred: A technology platform *and* business model for "leasing" resources emerged, and depending on the kind of workload, it provides a viable alternative to on-premises infrastructure, see Hill and Humphrey (2009).

In the case of DCIs as discussed in the previous chapters, this shift of paradigm allows to replace the common approach of mutual workload exchange with a method that bases on load-dependent resource configuration. That is, in case of a resource centre's over-utilisation, the SRM system can leverage the on-demand model to lease resources from other participants in the federation to keep up service quality for its local user community. Contrary, the granting of idle resources can increase utilisation in times of low local workload and thus ensure higher efficiency.

This approach—in conjunction with the general demand for adaptive reconfiguration—opens new challenges in the management of such systems. With respect to automated capacity planning, the lease-oriented model and corresponding SRM strategies need to be validated. In the following, a first step towards addressing these issues is being made: For the DCI scenario introduced in Section 2.4.2, we investigate the performance of two algorithms for leasing and granting resources. To this end, we introduce a new delegation layer and two negotiation-capable delegation policies.

Both delegation policies demonstrate their potential by realising significant increases in service quality for almost all participants. The second, more aggressive delegation approach turns out to be less robust against extremal differences in site size, leading to degradation of service quality in specific cases. Both show the dynamics of system reconfiguration in the proposed scenario: Participants frequently change their configuration in order to fit their workload. In fact, the fluctuations in local configurations range from completely granting all resources to leases that multiply the system's own size.

6.3.1 Workload Management through Leasing of Resources

The widespread offering of Cloud computing services and Infrastructure as a Service (IaaS) allows system administrators to provision additional resources,

† Parts of this section have been presented at the 15th Workshop on Job Scheduling Strategies for Parallel Processing in Atlanta, GA on April 23, 2010. The corresponding publication, see Fölling et al (2010c), is available at `http://www.springerlink.com`.

e.g. compute, storage, and even networking, on-demand without having to make a permanent investment into extending the local facilities. With respect to automated capacity planning and situation-adaptive scheduling of incoming workload, it is interesting whether the "leasing" of resources (instead of delegating jobs) is a feasible alternative to the approach used so far. Naturally, this is only true if the temporary give-away of local resources to a befriend resource centre within a larger federation yields a better overall performance for both involved partners.

System Model

The model of leasing resources instead of delegating jobs necessitates the introduction of corresponding policies to cater the change in viewpoint, and require the federation model used so far to be specified further. While the LRM layer stays responsible for the local allocation of workload to resources, the Federated SRM layer now takes care of the lease-and-grant mechanism and policy. Within the latter, resource requests are formulated and negotiated in order to adapt the local system to the current load situation.

Further—and in contrast to the previous setups—multi-site execution is allowed: Each job can be executed on any subset of processors[13] within the whole DCI environment. This is typically possible[14] for embarrassingly parallel jobs that comprise many sequential, independent invocations of the same application. Examples for this application class are parametric sweeps[15] or Single Process, Multiple Data (SPMD)-style programs. Iosup et al (2006) have shown that this class is the most widely spread one in production grids and DCI environments. Although distributed filesystem access and network latency may impair the execution speed of such applications in a multi-site execution scenario, Ernemann et al (2002b) have shown that the significant improvements in schedule quality often compensate for the inferior performance. Formally, such a multi-site job j is scheduled on $m_{j|k}$ own resources at the submission site $k \in K$ and $m_{j\nmid k}$ foreign resources using altogether $m_j = m_{j|k} + m_{j\nmid k}$ resources as defined before.

The Federated SRM layer is able to extend the local scheduling domain by leasing resources from other participants. In this way, each participant is able to gain exclusive control on foreign resources. For each job, the policy engine decides whether additional resources should be leased to increase the scheduling performance at the local site or not. Negotiation requests contain the number of desired resources and a timespan for which the participant wishes to gain exclusive access to them. Providing resource centres also apply

[13] Granted that those resources are under control of a single scheduler at that particular moment in time.

[14] Provided that data availability—for example, via a global shared file system such as HDFS as discussed by Shvachko et al (2010)—is guaranteed.

[15] Programs that repeatedly process the same input data, with varying parameter settings, that is.

their policies in order to determine whether to accept or decline the request. Decision making is based on multiple input features such as the users' job submission behaviour, the current resource usage, and the local backlog. Finally, it is not allowed to grant already leased resources to a third party.

Leasing Policies

At the Federated SRM level, the resource delegation policy steers the individual negotiation behaviour of each participant within the DCI. Here, two approaches are introduced that feature a very simple design, are minimally invasive to the LRM, and still achieve good scheduling results. Both are triggered by the submission of a single job and can be applied under restrictive information policies, that is without exchange of information between the interacting partners.

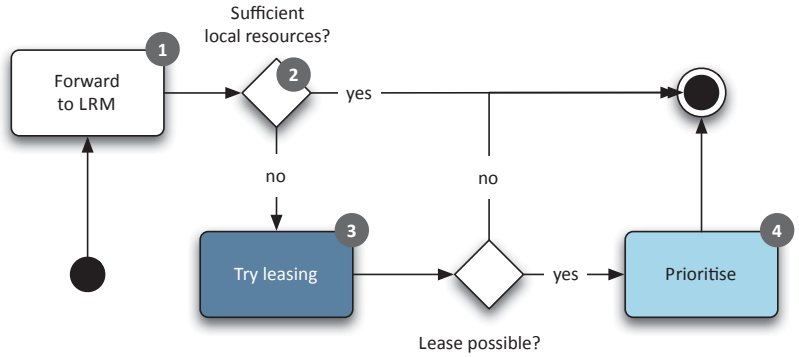


Fig. 6.20 *Decision process of the BLOS and CLOS policies.*

Basic Leasing on Submission (BLOS)

The first policy prioritises jobs that the federation can yield a matching lease for. Figure 6.20 presents the behaviour of this policy: Resource leasing is attempted every time the currently available resources cannot meet the submitted job's processor demand. After the submission of a new job to the participant, it is automatically forwarded to and enqueued at the LRM (1). Then, the federated SRM checks there are more resources idle than requested by the job (2). In the positive case, the federated SRM leaves further handling to the LRM. Otherwise, it creates a lease request with a space constraint of the difference between requested and idle resources and a time constraint of the user's runtime estimate of the job. This request is then posted to the delegation partners in the system (3). If the request is granted, the job is prioritised for direct execution (4). After completion of the job leased resources are returned to the granting site.

Chatty Leasing on Submission (CLOS)

The second policy builds upon the first, but repeats of steps (2–4), compare Figure 6.20: Each job in the queue (including the new job) is used to create a resource lease, starting at the queue's head. Obviously, this approach less penalises already waiting jobs, since they are considered first. This policy demands extensive inter-resource centre communication due to many additional resource requests and makes the extended approach less practical for the use in real scheduling systems. As such, it should be seen as an extremal case for excessive resource delegation in a federation in order to assess the achievable performance.

6.3.2 Experimental Setup

We build on the findings in Chapter 3 and reuse the same techniques for evaluation. However, it turns out that the metrics as defined in Section 3.3 require minor modifications in order to address the elasticity aspect of the infrastructure.

Assessment Metrics

The *Squashed Area* (SA_k), see Section 3.3.2 was introduced as "the amount of resources used" by the workload or a fraction thereof. Given that jobs may now partially execute on the processors of remote participants, it needs to be refined as follows:

$$\mathrm{SA}_k = \sum_{j \in \tau_k} p_j \cdot m_{j|k} + \sum_{l \notin \tau_k} p_l \cdot m_{l|k} \tag{6.12}$$

SA_k is determined as the sum both local ($j \in \tau_k$) and foreign ($l \notin \tau_k$) jobs' resource consumption fractions ($p_j \cdot \ldots$ and $p_l \cdot \ldots$) that are executed on resources belonging to site k ($m_{j|k}$ and $m_{l|k}$), see Equation 6.12.

$$\mathrm{SA}_k^{\lambda} = \sum_{j \in \tau_k} p_j \cdot m_{j\nmid k} \tag{6.13}$$

To further measure the amount of work running on leased processors from within the DCI environment, we define the "leased" squashed area SA_k^{λ} as the sum of local ($j \in \tau_k$) jobs' resource consumption fractions ($p_j \cdot \ldots$) that are executed on resources not belonging to participant k ($m_{j\nmid k}$), see Equation 6.13.

The *Utilisation* (U_k) describes the ratio between resource usage and overall available resources after the completion of all and measures how efficiently the processors of site k are used over time.

The entities

$$t_0(S_k) = \min \left\{ \min_{j \in \tau_k} \{C_j(S_k) - p_j\}, \min_{l \notin \tau_k} \{C_l(S_k) - p_l\} \right\} \tag{6.14}$$

and

$$C_{max,k} = \max\left\{\max_{j\in\tau_k}\{C_j(S_k)\}, \max_{l\notin\tau_k}\{C_l(S_k)\}\right\} \tag{6.15}$$

refer to the timespan relevant from the schedule's point of view, delimited by the start time of the first job, see Equation 6.14, to the end time of the last job, see Equation 6.15, in schedule S_k. Note that both points in time consider local jobs ($j \in \tau_k$) and fractions of delegated jobs ($l \notin \tau_k$).

$$\mathrm{U}_k = \frac{\mathrm{SA}_k}{m_k \cdot (C_{max,k} - t_0(S_k))} \tag{6.16}$$

U_k, formally defined in Equation 6.16, then serves as a quality measure from the site provider's point of view.

Input Data

As input data, three traces from the collection in Section 3.2.3 are used, namely KTH96, CTC96, and SDSC05. Our combinations are limited to two participants because we lack a leasing-tailored location policy implementation and thus cannot handle more than one partner for each participant in the federation. The setups and their corresponding combinations of workload traces are listed in Table 6.6.

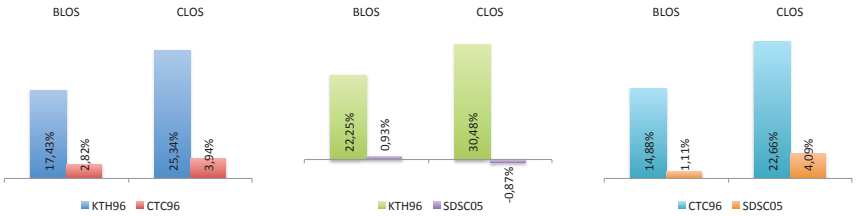
Table 6.6 *Setups f_{20} to f_{22} used for the analysis of the leasing approach, using the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup.*

Setup	KTH96 11 mos.	CTC96 11 mos.	SDSC05 11 mos.
f_{20}	✓	✓	
f_{21}	✓		✓
f_{22}		✓	✓

6.3.3 Evaluation

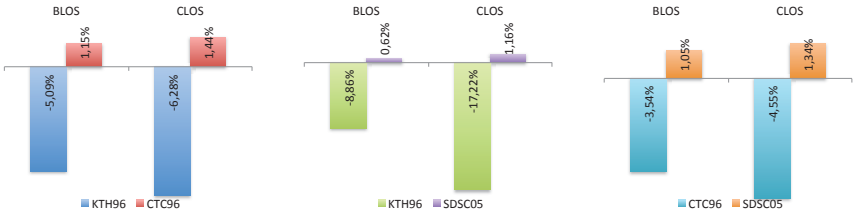
The performance of the different federation setups is evaluated in the following. All results are in comparison to the reference data as described in Section 3.4.

Almost all results show an improvement in AWRT compared to local execution, which indicates that both partners benefit from their cooperation. Figure 6.21 depicts the improvements obtained in three scenarios for both policies. The BLOS strategy yields good results, improving the AWRT of the



(a) f_{20}. **(b)** f_{21}. **(c)** f_{22}.

Fig. 6.21 *Improvements in* AWRT *for setups* f_{20}, f_{21}, *and* f_{22} *and the BLOS and CLOS policies, as percental change wrt. to the non-federated case.*



(a) f_{20}. **(b)** f_{21}. **(c)** f_{22}.

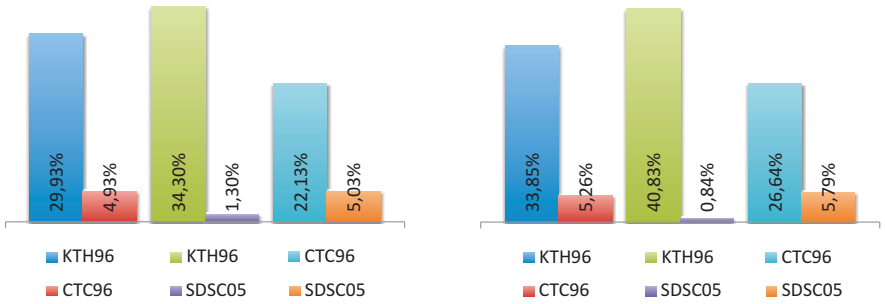
Fig. 6.22 *Changes in* U *for setups* f_{20}, f_{21}, *and* f_{22} *and the BLOS and CLOS policies, as percental change wrt. to the non-federated case.*

smaller partner for at least 15 % in all scenarios. However, a deterioration in AWRT occurs for the f_{21} compared to uncooperative processing. This behaviour is due to unbalanced exchange of resources indicated by SA_k^λ, see setup f_{21}: While the smaller KTH96 resource centre is able to increase its resource capacity, the larger participant's requests are frequently rejected for BLOS leading to higher utilisation and increased AWRT.

One can conclude that the extended strategy may yield better results for all participating resource centres but is less robust against large discrepancies in machine size: In CLOS, continuous workload submission results in frequent traversals of the complete queue. As a consequence, this gives small resource centres more opportunities to gain additional resources from the larger resource centres to execute long waiting jobs. The opposite is not necessarily true, as the resource capacity of a small participant restricts the larger participant's chances.

The large decrease in U for the small partners indicates that they benefit from the enormous resource potential provided by large partners; this was to be expected from the results of the previous experiments. However, simulations show that large partners also profit from cooperation with small

partners: Although this improvement is marginal for the SDSC05 site, the increase of utilisation indicates a compact schedule and thus better resource usage, see Figure 6.22.



(a) *BLOS.* **(b)** *CLOS.*

Fig. 6.23 *Amount of SA running on leased resources, in comparison to the total SA comprising the workload of the corresponding participant, in percent, for all setups and the BLOS and CLOS policies.*

Figure 6.23 shows the percentage of workload running on leased resources. Again, the behaviour is typical for the experiments so far: Smaller participants tend to outsource more workload than larger ones, and thus make more use of leasing opportunities. Besides this, both algorithms are very similar in terms of "leased" SA.

A look at the resource configurations over time, see Figure 6.24, indicates continuous changes: During the simulated workload period, KTH96 occasionally grants almost all its resources to the larger site, but also approximately triples its original size through leases. In the latter case, the reconfiguration almost switches the original sizes of the setup. On the other hand, they nearly keep their size on the average. This demonstrates the potential of a workload-triggered reconfiguration where the provider domain remains stable: resources adapt to submitted workload but offer an accustomed environment to users.

6.3.4 Related Work

Within a single resource centre, Subramaniyan et al (2006) analysed the dynamic scheduling of large-scale HPC applications in parallel and reconfigurable computing environments. They assess the performance of several common HPC scheduling heuristics that can be used by an automated job management service to schedule application tasks on parallel reconfigurable systems. However, their approach does not involve the interaction of multiple autonomous partners in a DCI environment. In the multi-site computing domain, Zhang et al (2006) present an adaptive algorithm that incorporates also common

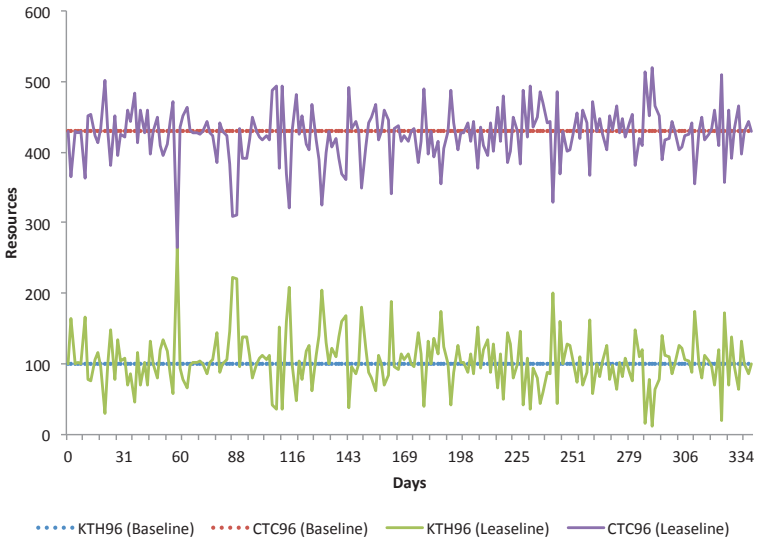


Fig. 6.24 *Leasing behaviour of KTH96 and CTC96 during workload processing of setup f_{20}. The data has been condensed to 36 hour steps; values within the intervals are indicated through the connecting lines in the diagram.*

local scheduling heuristics. Recently, Iosup et al (2007) proposed a delegated matchmaking method, which temporarily binds resources from remote sites to the local environment.

7

Discussion of the Results

Rolling in the muck is not the best way
of getting clean.
—Aldous Huxley

So far, we took a very broad approach to SRM in DCIs, discussing a plethora of results for different usage paradigms, federation architectures, and interaction models. Naturally, this diversity calls for a roundup regarding the question of how the results relate to each other.

In this chapter, we will cater to this need by making a comparison of the different algorithms introduced so far. We do this in two parts: one qualitative analysis, which compares the assets and drawbacks of the different approaches with respect to our initial research questions, see Section 1.1, and one quantitative analysis for selected algorithms to see how they perform with respect to each other regarding the metrics defined in Section 3.3.

After having discussed the benefits of our approaches to SRM in federated DCIs, we approach the question of "better scheduling performance" from the opposite side by asking for the limits of federation, directly aiming at Question 5. By using evolutionary computation, we explore these limits for a very simple reference setup and rank a selection of our own algorithms with respect to the results.

7.1 Comparison of Approaches

With a relatively large number of setups and overall six different approaches to SRM for DCIs, it is interesting to see whether and, if so, how the results relate to each other. This is especially important because of the distinction of centralised and distributed setups and the distinction of active and passive interaction as described in Section 4.3. In order to compare the results of the different algorithms and experiments with each other, we follow two approaches: qualitative and quantitative.

7.1.1 Qualitative Analysis

For the qualitative comparison, we resort to the research questions formulated earlier in this work, and transform them into five quality measures:

Deployment Effort
: The effort needed to have the algorithm under review deployed to the infrastructure. Assuming that, with a toolkit such as tGSF, the implementation effort for the Federated SRM (fSRM) level is equal for all approaches, we only review the additional effort of rolling out additional components, or preparing the algorithm for running in the environment. This measure is derived from Question 2; less effort gives a better score.

LRM Interference
: The level of intervention of the algorithm on the local level. Here, we rate how much the original strategy (such as FCFS or EASY) in the LRM system on-site is affected by the federation level algorithm. Since the re-routing of workload at the moment of submission is common to all algorithms discussed in this work, this aspect is omitted. This measure is derived from Question 2 and Question 3, as both implementation effort and participant autonomy[1] are affected; less interference gives a better score.

Autonomy Conservation
: The level of autonomy given up by the algorithm with respect to the rest of the federation. With the requirement of resource centres being independent, see Section 2.4.2, additional heteronomy introduced to the participants by the fSRM algorithm may be a showstopper. This measure is derived from Question 3; higher autonomy gives a better score.

Disclosure Level
: The amount of information disclosed by the algorithm to other participants in the federation. As discussed in Section 2.4.2, resource centres may not be willing to disclose data such as utilisation or response time. This measure is derived from Question 4; less disclosure gives a better score.

[1] Not every administrator of a resource centre is comfortable with modifying or replacing the implemented LRM algorithms.

Scheduling Performance
: The computational effort consumed by the algorithm for decision making, based on the throughput of jobs. Under simulation conditions, this aspect is important because of the extensive workload data necessary for proper evaluation. In production environments, it is crucial for the responsiveness of the overall fSRM system. This requirement is only a byproduct of the original research questions, but deserves separate discussion because of its high impact to the applicability of the designed algorithms; higher performance gives a better score.

A quality matrix resembling these measures for each of the discussed algorithms is depicted in Table 7.1.

Table 7.1 *Measure matrix for the qualitative comparison of the SRM algorithms presented in this work. Plus and minus symbols indicate better or worse results; the circle symbol indicates neutral quality.*

		Quality Measure				
		Deployment Effort	*LRM Interference*	*Autonomy Conservation*	*Disclosure Level*	*Scheduling Performance*
Scheduling Algorithm	Centralised Passive	o	--	o	o	++
	Centralised Active/ No-control	-	+	+	++	+
	Centralised Active/ Full-control	-	-	--	-	++
	Hierarchical	++	+	++	+	--
	Migration/ACF	++	+	+	++	+
	Migration/RQF	++	+	+	--	+
	Robustness	--	+	++	++	o
	Leasing	+	-	+	+	+

The centralised passive algorithm, see Section 5.1, exposes a very good performance, with approximately 0.15 ms/job, but requires very high interference with the LRM system, as the non federation-aware versions of classic strategies (FCFS, EASY, and LIST) must be replaced by their pooling-aware counterparts. The deployment effort, in turn, is moderate, as the implementation of a global pool component is rather simple. Regarding autonomy con-

servation and disclosure level, we also have a medium score, as the offering of jobs to the federation is always global, and not directed to a single, selected participant.

The centralised active algorithms, see Section 5.2, perform rather differently. In the no-control case, we have little LRM interference and good autonomy conservation, as the delegation from the coordinator to the workers requires consensus between the two levels. The disclosure level is very low, as the coordinator does not have any information whatsoever about the workers other than the response time per job after its completion. In the full-control case, we have the opposite situation; there, LRM interference and disclosure level are high, as the coordinator is allowed to analyse the workers' queues and schedules, and the workers cannot reject delegation requests, essentially giving up their autonomy completely. Regarding deployment effort, both require significant effort for implementing an active central server. The scheduling performance, however, is good, with approximately 0.25 ms/job for the full-control and 0.30 ms/job for the no-control case.

By its very nature, the hierarchical algorithm, see Section 5.3 excels at deployment effort, LRM interference, and autonomy conservation: There are no additional components to implement, Shaking-G does not interfere with the LRM at all, and autonomy is fully retained, as the flexibility of the propagation approach does not restrict participants in their workload handling. Information disclosure is little, but the algorithm relies on the exposure of the ECT to other participants for the calculation of shadow schedules. The latter is unfortunately also the reason for the comparably bad performance of 3.10 ms/job.

Regarding deployment effort, the migration algorithms, see Section 6.1 both get similar scores to the hierarchical case, for the same reason: The implementation of the fSRM layer is sufficient for both RQF and ACF, and only the fact that the algorithms rely on the assumption that, whenever a job could be executed immediately, adding to the end of the queue leads—on the average—to a good scheduling result. This is also why the score for LRM interference is slightly lower than for the hierarchical case. Autonomy conservation is also good, as the only dependency lies in the loose, consensus-based collaboration. By its very nature, RQF receives the lowest score for the disclosure level, as the algorithms requires each participant to effectively expose its complete waiting queue to the whole federation at all times. Here, ACF is much better, as its offer-centric approach does not require any information sharing whatsoever. The scheduling performance of 0.31 ms/job is equal for both algorithms and results in a good score.

For the robustness algorithm, see Section 6.2, the main showstopper is the deployment effort. With the current approach, the implementation requires significant preparations for learning a fitting rule set for a completely new environment, and the training itself relies on long-term recordings of workload trace data. Although this problem can be ameliorated by using a different approach for optimisation, see Section 6.2.4, the overhead is still high. LRM

interference is low again, although it is likely that the training result is dependent on the underlying LRM strategy. Because of the focus on building robust rules, autonomy conservation is expected to be very good, and the disclosure level is identical to the ACF algorithm discussed before. The scheduling performance receives a neutral score, as it is increased to 0.62 ms/job.

The leasing approach, see Section 6.3, requires some effort for the deployment, as it is necessary to modify the configuration of the LRM on both sides; however, most modern implementations have dynamic partitioning functionality already included as a feature. LRM interference is, by design of the algorithm, high, since the available resources are changing over time, and traditional LRM algorithms do not have explicit support for this. Autonomy conservation and disclosure level are acceptable, since the algorithm does not see the current workload of the system, although it can deduce some information from the amount of leasable resources[2]. The scheduling performance is good, with a throughput of 0.34 ms/job.

Summarising, none of the algorithms can be clearly preferred over the others, as all of them have strengths and weaknesses in certain areas. For the centralised case, the most balanced approach seems to be the no-control algorithm, which requires some deployment effort, but delivers good performance while keeping dependencies small; for large, highly independent setups, the hierarchical algorithm would be a better choice. For the distributed case, it depends on the requirements: Frequently changing environments might prefer to built upon the robustness approach; stable, long-term collaborations could also rely on the ACF algorithm. Overall, an accurate analysis of the requirements is recommended before selecting an individual approach.

7.1.2 Quantitative Comparison

For the quantitative comparison, we use the outcome in terms of metrics performance of the different setups. Naturally, the comparability is limited to the scenarios that use similar setups. Looking however at the evaluation data, it turns out that the number of comparable setups is small, and not all algorithms can be compared with each other. This has three reasons:

Repetition of Experiments
: The lack of experience in selecting reasonable setups led to changes in the "default" approach for the experiments over time. For example, the experiments in Section 5.1 showed that the LANL96 machine has special properties which influence the kind of workload eligible for submission (see

[2] This is ameliorated when deploying on fully virtualised environments; there, the administrator can underprovision and overcommit resources at any time as part of the hypervisor configuration, making it impossible to draw valid conclusions from the amount of leased resources. For as-a-Service environments, e.g. public Cloud infrastructures, it fades away completely, as the resource space appears to be unlimited.

also Section 3.2.3), but only after comparing the qualitative behaviour of this and the other approaches. We removed this workload from further experiments, but a re-simulation of all results was out of question because of the computation effort necessary for recreating the data.

Complexity of the Setup
: The evaluation of certain algorithms turned out to be time-consuming and led to the decision to employ limits on the workload number and size per simulation. For example, the computation effort for the hierarchical case, see Section 5.3) did not allow for a five-participant setup simulation because of the algorithm's inferior performance, although it would have been a good match for the centralised active (see Section 5.2) or distributed migration (see Section 6.1) approaches. Likewise, we had to limit the number of participants for the robustness experiments to a maximum of four (see Section 6.2), because the amount of participant combinations for the training phase grows linearly with the number of resource centres, and the evolutionary approach is highly time-consuming.

Properties of the Algorithm
: Limitations in the algorithmic approach do not always allow for all combinations. For example, the leasing algorithm (see Section 6.3) does not feature a means for participant selection, mainly due to the lack of reasonable metrics. This lack of location policy however limits the experimental setup to a maximum of two participants, and experiments with more resource centres could not be conducted.

That said, we actually have five[3] experiments that are eligible for a quantitative analysis: the three-participant setups $f_3 \equiv f_9 \equiv f_{11}$, allowing us to compare the centralised passive with the hierarchical and the distributed migration approach, and the five-participant setups $f_7 \equiv f_{13}$, allowing us to compare the centralised active with the distributed migration approach.

Three-participant Setups

For the three-participant setups, we do not see significant differences between the different setups regarding AWRT, see Table 7.2. It shows that, regarding overall performance of AWRT, no approach clearly outperforms the others, although the hierarchical algorithm provides good results regarding all metrics. Again, it is remarkable that, although being privileged in that sense, neither RQF nor P-EASY are able to significantly benefit from having a global view on the current workload. Contrary, the offer-based approach for ACF and the layered rescheduling approach for Shaking-G deliver similar or better with

[3] From the combination of participants, there are three more, namely $f_{14} \equiv f_{20}$, $f_{15} \equiv f_{21}$, and $f_{16} \equiv f_{22}$. However, we exclude them because the first setup of each pair is only the six-month training run for our robustness experiment and thus not eligible for comparison.

less information. This again supports our assumption that strict information policies in federated DCIs do not necessarily impair scheduling.

Regarding utilisation, the results are similar; see Table 7.3. There, we find that, while the hierarchical approach (Shaking-G) shows the highest changes in utilisation, all results lie within the same order of magnitude.

Five-participant Setups

For the five-participant setups, we find small deviations between the different setups regarding AWRT in most cases, with a few outstanding values, see Table 7.5. Still, the overall improvement within the federation stays similar, such that no algorithm is clearly favorable. It seems that the No-control case and RQF are slightly better than the others, but considering the small difference, this does not suffice for a recommendation.

Regarding utilisation, the values are fluctuating significantly. However, this is no direct indicator for the quality of the algorithm; it rather shows that the different strategies used within each setup exposes a different behaviour regarding workload migration, preferring certain participants for delegation. By all means, the significant changes with respect to the non-federated case show that there is ample room for improvement when joining a federation.

Table 7.2 *Setups f_3, f_9, and f_{11} in comparison to each other regarding their AWRT performance improvement, per involved workload, as percental change wrt. to the non-federated case. The respective best result is highlighted in bold.*

Setup	Algorithm	$KTH96_{11}$	$CTC96_{11}$	$SDSC00_{11}$
f_3	P-EASY	24.39 %	1.33 %	37.74 %
f_9	Shaking-G	**29.26 %**	**19.74 %**	33.30 %
f_{11}	RQF	25.36 %	4.69 %	**38.69 %**
f_{11}	ACF	19.84 %	2.08 %	35.26 %

Table 7.3 *Setups f_3, f_9, and f_{11} in comparison to each other regarding their U behaviour, per involved workload, as percental change wrt. to the non-federated case.*

Setup	Algorithm	$KTH96_{11}$	$CTC96_{11}$	$SDSC00_{11}$
f_3	P-EASY	−7.72 %	−4.53 %	−8.11 %
f_9	Shaking-G	−13.70 %	2.74 %	−17.73 %
f_{11}	RQF	−9.51 %	6.13 %	−11.25 %
f_{11}	ACF	−10.51 %	5.51 %	−8.95 %

Table 7.4 *Setups f_7 and f_{13} in comparison to each other regarding their AWRT performance improvement, per involved workload, as percental change wrt. to the non-federated case. The respective best result is highlighted in bold.*

Setup	Algorithm	$KTH96_{11}$	$CTC96_{11}$	$SDSC00_{11}$	$SDSC03_{11}$	$SDSC05_{11}$
f_7	Full-control	33.65 %	15.95 %	28.02 %	24.58 %	−5.38 %
f_7	No-control	**38.56 %**	**23.19 %**	34.52 %	**31.92 %**	−0.10 %
f_{13}	RQF	30.59 %	20.34 %	**44.75 %**	21.30 %	**12.82 %**
f_{13}	ACF	29.50 %	17.67 %	44.08 %	15.62 %	7.18 %

Table 7.5 *Setups f_7 and f_{13} in comparison to each other regarding their U performance improvement, per involved workload, as percental change wrt. to the non-federated case.*

Setup	Algorithm	$KTH96_{11}$	$CTC96_{11}$	$SDSC00_{11}$	$SDSC03_{11}$	$SDSC05_{11}$
f_7	Full-control	23.40 %	22.29 %	16.18 %	−2.43 %	19.01 %
f_7	No-control	−5.99 %	3.49 %	−5.45 %	−8.15 %	37.60 %
f_{13}	RQF	−13.98 %	7.57 %	−19.15 %	−2.76 %	3.03 %
f_{13}	ACF	−13.12 %	−6.39 %	−18.56 %	−2.97 %	6.78 %

7.2 Limits of Federation[4]

The methods for capacity planning in federated DCI environments we proposed in the previous chapters have—for the federation setups under review—shown to be beneficial in comparison to a non-federated environment. Improvements in AWRT up to 70 % could be achieved in many cases, and only for workloads with non-standard characteristics (LANL96, see Section 5.1.3) or the most simplistic algorithms (see Section 5.1.1), negative results were achieved at all. Naturally, this directly leads to the search for an upper bound regarding Question 5: What are the limits of the possible gain?

Obviously, this cannot be answered in full, because of the complexity of our research problem. Still, even some insight in the sense of an approximation is helpful, as it would allow to determine the distance of the results achieved so far from the bounds of an optimal solution.

In the following, we will therefore explore the limits of possible performance gain in SRM for federated DCI environments. To this end, we introduce a methodology for the approximation of optimal solutions for a capacity planning problem in the domain of distributed architectures using an NSGA-II-based approach. In detail, we evaluate a number of small federation scenarios using an evolutionary computation method and discuss the necessary steps towards this evaluation, including the development of appropriate encoding schemes and variation operators. By means of the approximated Pareto front we identify bounds for the possible gain of SRM in federated DCI environments and compare existing heuristics with respect to the Pareto front, finding new insights in quality assessment, allotment of profit, and fairness aspects. With these concepts and the obtained results, any kind of strategy for the problem domain of SRM research can be compared and ranked.

7.2.1 Preliminaries on Multi-objective Optimisation

Although so far, the presented strategies always yielded a balanced assignment of workload to the federation's participants, resulting in better results for the utilised assessment metrics, it is clear that, if not confined by the other participants' strategy behaviour, each resource centre will strive for the optimal assignment with regard to the metered objective. For AWRT, as an example, each resource centre will try to minimise the wait time component by delegating workload to other participants as often as possible[5], even at the cost of the other centres. This "egoistic" behaviour is the result of conflicting objectives for each participant: A benefit in performance for one will, at some

[4] Parts of this chapter have been presented at the 10th Annual Conference on Genetic and Evolutionary Computation in Atlanta, GA on July 14, 2008. The corresponding publication, see Grimme et al (2008b), is available at `http://dl.acm.org`.

[5] Provided that the other participants will not deteriorate the AWRT by endlessly postponing such workload.

point, lead to a detriment for the others. As such, we can consider the finding of good SRM strategies for federated DCI environments a Multi-Objective Problem (MOP).

With the notion of conflicting goals as described above, the original understanding of "optimality" needs to change: Rather than finding a single optimum, the aim in MOPs is to find good compromise or "trade-off" solutions to the problem. The solution, a set of "best compromises" has been formalised by Pareto (1964), who defined *Pareto Optimality*. Omitting a formal definition, it says that a solution $\mathbf{x}^*$ is Pareto optimal if there exists no other solution $\mathbf{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion (assuming minimisation). Such a solution is also called *nondominated*; if another solution $\mathbf{y}$ is worse with respect to one objective, and not better with respect to any other, it is *dominated by* $\mathbf{x}$. When plotting all non-dominated solutions in the objective space, i.e. depicting their objective functions' values, the resulting graph (and naturally its set of approximated values) is called the *Known Pareto front*, and the front of all **optimal** solutions is called the *True Pareto front*. With that, we follow the notation introduced by Coello Coello et al (2007).

While the domain of Operations Research has developed a considerable variety of techniques to handle such problems, the complexity of there developed solutions has spawned a number of alternative approaches. One of them can be found in optimisation algorithms that mimic the natural process of Darwinian evolution; these so-called Evolutionary Algorithms (EAs) are often applied for parameter optimisation when the fitness landscape of the optimisation problem is unknown. Nowadays, their usage is widespread in computer science and many engineering disciplines[6].

EAs translate the evolutionary process of genetic adaptation to problem-solving by assuming *individuals* to represent solutions to a given optimisation problem. Each individual comprises a *chromosome*, which is realised as a vector encoding a single solution. A chromosome consists of a number of *genes*, with each gene being an atomically modifiable component of the solution. The values that each gene can assume are called *alleles*, which comprise the alphabet of the gene. While the structure of the chromosome determines an individual's *genotype*, a concrete forming of values describes its *phenotype*; a common analogy from software engineering is the type-instance relationship. The union of all individuals form a *population* that undergoes an adaptation process.

This adaptation process is happening within the *evolutionary loop*. It is executed until a given termination criterion, like a fixed number of *generations* or a quality level within the objective space, is satisfied. After initialising

[6] In fact, many interesting industrial applications such as beam welding, aerofoil design, or reactor shielding construction comprise such problems.

the population with sensible[7] values, the evolutionary loop comprises three fundamental operations that are applied, in order:

EVALUATION
: A fitness function is used to assess the individuals with respect to the quality of the solution they represent. With that, a ranking of the population is produced that is used for the selection operations.

VARIATION
: From all individuals, λ offspring are bred by introducing modifications in their genome towards further improvement of their fitness. This modification happens either through mutation, i.e. by changing one or more alleles within a single individual's genome, or recombination, i.e. by interweaving alleles from two or more individuals, or both by applying them one after another.

SELECTION
: From the parent population and[8] the offspring, μ individuals are selected for the inclusion into the next generation. This selection is done using the ranking from the evaluation step, only allowing the fittest individuals to survive, while all others are extinct.

One concrete implementation of this concept for the multi-objective case is the Nondominated Sorting Genetic Algorithm, Version 2 (NSGA-II) as introduced by Deb et al (2002). NSGA-II's unique feature lies in the ranking procedure after the evaluation process: It classifies the individuals on the basis of non-domination. To this end, it evaluates the fitness values of all individuals and assigns the non-dominated ones to the first class or "rank" and removes them from the considered set. The removal develops a new non-dominated front in the modified set, consisting of the "second rank" individuals. This process is repeated until no more individuals are left for consideration. During the selection phase, individuals for the next generation are selected from the ranks in an ascending manner until μ is reached[9].

Of course, the domain of EAs is much broader and the afore discussed aspects only touch the surface of a large and intricate research area. However, the main topic of this chapter is the exploration of the attainable frontiers of SRM heuristics in general and federated workload management algorithms in particular and the shift of paradigm in analysis it necessitates. We therefore concentrate on the basics and the concrete algorithms used in this context,

[7] In most cases, however, this implies random initialisation, as no better starting points are known.

[8] This depends on whether a "plus"- or a "comma" strategy is used: The former, $(\mu+\lambda)$, selects from both parents and children, while the latter, (μ,λ), only allows selection from the offspring.

[9] Formally, a tournament selection is conducted, with the rank as primary and the crowding distance as secondary criterion.

and refer to Coello Coello and Lamont (2004) for an in-depth introduction into the topic.

7.2.2 Methodology

In order to apply NSGA-II to the setup discussed in Section 2.4, we need to slightly reformulate our problem to better fit the algorithm. We therefore no longer handle the workload distribution as part of the incoming workload stream processing, but rather assume an *a-priori* assignment of the workload submitted within in the whole DCI to the different resource centres. This way, the problem translates to finding an optimal partition of the overall workload; solutions can be therefore computed offline. Obviously, this approach will not yield the overall optimal workload distribution for *each* participant of the federation; instead, our methodology helps in generating reference values to rate newly designed location and transfer policies with respect to the best achievable interchange behaviour for *all* participants of the federation.

Technically, we modify the location and transfer policies, see Section 4.2, of the federation participants to forbid interaction with other participants completely; essentially, the DCI acts like a set of independent non-cooperating resource centres. To still induce interchange of workload between the partners in the federation, we model the change of workload in an offline manner by partitioning the union of all resource centres' local job streams. Obviously, this yields the same result as if the policy engines would have computed an exchange pattern leading to exactly this distribution of workload. If now a certain partitioning comprises a behaviour which leads to one participant migrating very large amounts of workload to another, the performance of the former will obviously improve, albeit at the expense of the performance of the latter. This obviously exhibits competitive behaviour, and we obtain a multi-objective optimisation problem of finding partitions with the best trade-offs for workload distribution to all participating resource centres.

Encoding Scheme

By merging each resource centre's workload trace to a single one for the whole federation and, in the second step, repartitioning them to mimic a specific distribution pattern, we attain a model that can be fit into the concept of genotype and phenotype of an individual in Evolutionary Computation: As our goal is explorative workload partitioning for the whole federation, we assume the genotype of our individuals to carry a specific assignment of workload to participants.

When merging the workload, we must keep the time order of it intact[10]. We therefore sort the merged job stream with respect to the release dates r_j

[10] Although we supposedly break the intrinsic correlations between jobs of a single workload, see Section 3.2.1.

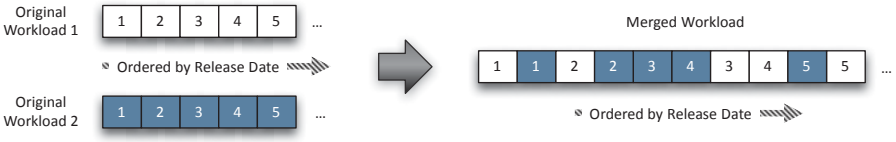


Fig. 7.1 *Process of merging the workload of each participant in the federation into a single stream of jobs.*

of the jobs, as depicted in Figure 7.1. The resulting sequence is considered as the reference workload, and its partitioning into specific resource centre assignments is represented by an individual.

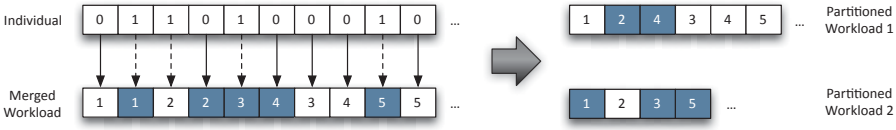


Fig. 7.2 *Scheme of encoding of an individual and the resulting assignment of workload to resource centres. In the given example, we assume two partitions.*

More specifically, each individual's genotype $\mathbb{I}$ stores a sequence of assignments $\mathbb{I} = [a_1, a_2, \ldots, a_{l(\mathbb{I})}]$ that expresses the belonging of a specific job to a specific resource centre. As such, the partitioning of the merged workload is represented by each individual's allele values. Since every job must be assigned to a specific resource centre, we require $l(\mathbb{I})$ genes; with n_k jobs for each $k \in \mathcal{K}$ out of the set of participants in the federation, $l(\mathbb{I}) = \sum_{k \in \mathcal{K}} n_k$.

Each resource centre is then represented by a number from the interval of possible alleles $a \in [0,|\mathcal{K}| - 1]$. For our examples here and the forthcoming evaluation, we limit the number of participants to $|\mathcal{K}| = 2$; this reduces the genotype to a binary representation. The mapping between geno- and phenotype then results in the desired two workloads, again preserving the order of the release dates, see Figure 7.2.

It is noteworthy that, without further provisions, this encoding scheme allows the creation of infeasible workloads: The manipulation of a gene may result in the assignment of a "foreign" job to a resource centre which requires more resources than available ($m_j > m_k, k \in \mathcal{K}$). As such, we veto such assignments during gene altering and reassign the job to its originating resource centre, using this as a kind of direct repair mechanism.

Variation Operators

Having the encoding of the problem into individuals of the NSGA-II algorithm set, their modification during the evolutionary cycle is yet to be defined. With

the plethora of variation operators available in literature, see Bäck (1996) for a survey, a proper selection that is suited to the problem at hand is necessary.

Mutation

Within the evolutionary loop, we apply two different mutations. The *random mutation* changes each gene $a_i \in \mathbb{I}$ with probability P_{rand} to

$$a_i = [a_i + \lfloor(|\mathcal{K}| + 1) \cdot \mathcal{U}(0,1)\rfloor] \bmod |\mathcal{K}| \tag{7.1}$$

where $\mathcal{U}(0,1)$ is a uniformly distributed number between 0 and 1. This mutation favours the exchange of jobs between different partitions and leads to a beneficial assignment of better fitting jobs.

Algorithm 2 Job Shift Mutation

Require: individual $\mathbb{I}$, shift step-size σ, federation size $|\mathcal{K}|$
1: $k_s = \lfloor(|\mathcal{K}| + 1)\mathcal{U}(0,1)\rfloor$
2: $n_s = \lfloor|\mathcal{N}(0,\sigma)| + 0.5\rfloor$
3: **while** $n_s > 0$ **do**
4: $p :=$ uniform random position in individual $\mathbb{I}$
5: **if** $\mathbb{I}[p] \neq k_s$ **then**
6: $\mathbb{I}[p] = k_s$
7: $n_s = n_s - 1$
8: **end if**
9: **end while**

Second, we apply a so-called *shift mutation*, which is described in Algorithm 2. During each mutation the following steps are taken: First, the participant k_s is randomly selected as the target for the workload shift (Line 1). Next, the number n_s of jobs to be shifted is determined (Line 2) and, until this number is reached (Line 3), shift operations are performed. For this purpose, a uniform random position in the individual is selected (Line 4) and, if a shift is possible[11] (Line 5), the corresponding gene is modified to reflect the change towards the target participant (Line 6). By varying the partition size through this operator, we gain a better diversity on the Pareto front.

The algorithm requires three parameters:

1. the individual $\mathbb{I}$ to modify,
2. the step-size σ which specifies the standard deviation of the normal distribution, and
3. the number $|\mathcal{K}|$ of resource centres participating in the federation.

In order to incorporate both variation methods within the evolutionary cycle we switch between both operators with probability P_{shift}. In each generation, we mutate every individual with probability P_{shift} using the shift

[11] That is, the job was not already assigned to the target site.

algorithm and with probability $(1 - P_{\text{shift}})$ with random mutation. For the latter, we change each gene with the above mentioned probability P_{rand}.

Recombination

The typical application of NSGA-II implies the dominant influence of the recombination operator. Since its specific effect on scheduling problems is not sufficiently studied, we resort to choosing among the standard operators, see Michalewicz (1996).

Uniform Crossover (UCX), where each bit is swapped between two parents with fixed probability, is not appropriate as a recombination operator. It only swaps jobs between parents, which would result in the same number of assigned jobs to each partition. As discussed before, this behaviour is undesirable, as it would result in almost no variation in partition size which is a necessity for discovering a diverse front.

n-Point Crossovers (NPXs), in contrast, are capable of combining both the desired variation characteristics in partition size and enabling exchange between the partitions at the same time. For the evaluation in this work, we therefore apply a Two-Point Crossover (TPX) operator with a probability of P_{recomb}.

Objective Functions

In order to rank the individuals during the optimisation process, it is necessary to formulate an objective function that allows to evaluate the quality of each evolving individual. As our focus lies on the improvement of workload distribution in federated DCIs, it is reasonable to choose among the metrics for performance assessment that have been used already throughout this work.

The natural choice seems to be AWRT: As discussed in Section 3.3.4, it is independent from the resource characteristics and expresses a holistic point of view on the SRM performance, embracing both aspects of utilisation and wait time.

Lower AWRT values for a certain resource centre necessitate in higher AWRT values for another, if both participants are sufficiently loaded; this exposes the conflicting nature of collaboration. We therefore determine the Pareto front of AWRT values that can be achieved in a federation of different participants, for each participant. More formally, we try solve the multi-objective optimisation problem

$$\text{MOP} := \min \begin{pmatrix} \text{AWRT}_1 \\ \vdots \\ \text{AWRT}_K \end{pmatrix} \tag{7.2}$$

by finding good assignments of workload to resource centres.

Unfortunately, the use of AWRT as the objective function for each individual's quality results in a search space that is extraordinarily large and therefore

hard to discover: When most jobs are migrated to one participant, leaving the other one with almost no local workload, the AWRT values for the former may deteriorate extremely. This is because even the most efficient local scheduling system is unable to compensate for a workload that over-utilises the available resources. Within the evolutionary optimisation process, such configurations would however still be considered as valid solutions, even although being of low quality.

In practice, however, configurations overloading one resource centre are not acceptable for production systems and therefore of minor interest for the evaluation in the context of this work anyway. In order to accommodate the need for expedient results, we impose a threshold on AWRT values for being still acceptable. The rationale behind this lies in the direct relationship between resource over-utilisation and increases in AWRT: Emerging local congestions directly result in very large AWRT.

It is therefore reasonable to limit the search space to meaningful AWRT values. We therefore assume that AWRT values which exceed the results obtained when completely obviating collaboration by 30 % and more foil the purpose of participating in the federation on the whole. Hence, we restrict possible response time values using

$$\mathrm{AWRT}_k \in [0 \ldots 100{,}000] \;\forall\; k = 1 \ldots K \tag{7.3}$$

such that if an individual achieves a higher AWRT, its objective is assigned an infinite value. This way, we effectively discard such results from the prospective evolutionary process and thus expedite the overall optimisation.

Generation of an Initial Population

Even with the limitations in search space applied as discussed above, the size of the search space further necessitates to start the evolution in an area of interest, i.e. improving from already "good" configurations. As the minimum improvement for any Pareto-optimal solution implies to perform better than the SRM performance in a non-federated setup, see Section 3.4, we take the allocation as computed by the EASY heuristic as the start solution.

In order to create a sufficient amount of start solutions, we use the workloads from the original recordings, simulate their processing under the EASY regime and shuffle them by uniformly swapping the allocation of 5,000 jobs. This is achieved by flipping their corresponding genes in each individual, guaranteeing a certain degree of diversity close to the reference solution.

7.2.3 Experimental Setup

Again, the experimental setup for the evaluation at hand requires a more thorough discussion. This includes input data and metrics as usual, but also

comprises the configuration of NSGA-II and the discussion of reference algorithms for the classification of the obtained Pareto front.

For the evaluation of the individuals themselves and the comparison with the reference algorithms, we use AWRT as metric, as defined in Section 3.3.

Table 7.6 *Setups f_{23} to f_{26} used for the analysis of the bounds approach, using the traces described in Table 3.2. A checkmark in a federation's row indicates that the trace from the workload column is used for the setup.*

Setup	KTH96 11 mos.	CTC96 11 mos.	SDSC00 11 mos.	SDSC03 11 mos.	SDSC05 11 mos.
f_{23}	✓		✓		
f_{24}				✓	✓
f_{25}	✓	✓			
f_{26}	✓			✓	

As input data, we use the workload traces from KTH96, CTC96, SDSC00, and SDSC05. The combinations of these for the different federation setups are listed in Table 7.6. As local scheduling strategy, we apply EASY, see Section 3.4.1.

Regarding the configuration of NSGA-II, we use a population size of $\mu = 70$ individuals with a total of 200 generations for each simulated setup. We apply tournament selection without replacement and a tournament size of 2. In detail, we randomly select two individuals without replacement and copy the best individual to the mating pool. This process is repeated until μ individuals are selected. Historically, NSGA-II applications favour recombination while mutation plays a minor role. Preparatory studies however showed that for having a good balance between exploration and exploitation, we need to apply both. This leads to the following parametrisation: As the crossover operator, we use TPX with a probability of $P_{\text{recomb}} = 0.9$. For mutation, we apply shift and random mutation with equal probability of 0.5, using a shift step size of $\sigma = 2{,}000$. The random mutation flips each gene with a probability of $P_{\text{rand}} = 0.1$. Both chosen values for probability are the suggested standard values, see Engelbrecht (2007).

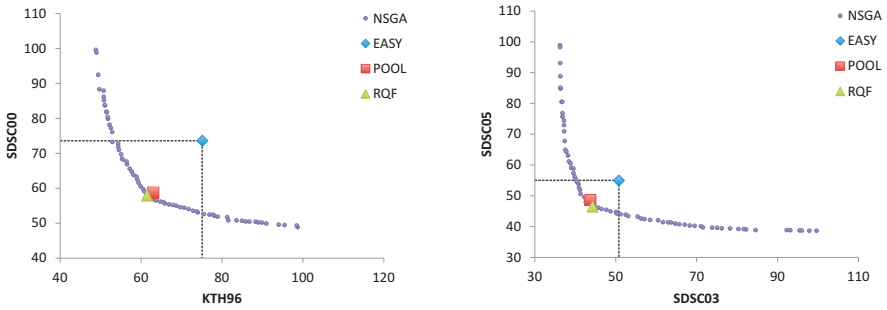
In order to verify the approximation results produced by our methodology, we generate a solution for job interchange between two resource centres using the RQF heuristic as discussed in Section 6.1.1. Due to the combination of requesting potential backfilling candidates from participants in the federation and the inherent characteristics of the EASY algorithm, we anticipate the RQF heuristic to perform very well for the given setup. As discussed before, the assumption of a holistic view on the waiting queues of remote participants is unrealistic, see Section 6.1 and Section 5.2. Still, this approach is justifiable against the background of our stated goal to discover theoretically achievable

results. Additionally, we use results determined using the P-EASY heuristic as discussed in Section 5.1.

7.2.4 Evaluation

The obtained results are depicted in four figures for each setup separately. In addition to the Pareto front, each figure contains the results of the P-EASY strategy, the RQF heuristic, and the EASY algorithm in the non-federated case, for both objectives. Further, we mark the *area of benefit* limited by the EASY algorithm results and the coordinate system's origin.

In all examined cases, it is possible to find a diverse front within the specified search interval. Obviously, there exists a large set of trade-off solutions for workload exchange apart from the already known heuristics' results. In order to judge on the quality of the NSGA-II-generated Pareto front we can only refer to the single RQF heuristic reference result. As all Pareto fronts contain this solution we conjecture that also the rest of the front is approximated appropriately. Further, this indicates that the application of NSGA-II to the presented problem of job exchange in federated DCI environments works well.



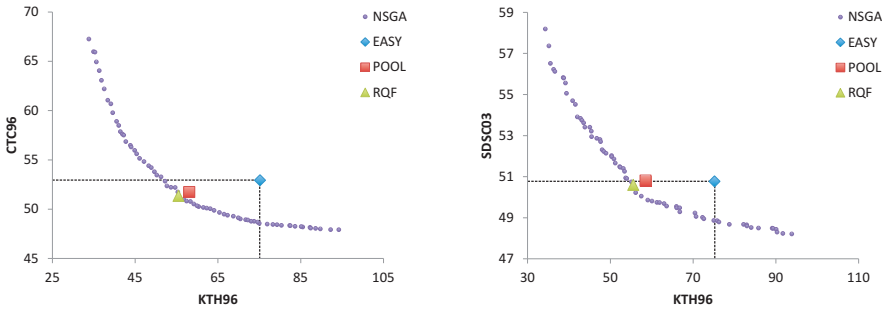
(a) f_{23}. **(b)** f_{24}.

Fig. 7.3 *AWRT results, in thousands, for the setups f_{23} and f_{24} with the corresponding workloads after* 200 *generations of evolution with NSGA-II. In addition, the results for the RQF and P-EASY heuristics are shown, as well as the performance with local execution using EASY. The dotted lines indicate the area of benefit compared to non-federated, local-only execution.*

In detail, Figure 7.3a and Figure 7.3b show the results for two similar-sized federation setups. In both cases, a convex Pareto front is obtained which covers the whole range of the search space. The heuristic algorithms produce results in the centre of the front where the RQF solution is actually reaching it. Figure 7.4a and Figure 7.4b yield similar results. However, compared to the non-federated EASY solution, the potential of AWRT improvement is much

smaller for the larger site than for the smaller site; note the different scale of the vertical axis in both figures.

Using our proposed methodology we discover two bounds for the benefit of federated DCIs: (a) the non-collaborative case where no workload is exchanged marks the upper bound, and (b) the Pareto solutions within the area of benefit which mark approximated lower bounds among the beneficial trade-off solutions. Reasonable real-world SRM heuristics should therefore yield performance results within these bounds in order to be competitive in a federated DCI scenario. Further, those heuristics become comparable as they can be ranked with respect to both the Pareto front and the EASY results.



(a) f_{25}. **(b)** f_{26}.

Fig. 7.4 *AWRT results, in thousands, for the setups f_{25} and f_{26} with the corresponding workloads after* 200 *generations of evolution with NSGA-II. In addition, the results for the RQF and P-EASY heuristics are shown, as well as the performance with local execution using EASY. The dotted lines indicate the area of benefit compared to non-federated, local-only execution.*

Exemplarily, we analyse the performance of the P-EASY strategy for the examined setups: By utilising a central pool, this strategy is empowered to reach an almost perfect compromise between partners. Furthermore, knowledge about the Pareto front allows more advanced evaluations of a heuristic's quality. In our case studies, it is possible to show that the common heuristics already tend to balance the load between participants. As a consequence, the resulting AWRT values range in the same order of magnitude while representing the most balanced trade-off solutions in the whole front. However, for the size-wise heterogeneous setups f_{25}, see Figure 7.4a and f_{26}, see Figure 7.4b, these solutions seem to be fair when the whole Pareto front is taken into account, but considering only the area of mutual benefit reveals that those "fair" solutions cannot develop the full potential of trade-offs disclosed within this area. In other words, a well-performing SRM heuristic for workload exchange

in federated DCIs should not just result in *a* compromise within the whole Pareto front, but rather achieve balanced results in the whole area of benefit. It is therefore needless to say that advanced heuristics should provide means for configuration that allow the operator to reach any point on (or near) the front.

7.2.5 Related Work

In the area of traditional production scheduling, Lei (2008) tackles the multi-objective job shop problems using Particle Swarm Optimisation and aims to minimising the makespan and total tardiness of jobs as competing goals. The non-federated case as introduced in this work is discussed by Grimme et al (2008d), who evaluate parallel machine scheduling using the Predator-Prey model on the bi-criterial $1|d_j|\sum C_j,L_{max}$ problem, focusing on the effects of the heuristic itself.

Computational Grids also have been tackled with these techniques: Liu et al (2010) propose a method for scheduling in computational Grids using Particle Swarm optimisation for single-objective optimisation of the total completion time and evaluate their approach using Genetic Algorithms. Xhafa et al (2007) apply cellular memetic algorithms to the optimisation of makespan in batch scheduling environments in a Grid context. Abraham et al (2008) apply different nature-inspired optimisation methods such as Simulated Annealing and Ant Colony optimisation on problems with minimum completion time and maximum utilisation objectives. The latter method is also used by Fidanova and Durchova (2006) who introduce a task scheduling algorithm for workflows using this method.

Although the application of nature-inspired approaches in SRM is a fairly new development, the broadness of methods is surprising. An overview of applied techniques in the context of SRM in DCI environments is given by Xhafa and Abraham (2008).

Part III

Blueprints for SRM in Federated DCIs

After demonstrating the benefit of federated SRM in DCIs on the algorithmic level, we now switch focus to the technical point of view, addressing our third goal and attempting to answer the second part of Question 1. The design of blueprints for the architecture of such systems naturally deserves its own analysis.

Since we aim for a technical foundation that has the potential of being adopted in real-world scenarios, we review the experience from several years of production in the Collaborative Climate Community Data & Processing Grid (C3Grid) and Plasma Technology Grid (PTGrid) projects. For the inevitable architectural changes over time, we find that, although its architecture has proven successful for the use case, the original design has a number of fundamental gaps with respect to the incorporation of the SRM methodologies developed in Part II: From the three general steps regarding SRM, namely interaction, decision, fulfilment, we so far only addressed the second. To overcome this, we introduce a generalised interaction model for capacity planning in federated DCIs called fedSRM.

Naturally, the two remaining parts require concretisation. We therefore adapt existing means of SLA handling to fedSRM and construct a protocol called AoR for managing the interaction part between participants of a federation through negotiation and agreement. For the fulfilment part, we focus on the problem of provisioning resources, whether capacity for a job delegation, or machines for a resource lease, and introduce a novel, fully standardised protocol called OCCI to cater this.

8

Lessons Learned from Modern Production Environments

We are stuck with technology when what we really want is just stuff that works.

—Douglas Noel Adams

In 2005, the D-Grid Initiative formulated the vision of a national e-Science platform capable of delivering collaboration technology for the handling of computation-intensive processing to specific scientific domains through an overarching, distributed infrastructure, see Schwiegelshohn (2010). As part of this initiative, two projects, namely C3Grid and PTGrid, developed a service oriented DCI to support workflow-based scientific applications their user communities (Grimme and Papaspyrou, 2009).

Since its initial deployment in 2007, this infrastructure had proven to be flexible enough to cater most of the communities' needs: Several changes in the use cases, including high-end visualisation, federated security, and basic accounting functionality, could be addressed by the architecture by only making minor changes in the implementation.

Naturally, every architecture has its limits. This became visible for the first time with the integration between C3Grid and the Earth Science Grid Federation (ESGF). Since this use case was not foreseen during initial design, it was necessary to "misuse" components for originally unintended purposes to achieve a prompt implementation. This, however, unearthed that, from the SRM perspective, fundamental gaps had to be bridged, as now the integration of cross-domain resource management became an issue. As we have seen in Part II, this was not an algorithmic problem: Intelligent distribution of workload, even among non-cooperative partners, has proven successful, and even the concept of leasing alien resources for a limited time is manageable with good results. The architecture, however, does not support either of the approaches without further ado.

In order to make the discussed SRM algorithms available to real systems, we therefore need to identify the obstacles that prevent the adoption in production environments to be able to propose a general blueprint for federated SRM architectures.

8.1 Architectural Obstacles towards Modern SRM

Considering the algorithms discussed in Part II against the background of the model defined in Part I the overall process of SRM can be split into three coarse parts:

INTERACTION
: where the SRM systems wishing to collaborate exchange information on the terms and conditions, essentially acting as service consumers and service providers;

DECISION
: where both the system's strategies are making the decision whether and how to contribute to the collaboration; and

FULFILMENT
: where the SRM system that provides the service ensures the accomplishment.

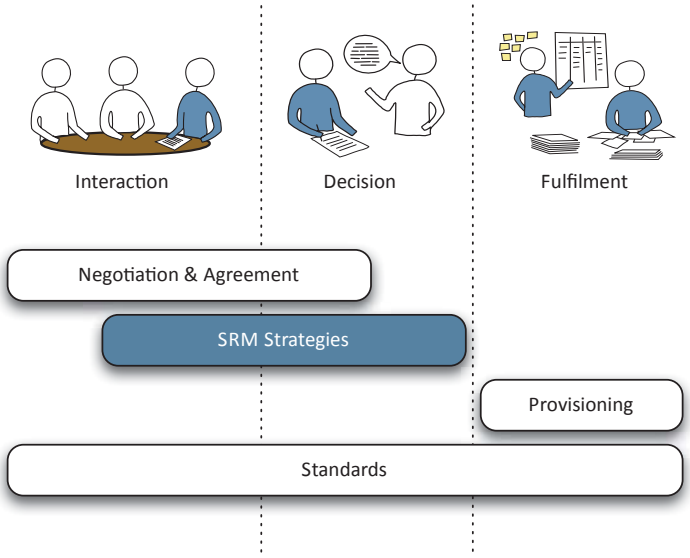


Fig. 8.1 *Core aspects of an SRM architecture for federated DCIs.*

For an architectural blueprint, these parts translate to a number of components, depicted in Figure 8.1. *Interaction* maps to the process of **negotiation and agreement**, essentially covering the inter-SRM protocols. *Decision* maps to the strategies discussed in Part II. However, both are interweaved, and partly take place in the domain of the other, respectively. *Fulfilment* maps

to the process of **provisioning**, therefore affecting the protocols between the SRM system and the underlying infrastructure.

The decision part is largely solved: An architecture can be directly derived from the model proposed in Section 2.5. The other two—together with the cross-cutting concern of **standardisation**—must be still considered as open problems.

8.1.1 Handling of Negotiation and Agreement

Apart from the decision making itself, the delegation of workload or leasing of resources is assumed to be a rather simple process: A call to action, either request or offer, is made from one participant to the other. The providing party then uses its strategy to evaluate whether it should be accepted or declined. Finally, the original call is replied to with a "yes" or "no". In the case of multi-round decision making, this process is iterated several times with changing parameters and may include countering requests or offers that eventually converge to a decision.

This process of agreeing upon such a delegation or lease is very important for real-world scenarios. For example, the IPCC Assessment Reports define certain criteria for participating DCIs—here, C3Grid—regarding the underlying infrastructure, deadlines for result submission, and quality of the data. On the architectural level, this translates to a certain QoS level, which needs to be enforced on the infrastructure. That is, it is necessary to request and provide binding commitments regarding the availability of certain services and resources between the IPCC and C3Grid officials. This is especially important because of the federated nature of DCIs. Since resource usage extends beyond the own administrative domain, QoS cannot be enforced anymore through local policies, but has to be agreed upon with the external service provider.

Although the theory behind negotiation and agreement between independent agents is well understood, see Jennings et al (2001), the technical realisation is rather complicated. This is mainly because of the term "binding", as the electronic assurance to deliver a certain service at a certain quality has to translate to a judicial "pacta sunt servanda[1]" when used in the context of business scenarios. For C3Grid, this became relevant with reinsurance companies expressing their interest in the technology for the assessment of cataclysm probabilities in certain geographical regions. For PTGrid, this was a necessity from the beginning, as project contracts between manufacturing companies and consulting bureaus built the basis for collaboration.

The transformation between the two worlds is done by electronic contracts that bear a non-repudiable statement on those terms, and are known as SLA. Their use is widely accepted in this context, and alone until 2007, more than ten projects were either already providing an implementation, working on an implementation or planning to implement Service Level Agreements (SLAs) in

[1] Latin for "agreements must be kept".

existing systems, as two surveys (Talia et al, 2008; Parkin et al, 2008) show. This trend is continued in recent years by efforts such as PHOSPHORUS, BREIN, and SmartLM, with different technical realisations.

8.1.2 Provisioning of Infrastructure

The second obstacle relates to the steps after the SRM decision: When having negotiated, decided, and eventually agreed on delegating workload or leasing resources, how are they provisioned so that they can be used?

While the algorithmic approach has ruled out this aspect as irrelevant, the deployed environment circumvented this problem by the use of a design pattern and the assumption of a certain software stack for the LRM level: The participating resource providers agreed to provide Globus Toolkit as the main interface for workload submission, and changes in the protocol or resources that have not adopted this middleware were addressed by proprietary handlers in the execution service layer[2].

This approach has two problems. First, it is tailored to DCIs that provide resources in an on-premises fashion, and does not work well if resources must be provisioned on demand. Second, it is inefficient.

The former, on-premises issue, becomes apparent when looking at the PTGrid user community. There, customers that build new machinery for plasma coating give their blueprints to specialised consultants who answer questions regarding the feasibility of certain configurations with highly compute-intensive simulations that mimic the subtle physical interaction through state-of-the-art research models. In this business ecosystem comprising Small and Medium Enterprises (SMEs) in mechanical engineering, physicist modellers of plasma gas behaviour, and consultancy bureaus, the community *relies* on large-scale Distributed Computing Infrastructure, but does not *operate* any. With the spot market for computing power offered by the Cloud computing business, this is not a problem per se. The traditional on-premises model, however, does not work anymore, as the customers can (and, for cost reasons, will) choose among resource centres and switch service providers over time. For C3Grid, similar requirements arose when users started to ask for the inclusion of AWS resources into the DCI.

The latter, efficiency issue, is a weakness of the proxy/adapter pattern itself. On the one hand, the use of an internal interface to handle differing providers introduces an additional level of indirection into the system, therefore affecting latency. On the other hand, it is hard to maintain: Whenever a provider changes his proprietary API, the corresponding driver needs to be updated. With support for a large number of providers being crucial for such bridging systems, the necessary maintenance overhead constantly increases.

[2] This follows the proxy/adapter pattern approach, see Gamma et al (1994), essentially exposing a common API and providing backend drivers for each infrastructure provider.

In the domain of Grid computing, a number of proposals for common provisioning interfaces have been made, and two of them, namely OGSA-Basic Execution Service (BES) and DRMAA are well adopted in production. For cloud computing, each vendor offers his own API: Amazon exposes the AWS API for its offerings, OpenNebula provides OCA, and so forth. A good replacement for the prevalent proxy/adapter approach that is able to cope with both, however, is still to be found.

8.1.3 Standardisation of Interfaces

The third obstacle relates to the interfacing between participants. For monolithically designed architectures, such as C3Grid, this is not a problem[3], as all stakeholders took part in the requirements engineering process, and certain decisions regarding interface compatibility could be mandated by project policy. In other contexts, this is not possible: The PTGrid example shows that, whenever external services get involved, a number of incompatible interfaces must be catered by the SRM system, which (as discussed above) is not desirable.

Integrators faced the same problem with network software APIs before the advent of TCP: Customers wanted to be able to buy from any vendor, possibly several at once, without having to change how their applications were written to use that vendor's software. A very practical solution for the market would have been to pick one vendor as the "standard," and for every competing system to then duplicate that vendor's API.

Unfortunately, this approach has some problems. The first is asymmetry: It privileges a single vendor who can then dictate the terms for use of its API. In most cases, this means that whenever the vendor changes its API, everyone else must follow. But the vendor is under no symmetric obligation to cooperate — for instance, by warning others of changes. Worse, the vendor can introduce commercial and legal frictions, including fees and patents. The second problem is fitness: In early stage markets, arguing that a single vendor is fit for all common purposes is difficult, because use cases are still emerging. Especially from a federation integrator's point of view, both aspects pose serious obstacles.

With the standardisation of TCP, the community solved these problems by picking a technology specification that described real systems with broad cases that were not vendor controlled. Furthermore, by choosing a suitable legal framework under the patronage of the Internet Engineering Task Force (IETF), TCP users could have confidence that using the technology does not entail legal problems or even litigation. By enabling interoperable networking, TCP solved the integration problem.

This yielded two chief benefits. The first was commoditisation: The creation of an open marketplace for TCP software and solutions providers, as

[3] But will become one in the future, for the reasons stated in the previous section regarding external infrastructure providers.

well as an ecosystem of add-on applications, drove down costs. The second was federation: It became possible to build upon a common interface layer and build larger structures without having to separately establish agreements on the behaviour with each and every provider. So, not only did this solve the integration problem, but everyone could build better systems faster and bring new business to market at lower cost, and eventually, TCP became widely accepted as the *lingua franca* for networking.

8.2 A Blueprint for SRM in Federated DCIs

With the core aspects of an SRM architecture as discussed in Section 8.1 and the proposed solution strategies—the use of SLAs, a common provisioning interface, and standards—at hand, the shape of blueprint becomes clear: an architecture for infrastructure federation through automatically negotiated, SLA-secured, dynamically provisioned resources and services. In order to make this more tangible, we identify the two main kinds of capacity sharing used in the models for the algorithmic analysis:

Workload Delegation
: Here, one SRM system hands over a certain amount of workload[4] and the management of its execution over to the management domain of another SRM system, underpinned by a previously negotiated SLA. This corresponds to most algorithmic scenarios of this work where jobs are exchanged between resource providers.

Resource Leasing
: Here, one SRM places resources available within its own domain under the exclusive capacity planning authority of another SRM system outside of his administrative control, for a time window previously agreed upon in an SLA.

In both cases, we refer to *the system that shares resources* as the **providing SRM**, and we refer to *the system that shares workload* as the **requesting SRM**.

8.2.1 The fedSRM Protocol

In order to realise the two kinds of capacity sharing in a common way within the architectural blueprint, we propose fedSRM, see Figure 8.2, a technology agnostic interaction model for federated capacity planning. A capacity planning session in fedSRM has the following steps:

1. The requesting SRM fetches available SLA templates from the providing scheduler. The SLA templates describe the possible service levels the

[4] For example, one or more jobs.

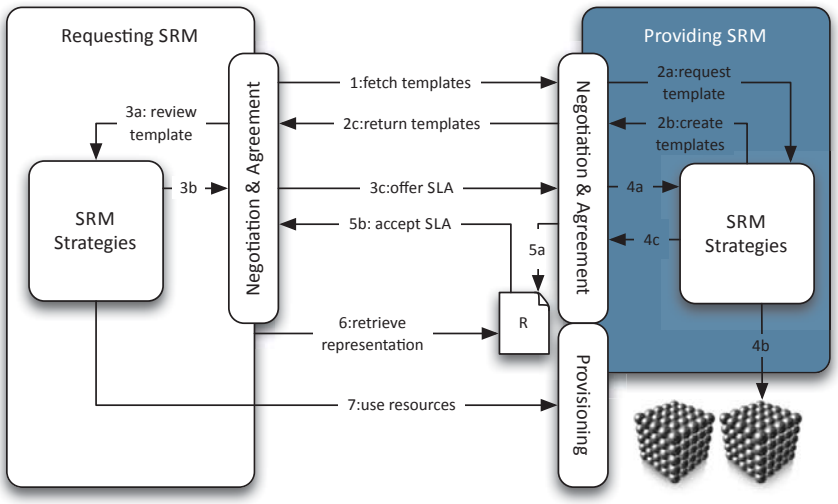


Fig. 8.2 *Schematic depiction of a capacity planning session in fedSRM, see also Birkenheuer et al (2011), for workload delegation and resource leasing. Steps* 8 *and* 9 *are not shown, as monitoring is a continuous activity performed by both sides, and decommissioning only involves internal operations on the providing SRM side. Also, standardisation is not depicted as a separate entity, as it is a cross-cutting concern and does affect all used protocols and interfaces.*

providing SRM can assure. For workload delegation, this can range from pre-populated application templates that only require the inclusion of input data paths, to abstract "job slots" that describe an execution period on the underlying infrastructure. For resource leasing, the service levels could map to certain machine configurations, e.g. "small", "medium", and "large". Rana and Ziegler (2010) give an overview of potential parameters and template design.

2. The providing SRM returns a set of templates describing services it can offer with respect to the current situation. These can now be concretised with additional requirements from the requesting SRM's side.
3. The requesting SRM reviews the offered templates (a, b) and offers a concretised version (depicting the aspired SLA) to the providing SRM (c).
4. The providing scheduler checks the general feasibility and adherence to creation constraints of the SLA (a) and, if correct, ensures that the requested resources will be provisioned until the earliest agreed time of availability (b); this step can also be performed asynchronously. In addition, the providing SRM creates a representation of the resource (c) for the requesting SRM.

5. The providing SRM accepts the proposed SLA template and makes a representation of it available to both agreement parties.
6. The requesting SRM retrieves the representation of the provided resource and examines its capabilities for further use. Depending on the kind of capacity sharing, the capabilities will differ: In case of workload delegation, it will provide submission and monitoring endpoints for jobs; in case of resource leasing, it will expose provisioning and management endpoints for machines.
7. After granting permissions by the providing SRM, the requesting SRM is able to use the delegated capacity like its own: In case of workload delegation, it submits jobs to the recently provisioned service endpoint, using the data management capabilities of the service interface for stage-in/out; in case of resource leasing, it binds the newly available resources to its underlying infrastructure and uses them as its own.
8. Both the requesting and the providing SRM monitor the use of the shared capacity to ensure that usage is within the negotiated and agreed terms of service.
9. After the agreed period of capacity usage, the providing SRM revokes permissions from the resources' representation. For workload delegation, this period ends after all jobs that have been part of the SLA were executed; for resource leasing, this happens after the agreed period of time.

fedSRM realises the different approaches discussed in Part II: The provisioning part is general enough to cover both workload-centric and resource-centric delegation models, and the negotiation part is independent of both. Furthermore, it is independent of the DCI architecture, as only the interaction between the SRM systems is directly affected. As such, we can integrate it into existing architectures without breaking compatibility, and this has been demonstrated for a number of SRM systems already, as shown by Birkenheuer et al (2009).

8.2.2 Technology Considerations

Since fedSRM is technology-agnostic but the realisation of production DCIs is not, we need to decide on the realisation of our interaction model: As protocols for both negotiation and provisioning need to be designed, we cannot stay on the architectural level. The decision on the technology used for realisation, however, turns out to be non-trivial, as there is a growing chasm between the architectural styles in academia and industry driven DCIs.

Many Grid architectures base on the idea of services that interact with each other (Foster et al, 2005) and adopt the paradigm of Service Oriented Architectures (SOAs), and therefore built upon Simple Object Access Protocol (SOAP) and heavyweight web services. Industry's Cloud offerings assume the exposed resources as the central idea and follow the paradigm of Resource Oriented Architectures (ROAs) instead, and therefore built upon

Representational State Transfer (REST) and lightweight web services. Neither of the two can be considered generally better, as both styles have their merits, see Thies and Vossen (2009), and both fit to various kinds of use cases. Here, we choose to prefer the latter over the former for the protocol realisation, for three reasons.

The first reason lies in the way the interaction model is designed: While it is easy to model the process of negotiation over both alternatives, the exposure of a resource's representation from the providing SRM to the requesting SRM implies a separate service in the SOA paradigm. As the representation approach is fundamental to the later use of the resource in steps 6 and 7, the ROA paradigm is the more natural choice.

The second reason lies in the use of capabilities: Since the resources change their behaviour depending on what they represent, each representation needs its own interface. Using SOAP, however, this is very hard to achieve, as the protocol provides no integrated support for this use case, and this would require the management of different service endpoints for different interfaces in a SOA based model. REST also has no explicit support for changing the behaviour of a resource, but since the endpoint is bound to the resource rather than the resource to a service, capability handling can be implemented on top of a ROA based model.

The third reason lies in the adoption: As with TCP and networking, the Hypertext Transfer Protocol (HTTP) has become the *lingua franca* for distributed computing APIs, and more and more modern web services build RESTful interfaces upon the ROA style[5], thereby embracing HTTP.

[5] This is particularly visible in the domain of Cloud computing, as discussed by Buyya et al (2009).

9

Protocols for Negotiating Authoritative Agreements[1]

Unless both sides win, no agreement can be permanent.
—James Earl "Jimmy" Carter

The business model of providing IT "as-a-Service" has become a new driver for the global market. With its vision of delivering services to the customer over simple, commoditised interfaces and its paradigm of paying per usage for the provided resources, it poses a main actor in the ongoing change of the IT landscape. Still, most interaction with respect to service guarantees is still done in a very traditional way, that is providing the customer with the Terms of Service (ToS) usually through a written document which are implicitly accepted with the usage of the service.

While this is necessary for legal reasons, it is obviously not sufficient, as the management of the ToS is disconnected from the service itself. Moreover, no electronic representation for the individual contract is available. Also, it is impossible to induce flexibility into the QoS terms on a per-customer basis without complex out-of-band communication (typically via special contracts between the legal representatives of both the consumer and the provider). With respect to the afore discussed fedSRM architecture, this obviously is a problem, as the dynamic character of the federation model requires on-the-spot negotiation and agreement between automated services.

Electronic SLAs provide a solution to this problem for many years already. These can be used to make an machine-readable agreement on the ToS while at the same time providing enough flexibility for both parties to change them to their needs. Here, we embed them into a RESTful protocol, called Agreements over REST (AoR) and define means for negotiating, creating, modifying, agreeing, and managing SLAs by leveraging the features of HTTP-based interaction. With respect to the obstacles depicted in Section 8.1, AoR closes the first gap towards a federated architecture blueprint for DCIs as denoted in Section 8.1.1, namely the negotiation and agreement of SLAs in

[1] Parts of this chapter have been presented at the 9th International Conference on Dependable, Autonomic, and Secure Computing in Sydney on December 14, 2011. The corresponding publication, see Blümel et al (2011), is available at `http://ieeexplore.ieee.org`.

a RESTful architecture. Thereby, AoR provides the first building block for the realisation of the fedSRM protocol.

9.1 Background

The IT Infrastructure Library (Cabinet Office, 2011) defines SLAs as

> "[...] agreement[s] between an IT service provider and a customer. The SLA describes the IT service, documents Service Level Targets, and specifies the responsibilities of the IT Service Provider and the Customer. [...]"

As such, they describe a contract between a consumer and a provider regarding a specific service and the quality of its delivery. This includes certain characteristics of the service and its performance, but also information about potential forfeits on non-fulfilment. With that, the service consumer has a binding description of the features he will get, and the necessary background information to monitor their performance. The service provider, in turn, has an overview of the warranted service level and the necessary background information to guarantee compliance.

The essential parts of an SLA (Sturm et al, 2000) are:

CONTRACTOR INFORMATION
: that is, verifiable identities of the signatories of the contract.

PERIOD OF VALIDITY
: including information about coming into effect and phasing out of the contract.

SCOPE
: in the sense of statements regarding the obligations and non-obligations of the service provider.

KEY PERFORMANCE INDICATORS
: that build the basis for deciding on fulfilment regarding the scope, including details on the means of measurement.

OBJECTIVES
: that define valid ranges for the key performance indicators; these are also called "Service Level Targets".

MEANS OF SANCTIONING
: if either of the signatories culpably violates the scope; these are also called "penalties".

STATEMENT OF CONSENT
: for both parties to put the agreement into effect.

The notion of scope is in fact very broad, even for the already narrowed-down use case of DCI federations: Obvious parameters with respect to fedSRM and the algorithms presented in Part II are the number of resources to be leased or a description of the job characteristics (such as number of requested machines m_j) to be delegated. In real-world scenarios, a natural candidate is the pricing of a lease or a delegation. However, more intricate considerations could be expressed as well: Data privacy law in many countries requires certain protection for stored information being exposed to such services, to the extent of determined physical locations requirements on the data centres. Here, we will focus on the protocol level only and leave the scoping as an exercise of creating an appropriate Domain Specific Language (DSL). For SRM, this has been already done by Schwiegelshohn and Yahyapour (2004), who describe a number of attributes that can be exploited by strategies.

The most widely adopted[2] product for electronic SLA management and negotiation is WS-Agreement (Ludwig et al, 2006). Its objective is to define a language and a protocol for advertising the capabilities of service providers and creating agreements based on templates and for monitoring agreement compliance at runtime. WS-Agreement addresses all parts of SLA description as part of its Extensible Markup Language (XML)-based document format, provides a state model for the agreement, service, and guarantee terms, and formulates a SOAP-based protocol for agreement negotiation and exchange. With its negotiation extension, see Wäldrich et al (2011), it supports multi-round negotiation and renegotiation capabilities next to simple "take it or leave it" semantics[3].

Because of the maturity of WS-Agreement, it is an ideal candidate for addressing the SLA negotiation and agreement aspect within a DCI federation, and we therefore choose to adopt it as the underlying technology for the fedSRM protocol realisation.

9.2 The Architecture of AoR

Since WS-Agreement was designed with the SOA paradigm in mind, it is necessary to make a number of adaptations in order to render its functionality over a REST based protocol. This is mainly because of four fundamental principles formulated by the ROA paradigm, see Fielding (2000):

Addressability of resources
: Every resource in a ROA based model must be addressable; that is, each resource has to carry a unique token that can be resolved to its location. In most realisations, this is achieved by assigning each resource a Uniform

[2] Others such as WSLA and SLAng enjoy some popularity as well, but do not support negotiation.

[3] The normative specification has been released by Andrieux et al (2007) for agreements and by Wäldrich et al (2011) for negotiation.

Resource Identifier (URI) and a resolver service, or a directly addressable Uniform Resource Locator (URL). The union of all URIs representing a resource in the ROA then builds its namespace hierarchy.

Statelessness of requests
: All requests performed on a resource must be stateless; that is, the client cannot be expected to save any application context. Resources are managed by the server only, and the client modifies the application state by sending messages to the resource. Every request must be self-contained, and the underlying operation must be consistent.

Uniform operations for resource manipulation
: All resources ("nouns") must expose a coherent set of operations ("verbs"). This ensures a high degree of reusability on the client side, and ensures that, in a distributed environment, communication is transparent for mediators in the system, as no application-specific knowledge is needed for optimisations such as caching and proxying.

Hypertext As The Engine Of Application State (HATEOAS)
: Every resource must provide references to related resources as part of its representation when delivered to the client, and the reference must describe the type of the related resources. Apart from well-known starting points, all other resources must be discoverable by only following such references, regardless of the used representation format.

Addressability, although not being directly supported by SOAP, has some resemblance in WS-Agreement already, as the protocol builds on the Web Service Resource Framework (WSRF)[4] which adds state to web services and implicitly introduces a model of directly addressable resources with means for the modification of their properties. Still, additional considerations regarding the transformation of service instances to resources need to be made, as REST assumes resource fragments, often called sub-resources to be addressable as well.

Statelessness is more problematic: WS-Agreement has the notion of "symmetric interaction" which requires the consumer side to implement its interfaces as well. In this particular flavour, both sides are eligible for updating the other's negotiation and agreement resources in order to reflect a synchronous state for the client and the server side. Obviously, this clashes with the second requirement and needs to be replaced by a mechanism conforming to the ROA paradigm. We therefore choose to omit the symmetric case in favour of interactive manipulation of negotiations and agreements.

Although the service operations exposed by WS-Agreement are complex, **uniformity** of simple "verbs" can be still achieved. This is mainly because it follows the *Factory/Instance* pattern (Gamma et al, 1994) for the management of negotiations and agreements. Every time it is necessary to create a

[4] As introduced by Foster et al (2005).

Table 9.1 *The impact of the CRUD operational model on the namespace hierarchy used in the context of AoR.*

		Namespace Hierarchy	
		Collection resources (e.g. `collection/`)	Entity resources (e.g. `collection/entity`)
CRUD verb	*Create*	Creation of a new (sub-)resource within this collection. The newly created entity is returned with the protocol reply message.	Creation of a new (sub-)resource within the parent collection, with the specified URI. The newly created entity is returned with the protocol reply message.
	Retrieve	Retrieval of all available (sub-)resources within this collection as a list of URIs, using a specific media type, as default. Other representations are still allowed.	Retrieval of the entity's representation according to the requested media type.
	Update	Not implemented.	Overwrite of the specified fragments' values with the data provided in the operation's payload.
	Delete	Deletion of this collection resource, including all embodied entity resources **or** removal of all members from the collection, depending on whether the collection resource is structural or not.	Deletion of the entity.

new instance of either of the two, the client calls the corresponding factory service which creates a matching WSRF resource and returns its identifier. We will reuse this concept for the ROA paradigm, as it maps nicely to the Create/Retrieve/Update/Delete (CRUD) model proposed for REST. To this end, we assume two general types of resources: *collection resources* and *entity resources*. The former will act as the ROA implementation of the factory service and the latter as the WSRF resource implementation. Table 9.1 shows the impact of the fundamental CRUD operations proposed by REST on the namespace hierarchy.

In order to adhere to the fourth principle, **HATEOAS**, the coupling between the resources must be loose, but discoverable. For collections of entities, this is already solved by the *Retrieve* operation described in Table 9.1. For the relations between entities, we assume links on the architecture level and discuss their realisation in the context of the resource and representation design.

With these principles at hand, we propose the overall architecture of AoR as depicted in Figure 9.1. There exist collections and entities for the management of negotiations and agreements, and as a means for bootstrapping the process, each of the two allow the handling of templates. The well-known entry points to the system are the collection resources, and the associations describe the relation structure. As discussed previously, this model allows us to follow the factory/service pattern in the AoR operational model as well: For both

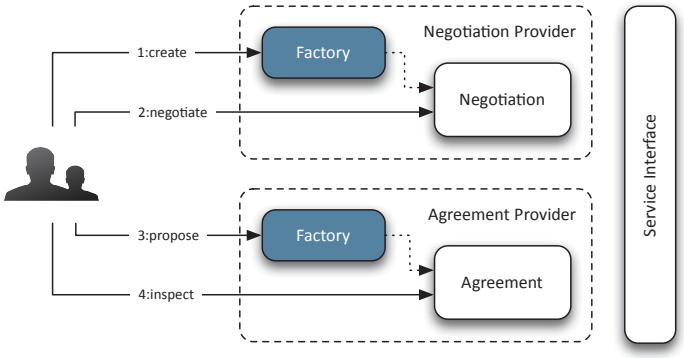


Fig. 9.1 *Schematic depiction of the general architecture of* AoR*, based on the corresponding patterns used in WS-Agreement. The three components are directly exposed to the consumer and can be used as a coherent system. Because of the* HATEOAS *principle, the different components can be implemented as distributed services, as the coupling between each is kept loose, by referrals through resolvable* URIs*.*

negotiation and agreement, a factory operation is executed first (steps 1 and 3) which yields an instance of the service instance to be manipulated (steps 2 and 4), and while the factory part binds to collection resources, the service part binds to entity resources.

All operations for manipulating them are mapped to one of the CRUD primitives described in Table 9.1. Following the REST paradigm, resources can be rendered in different ways, denoted by a media type. The conveyance of media type information is part of the transport protocol; in the case of HTTP, this is implemented through MIME content type definitions.

9.2.1 Resources

Resources build the foundation of every ROA. For AoR, we group them along the components described in the previous section, and adopt their internal structure from the XML document model of WS-Agreement.

The overall structure of resources and their relations are depicted in Figure 9.2, and their meaning and relationship is detailed in the following sections. In the UML model, specialised associations, i.e. aggregations and compositions, imply forward links from the whole to its parts and vice-versa, undirected associations model mutual links without hierarchical implications, and directed associations model navigable links without back references.

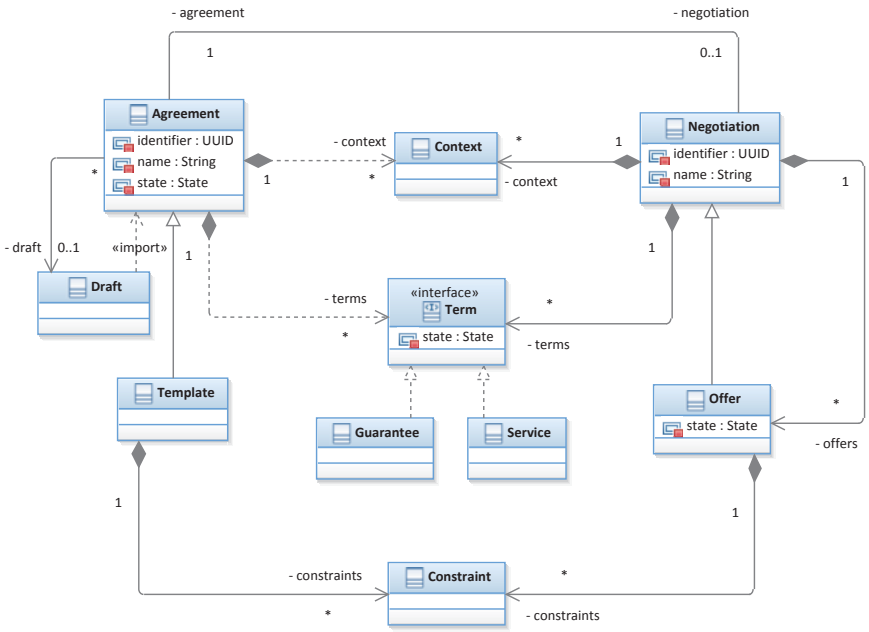


Fig. 9.2 *Structure and relation of collections and entities in the AoR resource model. The different types are grouped along the two main use cases, negotiation and agreement, and relationships between them indicate HATEOAS coupling.*

Agreement

Following the data model of WS-Agreement, all agreement-related resources comprise three basic parts:

METADATA
: Every agreement has an identifier and a name. We expose both as sub-resources of a `Metadata` resource. The identifier is immutable and must be unique, while the name is for readability purposes and can be changed by the client. The identifier is not necessarily equal to the agreement resource's URI and thus not necessarily needed for AoR, but kept for compatibility reasons.

CONTEXT
: The `Context` of an agreement denotes information about the signatories' identity and contract validity, i.e. expiration time. While the identity part will usually require the usage of digital signatures using X.509 certificates or similar means, the technology for this is specific to the rendering and therefore out of scope.

TERMS
: Here, the terms of agreement are exposed. In addition to the descriptions of services and guarantees for their quality, this also includes associated business values. The `Terms` resource is a collection resource, comprising a set of addressable `Term` resources, which are domain-specific.

Templates

The `Templates` collection contains the set of available templates that are generally available at the provider. Each individual template is modelled as a `Template` sub-resource and, in addition to the basic parts, may expose constraints for client customisation. These are kept in the `Constraints` sub-collection, and its elements are `Constraint` sub-resources.

Drafts

The `Drafts` collection provides a workspace for the client to interactively modify agreements before entry into force. Drafts have no resemblance in WS-Agreement, and were added to compensate for the lack of symmetric agreement handling in AoR: They allow the client to "work" on an SLA draft until ready for agreement, and eventually ask for consent by the server. This way, functionality of symmetric agreement handling is retained, while the client can be kept stateless. If the server cannot be trusted for managing the draft workspace, the HATEOAS principle allows it to reside on a trusted third party system as well. Every `Draft` contains only the basic parts, and a relation to the template it was derived from.

Agreements

The `Agreements` collection models the set of agreements that have come into effect, including operative, rejected, terminated and completed ones. Depending on the application, it may be necessary from a regulatory point of view to keep past agreements accessible; the decision on removal is therefore domain specific and out of scope. If the agreement is based on a template, its `Context` contains a relation to it.

The aforementioned state of an agreement is modelled by a corresponding `State` sub-resource. This unmodifiable collection contains two sub-resources, namely `Guarantees` and `Services`. The `State` resource exposes the current state along the state model depicted in Figure 9.3. There, an agreement can be Pending (the agreement is not yet in effect), Observed (the agreement is in effect), and Rejected (the agreement cannot come into effect), with corresponding transition states for the former two (e.g. for long-running operations). Moreover, an agreement can be Terminated (the agreement ended before fulfilment, with the consent of both parties) or Complete (the agreement ended after fulfilment).

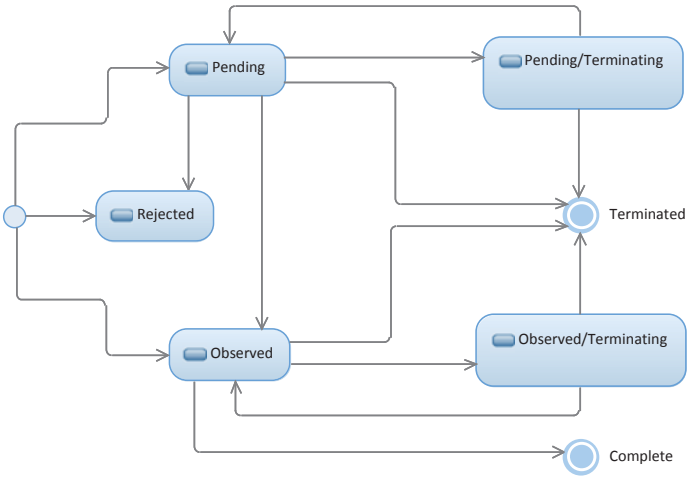


Fig. 9.3 *State machine for agreement resources in AoR resource model, following the original states in the WS-Agreement specification.*

Negotiation

Negotiations, as defined in the corresponding part of WS-Agreement, can be handled in a simpler way than agreements. We therefore distinguish only two basic resource types for the implementation in AoR. As a commonality, both resources feature an expiration date indicating that, after a certain point in time, either of the parties is not interested anymore in further pursuing the negotiation.

Offers

The `Offers` collection contains all offers that have been made by either party within the corresponding negotiation. Each `Offer` comprises the basic parts of an agreement, namely metadata, `context`, and `Terms`, and holds a number of domain-specific negotiation `Constraints`, see Section 9.2.1. In addition, it relates to the underlying `Template`, if appropriate.

Every offer has a state within its corresponding state model, see Figure 9.4. The states have the following semantics:

ADVISORY
: indicates that further negotiation is necessary before the creator of the offer is willing to accept the current agreement draft.

SOLICITED
: indicates that the creator of the offer wishes to converge toward an agreement. Any counter-offer to a SOLICITED offer must be either ACCEPTABLE or REJECTED.

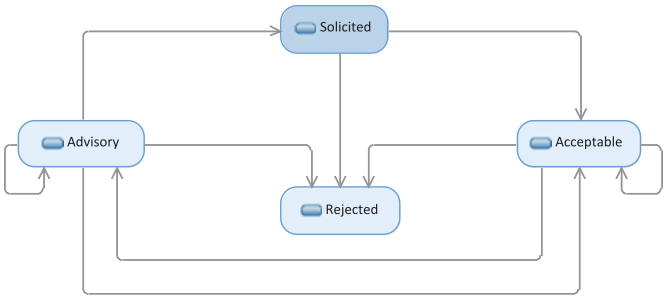


Fig. 9.4 *State machine for offer resources in AoR resource model, following the original states in the WS-Agreement specification.*

ACCEPTABLE
: indicates that the creator of the offer would be willing to accept the current agreement draft without further changes.

REJECTED
: indicates that the current offer/counter-offer path in the negotiation must not be pursued any more by either of the parties, as an agreement is unlikely to be reached.

Negotiations

The `Negotiations` collection models the set of negotiations that have come into effect, see Section 9.2.1. Each negotiation comprises `Metadata` and `Context` and exposes relations to the negotiation participants[5]. In case of a renegotiation, a relation to the original `Agreement` resource is also included. Finally, a negotiation resource keeps an `Offers` sub-resource containing all offers made so far.

9.2.2 Representation

So far, the structure of the resources involved in AoR has been defined, but their concrete representation is still missing. While for resources such as `Context` or `Metadata`, this is either not possible (since they are domain specific) or not necessary (as they comprise a primitive data type), it is crucial for others.

In general, we assume content negotiation for requests and responses within the AoR protocol. This allows us to represent all collection resources by exposing them as lists of URIs, using the `text/uri-list` Multipurpose Internet Mail Extensions (MIME) media type as described in RFC 2483. Besides

[5] In the WS-Agreement concept, this is the *Negotiation Initiator* and *Negotiation Responder*.

this, we expose the top-level resources available in WS-Agreement, namely `Template`, `Agreement`, and `Negotiation`, as their XML representation as defined in the original specification, using `application/xml` as media type.

For the domain specific parts, we do not make any assumptions on their concrete representation. This is not necessary anyway, as content negotiation allows the client to request those representation formats that he understands. Naturally, this poses a problem with relations between domain-specific resources. To address this issue, we leverage the HTTP *Link* header as specified in RFC 5988[6]: With that, we can render relations between resources independently from their representation format. This way, it is possible to have service references within domain specific `Term`s as required by the original WS-Agreement specification. The expirations required for the negotiation part are rendered through the HTTP *Expires* header.

As a side effect of the content negotiation model, AoR can support arbitrary representation types without any changes to the resource model or interaction protocol. For example, the state of an agreement can, besides its primitive representation as an enumeration item, be exposed as a history of changes by implementing support for the Atom syndication format as published in RFC 4287. Another use case is to provide rich web based views through Hypertext Markup Language (HTML) representations of certain resources; in conjunction with client side technologies such as AJAX (see Eernisse, 2006), it is even possible to deliver browser clients for interaction.

9.3 Rendering of the Protocol

Like the vast majority of ROAs, we follow the recommendation of Fielding (2000) and render our RESTful protocol over HTTP (RFC 2616). This enables us to directly translate the CRUD verbs described in Table 9.1 into HTTP requests, namely `POST`, `GET`, `PUT`, and `DELETE`. To indicate the success or failure of requests, we reuse HTTP status codes.

As an optimisation, we further allow the `HEAD` request, which only returns the header data. In conjunction with the afore discussed Link header approach, this allows clients to discover the resource space along the HATEOAS principle without having to retrieve a full resource representation on each request.

It is noteworthy that this feature must be used with care: Proxies and other intermediates are allowed to discard HTTP headers, if necessary[7], and the size of the full HTTP header is limited. Most implementations allow a total length between 8 Kbytes and 64 Kbytes, but since the number of headers used for the linking model is limited to a few per resource, this concern is secondary.

In the following, we discuss the different use cases for negotiation and agreement and illustrate the protocol for conducting them with examples of

[6] All noted Requests for Comments (RFCs) are published by the IETF and are available at `http://www.ietf.org/rfc.html`.

[7] Although in practice this is seldom the case.

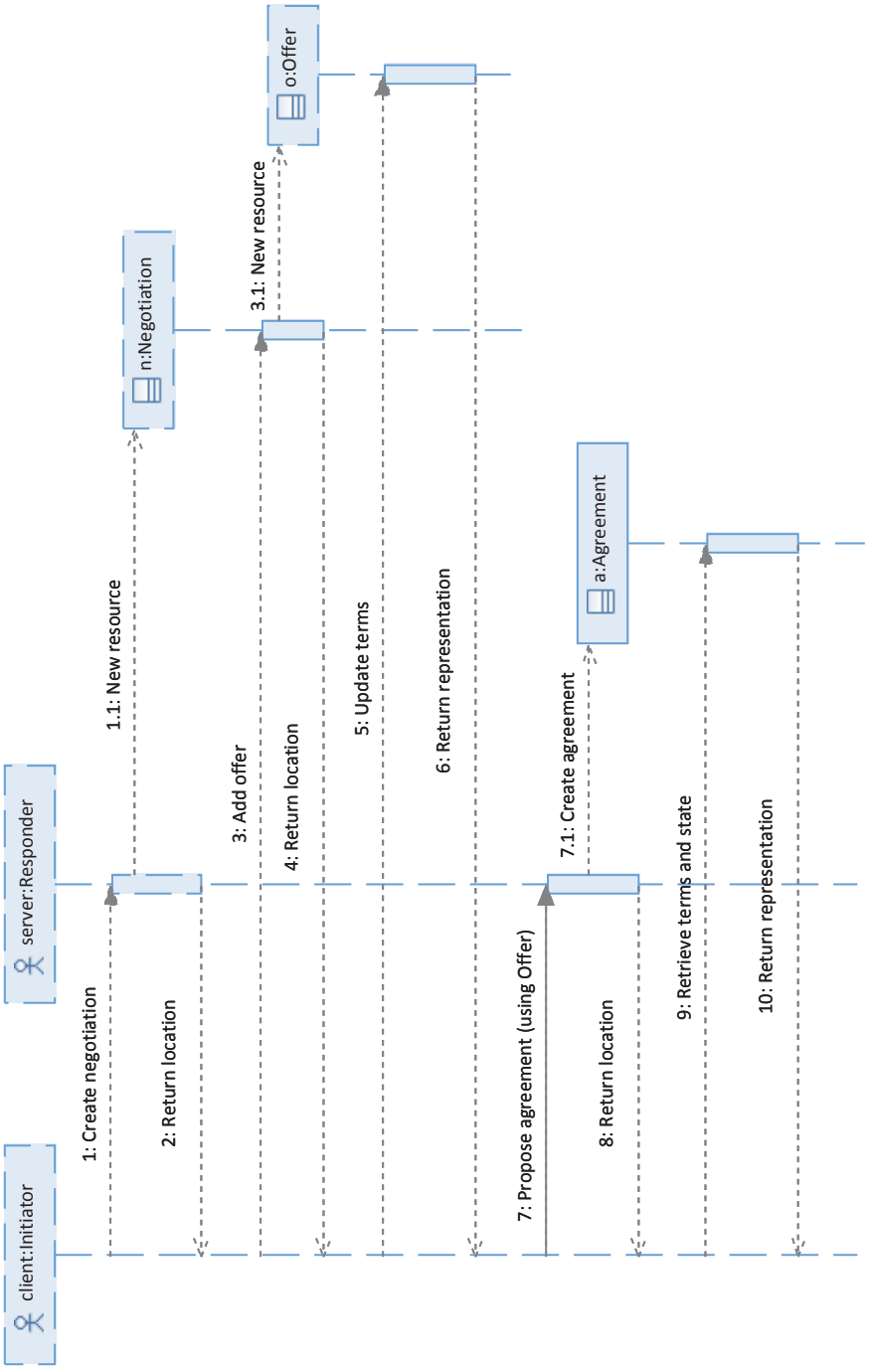


Fig. 9.5 *Schematic depiction of a negotiation and agreement session's protocol steps in AoR. With the client side being stateless, a session can be suspended and resumed at any time, thereby allowing for parallel negotiations. With the creation (and acceptance) of an* `Agreement` *resource in steps* 7 *and* 8*, the SLA is put into effect and binds both signatories.*

requests and responses in HTTP. For the content type of domain specific fragments, we use the Java Script Object Notation (JSON), see RFC 4627, with its corresponding media type `application/json`. An abstract view on the protocol for the whole process described in the following is shown in Figure 9.5; the steps in the protocol are detailed in the following.

9.3.1 Negotiation

To start a negotiation session, the client sends the server a request to create a new `Negotiation` resource (step 1 in Figure 9.5). Following the model described in Table 9.1, this is done via `POST`ing to the `Negotiations` collection, with the corresponding *Link* and *Expires* headers:

```
1  > POST /negotiations/ HTTP/1.1
2  > Link: <http://client.example.com>; rel="aor-initiator"
3  > Link: <http://server.example.com>; rel="aor-responder"
4  > Link: <http://server.example.com/templates/de0f11>; rel="aor-template"
5  > Expires: Sun, 24 Jun 2012 16:00:00 GMT

7  < HTTP/1.1 201 Created
8  < Host: server.example.com
9  < Location: /negotiations/6d73b2
```

In the background, the server will create an initial offer as a sub-resource of the newly created negotiation resource (step 1.1 in Figure 9.5); this offer is built by using the provided `aor-template` link the client has used to indicate on which template to base the negotiation (l. 4) and expires after a certain time period (l. 5). If the `POST`ed negotiation request refers to an existing agreement, i.e. for re-negotiation purposes, the client adds an additional link header with `rel=aor-agreement` that points to the corresponding `Agreement` resource.

The server is free to defer the decision on whether to negotiate. In that case, the reply carries the status code `202 Accepted` instead. Either way, the server returns the location (l. 9) of the newly created resource (step 2 in Figure 9.5). When the client then performs a `GET` on the `Location` target, the server replies with `304 Not Modified` until the decision has been made. In the negative case, the server replies with `403 Forbidden`.

The client can also list the negotiations (l. 1) that have come into effect. As this is a request to a collection resource, the server will reply with a list of negotiation resources (ll. 7–9) visible[8] to this client:

```
1  > GET /negotiations/ HTTP/1.1

3  < HTTP/1.1 200 OK
4  < Host: server.example.com
5  < Content-type: text/uri-list
6  <
7  < http://server.example.com/negotiations/6d73b2
8  < http://server.example.com/negotiations/07dfbe
9  < http://server.example.com/negotiations/1edd2e
```

[8] In most cases, this list will not comprise all negotiations, but only those the client is supposed to see because of Access Control Lists (ACLs) or multi-tenancy.

Now the client can examine a specific negotiation. Again, he will run a `GET` operation on one of the locations indicated by the previous reply (ll. 1–2):

```
1  > GET /negotiations/07dfbe HTTP/1.1
2  > Accept: application/json

4  < HTTP/1.1 200 OK
5  < Host: server.example.com
6  < Content-type: application/json
7  < Link: <http://client.example.com>; rel="aor-initiator"
8  < Link: <http://server.example.com>; rel="aor-responder"
9  < Link: <http://server.example.com/templates/de0f11d>; rel="aor-template"
10 < Link </negotiations/07dfbe/terms/>; rel="aor-terms"
11 < Link </negotiations/07dfbe/offers/>; rel="aor-offers"
12 < Link </negotiations/07dfbe/offers/e115e8>; rel="current"
13 < Expires: Sun, 24 Jun 2012 16:00:00 GMT
14 < Cache-control: no-cache
15 <
16 < {
17 <     "metadata" : {
18 <         "identifier" : "07dfbe",
19 <         "name"       : "Example Negotiation"
20 <     },
21 >     "context" : {
22 <         "initiator" : "...",
23 <         "responder" : "..."
24 <     }
25 < }
```

Here, the representation contains a flattened rendering of the `Metadata` and `Context` sub-resources as part of the response body (ll. 16ff.). It is noteworthy that a *Cache-control* header (l. 14) is returned indicating that the reply must not be cached. The server provides this value in spite of the *Expires* header (l. 13), because AoR extends the semantics of the latter to the resource rather than only the content of the reply. The server further indicates that, besides the original information submitted by the client on creation of the agreement (ll. 7–10), a collection resource `offers/` exists (l. 11), and the current (latest) offer can be found under the provided URI (l. 12).

By listing the latter, the client can fetch the URIs of the corresponding offers and query them individually:

```
1  > GET /negotiations/07dfbe/offers/cce50a HTTP/1.1
2  > Accept: application/json

4  < HTTP/1.1 200 OK
5  < Host: server.example.com
6  < Content-type: application/json
7  < Link </negotiations/07dfbe/offers/c09fc2>; rel="prev"
8  < Link </negotiations/07dfbe/offers/e716db>; rel="next"
9  < Link </negotiations/07dfbe/offers/e115e8>; rel="next"
10 < Expires: Sun, 24 Jun 2012 16:00:00 GMT
11 < Cache-control: "no-cache"
12 <
13 < {
14 <     "metadata" : {
15 <         "identifier" : "cce50a",
16 <         "name"       : "Example Offer"
17 <     },
18 >     "context" : {
19 <         "initiator" : "...",
20 <         "responder" : "..."
21 <     },
```

```
22  <     "state": "ADVISORY",
23  <     "terms": {
24  <         ...
25  <     }
26  < }
```

The server response in this case again contains a flattened representation of the offer and its sub-resources, including the current state of the negotiation following the state model described in Section 9.2.1. More importantly, it expresses ancestry relations to other offers: The *Link* header with the *prev* relation points to an offer this one is based on, and the *Link* headers with a *next* relation points to offers based on this one[9]. With that, clients are able to discover the complete offer tree.

In order to proceed with the negotiation process, the client can create a new offer (step 3 in Figure 9.5) or update an existing one's terms (step 4 in Figure 9.5); here, basic `POST` and `PUT` operations as described in Table 9.1 are used. In any case, a representation of the modified offer resource is returned (step 6 in Figure 9.5).

9.3.2 Agreement

In order to eventually converge to an agreement, clients can either undertake the negotiation process as described in Section 9.3.1, or directly use available standard offerings. These are provided as templates which describe commoditised service levels the server is willing to provide to clients without a convergence process.

In this case, the client starts with selecting an agreement template from the list of available templates (ll. 1, 11):

```
1   > GET /templates/ HTTP/1.1

3   < HTTP/1.1 200 OK
4   < Host: server.example.com
5   < Content-type: text/uri-list
6   <
7   < http://server.example.com/templates/4102e8
8   < http://server.example.com/templates/28ff42
9   < http://server.example.com/templates/df61d7

11  > GET /templates/df61d7 HTTP/1.1
12  > Accept: application/wsag+xml

14  < HTTP/1.1 200 OK
15  < Host: server.example.com
16  < Content-type: application/wsag+xml
17  <
18  < <?xml version="1.0" encoding="UTF-8"?>
19  < <Template
20  <   xmlns="http://schemas.ogf.org/graap/ws-agreement/2011/03"
21  <   TemplateId="df61d7">
22  <   <Name>An Example Template</Name>
23  <   <Context>...</Context>
```

[9] To ensure that the relations are dependable, offers cannot be modified after creation.

```
24  <   ...
25  < </Template>
```

At this point, WS-Agreement requires a "take-it-or-leave-it" approach: The client sends a request to the server containing the proposed agreement, and the server either accepts or rejects this proposal. This is mainly due to the SOA paradigm, as the agreement procedure is conducted through a service model. AoR is resource-oriented instead, and this allows us to make the creation of the agreement interactive. As such, the server provides the client with an additional collection resource called `drafts` with no special properties under which he is allowed to create (ll. 1–2) and stepwise modify (ll. 8–13) an agreement draft until ready for the binding step:

```
1   > POST /drafts/ HTTP/1.1
2   > Link </templates/df61d7>; rel="aor-template"

4   < HTTP/1.1 201 Created
5   < Host: server.example.com
6   < Location: /drafts/2f79ac

8   > POST /drafts/2f79ac/terms/ HTTP/1.1
9   > Content-type: application/json
10  >
11  > {
12  >      "availability": "0.99"
13  > }

15  < HTTP/1.1 409 Conflict
16  < Host: server.example.com
```

In this example, the creation of the agreement draft succeeded (ll. 4–6), but the addition of a term failed due to a conflict with statements in the `Constraints` collection resource, which is indicated by the corresponding HTTP status code (ll. 15–16). In contrast to the negotiation process, the server does not propose any alternatives, but only indicates that certain modifications of the template are not acceptable. Most importantly, the server is not allowed to modify the draft resource independently; only the client is allowed to make changes. As such, the `drafts` collection should not be seen as a simple replacement for negotiation, but rather as an orthogonal functionality that provides interactive creation.

Regardless of whether starting from a negotiation that has converged towards an acceptable offer or a draft that is sufficiently concretised from both the client and the server perspective, the final step of **binding** must be performed (step 7 in Figure 9.5). To this end, the client indicates to the server that he wishes to make a binding agreement (l. 1):

```
1   > POST /agreements/ HTTP/1.1
2   > Link </drafts/2f79ac>; rel="aor-draft"

4   < HTTP/1.1 202 Accepted
5   < Host: server.example.com
6   < Location: /agreements/a5f24a
```

In the depicted case, the client has referred to the previous draft for creation (l. 2), and the server has accepted the agreement for evaluation, but has not

finally decided upon it (l. 4); the `State` of the agreement is therefore PENDING with respect to the model in Figure 9.3, and the agreement is considered binding for the client. If the server replies with `201 Created` instead, the agreement is directly binding for both parties (step 7 in Figure 9.5) and the agreement state is OBSERVED. In either case, the reply contains a location (l. 6 and step 8 in Figure 9.5). If the proposed agreement is not acceptable[10], the server replies with `403 Forbidden`. For starting from a converged offer (i.e. an offer in the SOLICITED state) as part of a negotiation, we simply change the *Link* header from `rel="aor-draft"` to `rel="aor-offer"` and refer to the corresponding resource (l. 2).

Note that in both cases the request body is empty, as we refer to an already existing draft or offer. If the client has stored available agreement proposals locally (e.g. from previous requests), he can also directly ask for binding:

```
1  > POST /agreements/ HTTP/1.1
2  > Content-type: application/wsag+xml
3  >
4  > <Agreement
5  >   xmlns="http://schemas.ogf.org/graap/ws-agreement/2011/03"
6  >   AgreementId="c504d9">
7  >   <Name>An Example Agreement</Name>
8  >   <Context>...</Context>
9  >   ...
10 > </Template>

12 < HTTP/1.1 201 Created
13 < Host: server.example.com
14 < Location: /agreements/c504d9
```

Again, the agreement is binding for both sides, its state is OBSERVED, and the terms can be examined by corresponding `GET` operations (steps 9 an 10 in Figure 9.5).

Terminating an agreement corresponds to requesting a transition of its `State` resource to TERMINATED, using a `PUT` request. If the request is **accepted**, the response is `200 OK` or `204 No Content`. Otherwise, if **rejected**, this is expressed with `403 Forbidden`; a domain specific reason can be supplied as message body. The decision can also be deferred, resulting in an immediate state transition to OBSERVEDANDTERMINATING or PENDINGANDTERMINATING, a `202 Accepted` response, and a later transition to TERMINATED or back to the original state.

Termination requests for REJECTED or COMPLETE agreements are rejected with `409 Conflict`; likewise, any transitions to other states than TERMINATED are rejected with `403 Forbidden`.

9.4 Evaluation

For the AoR, we do a qualitative evaluation only, and focus on two main questions: to which extent the REST approach introduces limitations compared

[10] For example, because mandatory fields are missing, or the server is not able to fulfil the service terms.

to the original protocol, and how the new protocol integrates with already existing RESTful systems.

9.4.1 Limitations

Although the overall interaction model of WS-Agreement can be properly translated to the ROA paradigm, we did not consider two important aspects.

The first is related to the missing feedback channel towards the client: With the HTTP primitives only, a push-based model for notifying the client on changes on the server side cannot be realised. One option to workaround this issue is the use of HTTP polling: Since `HEAD` operations are cheap, the client can repeatedly ask whether the resource has changed, and fetch the full representation only if this is the case. This approach is very common in RESTful systems, and well supported by the HTTP protocol through the *ETag* header. Furthermore, it can be supported by feed formats such as Atom, see RFC 4287, which allow the server to additionally indicate what has changed.

The second is related to the requirement of electronic signatures for legally binding SLAs. While SOAs support this natively through XML and SOAP[11], RESTful systems have no corresponding mechanism integrated. It is however possible to detach the signatures from the document itself, store them as separate resources on the server (or a trusted third party), and link back to the original agreement. Still, no standard protocols and formats are available to do this, and implementation support is very limited.

9.4.2 Integration

Since SLAs do not exist independently of the service they describe, it is not only reasonable to expect an SLA instance to contain a reference to the service, but also vice-versa. This integration is, thanks to the resource-centric model, very simple: The service provides a back reference to the SLA as part of its response message to the client. For this, we have two options. First, we can use the *Link* header model:

```
1  > GET /service/bd1f82 HTTP/1.1
2  > Content-type: text/html
3  >
4  < HTTP/1.1 200 OK
5  < Host: server.example.com
6  < Link: </agreements/c504d9>; rel="aor-agreement"
7  <
8  < ...
```

Here, the server indicates that a certain service instance is covered by the agreement referred to within the link. The other option is a virtual subresource of the service, e.g. `/service/bd1f82/agreement`, which has no content by itself but produces a `301 Moved Permanently` response, redirecting the client to the original agreement. While the first approach is more concise,

[11] Using WS-Security as described by Naedele (2003).

the second is easier to implement: The link header approach requires to incorporate this information into the *ETag* computation of the resource itself to be cacheable; the sub-resource approach only requires the handling of a virtual URI in the resource namespace, and caching is handled transparently through HTTP.

9.5 Related Work

A number of projects employ SLAs in their architecture: PHOSPHORUS (Eickermann et al, 2007) makes use of SLAs to manage co-allocation of resources from the compute and network domain for traditional HPC job scenarios which require dedicated QoS parameters. BREIN (Schubert et al, 2009) aims to support outsourcing for an increased competitiveness of SMEs by enhancing classical IaaS scenarios with a standards-based environment for eBusiness, including SLAs, to allow for provision and sale of services.

Regarding SLA languages and protocols, only WS-Agreement is currently under active development and uses a broad approach to SLAs. Other languages like SLAng (Lamanna et al, 2003) and WSLA have been developed as well, but they are not well-established and support therefore ceased. As such, most projects and efforts (including the ones listed above) base on WS-Agreement, however focus on working the business requirements rather than introducing new ideas to the protocol level.

Only little effort has been put into the RESTful world so far besides the work from Kübert et al (2011), who propose a direct transformation of WS-Agreement to a REST-based interface. They describe how the entities in the WS-Agreement specification can be mapped to a RESTful interface; however their approach does not incorporate the protocol part for interactive agreement creation, and does not address negotiation.

10

Standards for Differentiated Resource Provisioning[1]

I do very few standards. Hardly any. I do a couple of Duke Ellington tunes that are well known.

—Mose Allison

With the mechanisms for negotiating agreements between the participants in a DCI, the first part of fedSRM is largely set. Still, a unified interface that allows to expose resources from the service provider side, and provision them from the service consumer side, is missing.

Today's cloud ecosystem impressively proves the need for an open means of describing, provisioning, and accessing resources, as the requirement for dynamically adding them to a federated environment raises. This is mainly because we consider a young market: Offerings change quickly and often, and the risk of providers being ephemeral must be mitigated by allowing consumers to switch providers. Several projects aim to provide a single API for the plethora of proprietary service provider interfaces, and the most popular of ones follow a proxy pattern approach.

The increasing number of providers on the market makes it reasonable to question this approach: Besides the aspects if latency and cost of ownership, see Section 8.1.2, issues around intellectual property arise, and the coupling of interface definition and implementation is disputable. Due to the current lack of serious alternatives—with the notable exception of DMTF's early Cloud Infrastructure Management Interface (CIMI) draft (see `http://dmtf.org/standards/cloud` for an overview of the working group's activity and a non-normative interface primer)—we make an attempt towards an open standard for provisioning in "as-a-Service" environments called Open Cloud Computing Interface (OCCI). With that, we close the final gap towards a federated architecture blueprint for DCIs as denoted in Section 8.1.2: the provisioning of delegated workload and leased resources. While the latter is an inherent capability of the original OCCI standard, we demonstrated the former through the rendering of a job submission over the OCCI protocol and interface.

[1] Parts of this chapter are based on an article in the IEEE Internet Computing in July 2012. The corresponding publication, see Edmonds et al (2012), is available at `http://ieeexplore.ieee.org`.

10.1 The Architecture of OCCI

Any attempt to standardise interfaces for commercial IT service offerings has to cope with the field of tension between unification and differentiation: While from the standardisation point of view, the agreed interface should be alike for all service providers, each vendor naturally strives for a way to expose his unique features and ensure customer retention.

With the OCCI, we aim to address both aspects by providing a unified API while at the same time offering a means for extensible, discoverable capabilities. To achieve this, we stipulate three main design goals:

DISCOVERABILITY
: Service consumers can query the service provider to find out what capabilities are available. The information is self-describing and complete. If the service consumer is a broker, it can request that multiple service providers describe what is offered and then choose from among them.

EXTENSIBILITY
: Because Cloud computing spans a broad set of offerings, from infrastructure to applications, the OCCI specification allows service providers to extend the API to cater their specific needs while allowing service consumers to discover these extensions at the same time.

MODULARITY
: Because of its extensibility, OCCI must be modular. Indeed, even the different components of OCCI should not unnecessarily mandate the use of unrelated parts. Instead, each component should be usable individually, or be replaced as the situation requires.

The former two requirements necessitate a general description for resources and capabilities that features extension points. The latter requirement imposes structure in terms of the separation of concerns. We therefore start with the creation of an API-independent model.

10.1.1 Core Model

With the core model for OCCI (Behrens et al, 2011a) we introduce a general means of handling general resources, providing semantics for defining the type of a given entity, interdependencies in between different entities, and operating characteristics on them.

The core model can be separated into two distinct parts, namely the meta model and the base types. The different components of the core model as shown in Figure 10.1 are described in the following.

Meta Model

In order to address the requirement of extensibility, we introduce a type system as the meta model. This ensures that each resource in the OCCI namespace

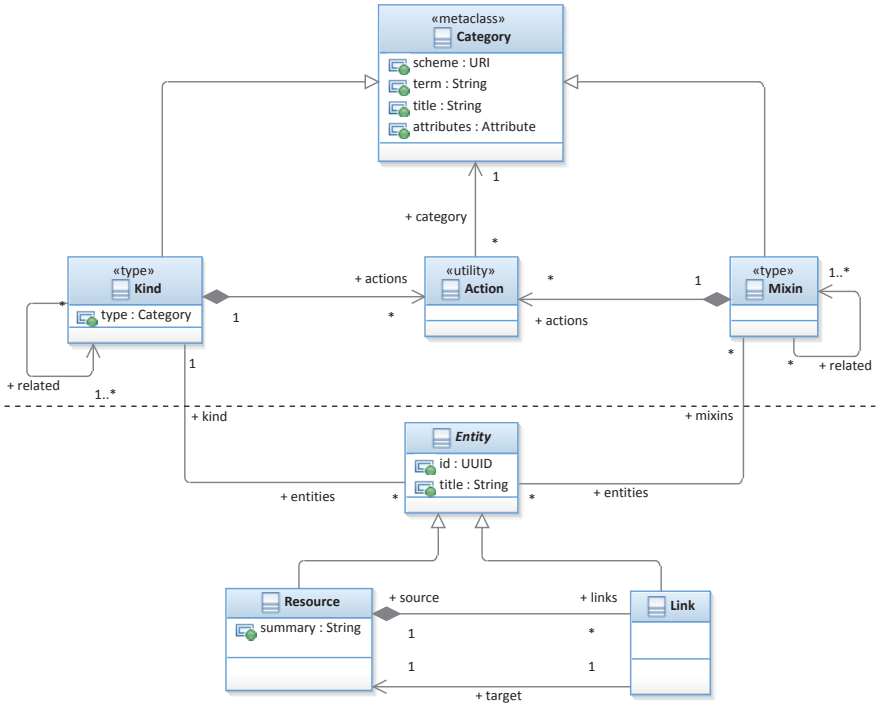


Fig. 10.1 *Relationships of the different core components in the Open Cloud Computing Interface. Note the separation into base and meta model below and above the dotted line. The former describes the base types of the OCCI type system, introducing the base* `Entity` *class, whereas the latter comprises the descriptive framework within the model that allows introspection into model instances.*

has a common way of defining its capabilities (attributes, actions, and so on). The foundation for these descriptive types is the *Category*. Categories are freely definable and uniquely identifiable using a term and a scheme, e.g. `http://example.com/category/scheme#term`. They can relate to each other and thereby define a hierarchy.

Using categories, however, we only define a general type and build the basis for discoverability, but do not directly impose structure to resources. For this purpose, we introduce the first specialisation of it: the *Kind*. Each resource instance will have one Kind[2] that is defined by a category in the OCCI model. A Kind is immutable and specifies the characteristics of a resource, which include its location in the hierarchy, assigned attributes, and applicable actions. Kinds are mandatory and singular to each resource. This way, we

[2] In that sense, a Kind is very similar to the `Class` type in the Java™ language.

enforce a base classification of each instance in the OCCI system and ensure static capabilities at design time.

With Kinds only, it is however hard to let resources assume certain characteristics during runtime. We address this by introducing an additional component into the meta model: the *mixin*. A mixin is a way of dynamically manipulating a resource instance's capabilities. We can think of mixins as a way to inherit additional capabilities by composition as described by Gamma et al (1994). This concept can be found in many modern programming languages[3] and allows for bundling reusable features. Each OCCI resource can have zero or more mixins. An OCCI mixin also has a set of capabilities such as its location in the URI hierarchy, applicable actions, and attributes. This means that a resource instance's capabilities can be altered over time. We also leverage mixins as a templating mechanism in OCCI by defining templates as mixins that can be applied to resource instances, which then assume the template's characteristics. Finally, we use mixins in OCCI to tag resource instances for enabling folksonomic organisation (see Peters and Stock, 2007).

With the ability to describe resources comes the need for manipulating them. In the meta model, we allow for two interaction mechanisms: one resource-centric, and one operation-centric. The resource-centric CRUD fundamentals, as introduced with AoR in Section 9.2, are complemented by their operation-centric counterparts, the *Actions*, which denote complex operations on a resource that are difficult to model with the resource-centric primitives. In contrast to the CRUD operations, which are an integral part of every resource, we make Actions artefacts of their own; thus, they expose a Category and are attached to either a Kind or a Mixin. This way, we allow Actions to have specific arguments that are defined as part of their Categories, while CRUD operations are limited to the resource's representation as their single parameter.

Base Types

In order to build a starting point for the development of new resource types within the core model, we define two fundamental types in OCCI, namely *Resource* and *Link*. Both inherit from the abstract *Entity* that serves as a foundation type to enforce the correct usage of the meta model.

Resource represents the resources that are exposed to the service consumer. Essentially, they are abstractions of the service provider's backend model exposed through the underlying resource management framework. Keeping in mind that the core model imposes inheritance through relation, any resource sub-type exposes at least three Kinds: its own, one from Resource, and one from Entity. *Link* allows to create a typed association between resource entities. A link is a directed association between two resources, but because

[3] Scala, Ruby, and Objective-C support mixins out of the box.

we derive it from an entity, it is also a resource itself and, as such, can be manipulated in the same way.

A side effect of the inheritance from Entity is that we are able to group resources of the same Kind. In the OCCI core model, we leverage this by-product as an integral feature: Entity instances associated with the same Kind automatically form a collection, and each Kind identifies a collection consisting of all of its Entity instances. Collections are navigable and support the retrieval of a specific item, any subset, or all entries.

Discovery Model

Through the *query interface*, we allow service consumers to find all categories that are usable in a service provider's namespace. It exposes all registered categories and their corresponding hierarchy and describes how each individual category is composed (such as its capabilities and so on).

Because of this, Kinds and Categories that are supported by a particular service provider can be discovered. By examining them, a client is enabled to deduce the Resource and Link types a service provider offers, including domain-specific extensions, the attributes they expose, the invocable operations on them[4], and additional mixins a Resource or Link can currently assume. Categories in the query interface can of course be filtered and searched for.

10.1.2 Infrastructure Extension

With the infrastructure model (Behrens et al, 2011b) we define an extension to the core model that describes entities in the IaaS domain. In the context of the core model, it introduces a set of resources with their capabilities and defines how to link those resources together if required.

The infrastructure specification describes three kinds, which can be used to create instances of compute, storage, and network resources. Figure 10.2 shows how these entities relate to their corresponding parts in the core model. In the extension, we further specify links between resources, such as a compute instance that links to a layer 2 network router[5]. Another example is attaching a storage resource—for example, a database, or block device. This approach is also helpful when mashing up with other standards: For instance, OCCI accomplishes interoperability with the Cloud Data Management Interface (CDMI) through its linking mechanism as described by Edmonds et al (2011a).

Basing on the core model, we allow service consumers to apply mixins to some of the described infrastructural entities. We mainly use this feature with

[4] I.e. the Actions related to it through its Kinds or Mixins

[5] Following the Open Systems Interconnection Reference Model (OSI), see Zimmermann (1980).

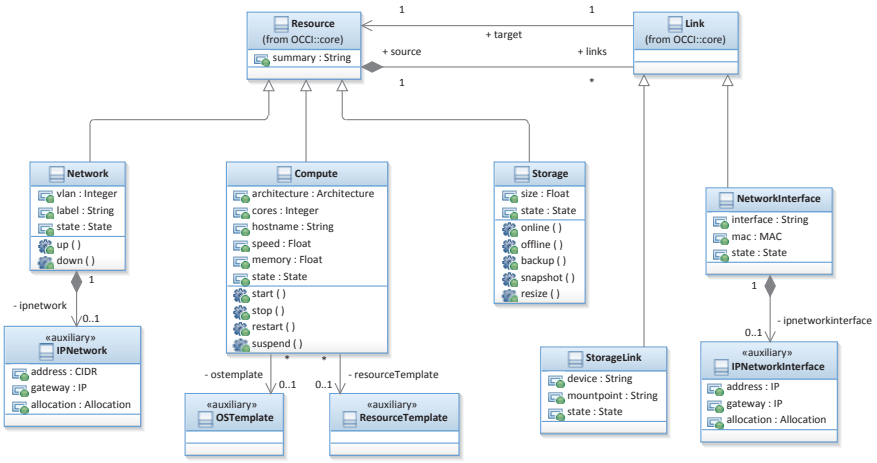


Fig. 10.2 *Relationships of the infrastructure components in the Open Cloud Computing Interface. On the left side, the types denote a service provider's physical resources (such as a compute resource instance); on the right side, ephemeral entities are described. (for example, a storage mount point).*

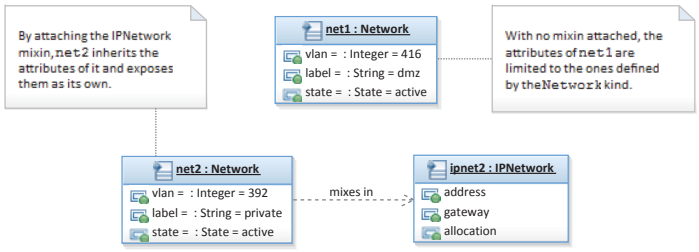


Fig. 10.3 *Relationships between different mixins in the infrastructure extension of the Open Cloud Computing Interface. The upper resource,* `net1`*, depicts a physical network without OSI layer 3 capabilities. The lower resource,* `net2`*, attaches an* IPNetwork *mixin that adds these capabilities — in this case, a network address, a gateway, and an allocation.*

network-related entities in order to give them OSI layer 3 and 4 networking capabilities[6]; this concept is illustrated in Figure 10.3. By dynamically adding an *IPNetwork* mixin to the *Network* resource, we add extra capabilities to the resource instance. Note we can bind multiple mixins to a single resource,

[6] If we stayed on the resource level without linking, such entities would represent only layer 2 networking entities, which are not particularly useful for internetworking.

and—because they are entities themselves—we can add mixins to links as well.

The categories hierarchy ensures that only mixins relevant to the resource instance in question can be added. This way, service consumers and providers are limited to adding a mixin to a resource instance that is in the mixin's hierarchy; we thereby ensure that applying networking mixins to a *StorageLink* is not possible. Because the hierarchy itself is not limited, mixins can however build on other mixins.

10.2 Delineation from the Proxy Approach

OCCI resides on the boundary between the resource management framework and the service consumer. It is not a "Web-API-to-Web-API" proxy pattern, such as those used in DeltaCloud[7] or libcloud[8]. Although a proxy pattern offers flexibility, it also affects latency and manageability: The proxy pattern is another level of indirection, so requests to or from a client incur an additional delay. Moreover, the software implementing the proxy pattern is another entity to manage and maintain. Finally, proxy pattern systems implement support for various providers through drivers: If one provider changes its interface, the driver within the implementing proxy software must also be updated, and with multiple providers being present, the maintenance requirement again increases.

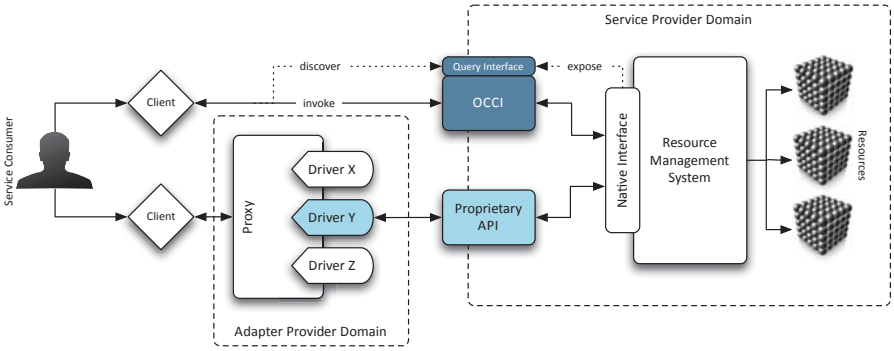


Fig. 10.4 *The Open Cloud Computing Interface as a replacement for proxy-based API approaches. Here, we see the overhead of an additional software (or even middleware) layer in the process, adding overall latency and, more importantly, maintenance costs per additional driver.*

[7] See `http://deltacloud.apache.org/`.
[8] See `http://libcloud.apache.org/`.

Using OCCI as the interface to the resource management framework removes the need for proxies and drivers, see Figure 10.4. We should thus view proxies as a temporary solution to support cloud operators wishing to expose proprietary, legacy interfaces. With OCCI, we enable resource management through a standardised API that directly targets a resource management framework-specific API; this is quite different in intent from proxy-style frameworks. Also, we reduce the amount of indirection and abstraction required to get to the final target resource management framework.

Overall, this enables system architecture optimisation by bringing the API closer to the managed resources: We avoid additional dependencies and inefficiencies[9], and reduce the overall management and maintenance necessities for system components.

10.3 Usage of Web Technologies

With the OCCI HTTP specification (Edmonds et al, 2011b) we define how the core model and its extensions can be transported over the wire. When implemented and deployed, OCCI uses many of today's available HTTP features. It builds on the ROA paradigm and uses REST to handle the client and service interactions. Additionally, it defines ways to filter and query the service provider. Each entity (that is, resources and links) is exposed through URIs. To manage these resources, service consumers can use the normal set of HTTP verbs (`POST`, `GET`, `PUT`, and `DELETE`), as they directly map to the meta model's CRUD paradigm. This way, they can alter resource instances by updating their representation.

For other than primitive operations in HTTP-based architectures, Bray (2009) notes the necessity of controller functions for REST-based protocols: Like all requests that reflect an update of a resource instance using HTTP `PUT`, updating a resource should be idempotent. This means that repeated requests against a resource will always have the identical output result and effect on the system. Triggering operations such as shutdown, however, might lead to halting, killing, or suspending. Naturally, the result of the operation cannot be identical to the request in such a case, due to the transition in state. We adopt this viewpoint in OCCI through the notion of actions, triggered by the HTTP `POST` verb, which lead to different state changes in a life cycle.

Actions within the OCCI model can have parameters and, as detailed in the core model, are exposed using a category definition, are therefore discoverable, and can be associated with resource instances. Service consumers can use such a request to retrieve a service provider's single category through the filtering mechanism. This behaviour is exemplified in Listing 10.1: A request against the query interface (ll. 1–4) is enriched with a category definition for

[9] Given that proxies need to hold information about ongoing interactions and therefore must manage state.

compute instance (l. 5). The response of the server (ll. 7ff.), although referring to the query interface location (l. 1) which—by definition—should return *all* known categories, contains the definition of the "compute" category (ll. 10ff.) only, indicating its attributes (ll. 13–18), actions (ll. 19–22), and the canonical location of all instances (l. 12).

Listing 10.1 *A HTTP message exchange for filtering categories through the query interface. Note that, in the reply part, (l.. 10–22), the line breaks are for readability purposes only; normally, the* `Category` *header is sent in a single line in order to comply with the HTTP protocol conventions.*

```
1  > GET /.well-known/org/ogf/occi/ HTTP/1.1
2  > Host: example.com:8888
3  > Accept: */*
4  > Content-type: text/occi
5  > Category: compute; scheme="http://schemas.ogf.org/occi/infrastructure"

7  < HTTP/1.1 200 OK
8  < Server: pyssf OCCI/1.1
9  < Content-Type: text/plain
10 < Category: compute; scheme="http://schemas.ogf.org/occi/infrastructure#";
11     class="kind"; rel="http://schemas.ogf.org/occi/core#resource";
12     location=/compute/; title="A compute instance";
13     attributes="occi.compute.architecture
14                 occi.compute.cores
15                 occi.compute.hostname
16                 occi.compute.speed
17                 occi.compute.memory
18                 occi.compute.state";
19     actions="http://schemas.ogf.org/occi/infrastructure/compute/action#start
20              http://schemas.ogf.org/occi/infrastructure/compute/action#stop
21              http://schemas.ogf.org/occi/infrastructure/compute/action#restart
22              http://schemas.ogf.org/occi/infrastructure/compute/action#suspend"
```

In the current OCCI HTTP specification, we leverage a number of RFCs specifications for the implementation. These include URIs that can identify and handle resources (RFC 3986), well-known locations for the query interface's entry point (RFC 5785), and HTTP Authentication (RFC 2617) to deal with authentication mechanisms. Although we build OCCI on these specifications, service providers might choose to leverage other RFCs[10] to offer clients an even richer API.

In the light of the HTTP protocol and its inherent feature of content types, one might view OCCI's core model as a remake of MIME media types. In fact, they are orthogonal to them, because the main difference lies in their purpose: MIME media types indicate *how* the data delivered is being rendered, whereas categories indicate *what* data is being rendered[11]. In fact, categories

[10] For example, RFC 5246 for transport-layer security.

[11] Remember that in OCCI, the model is decoupled from the rendering, and other renderings might not have MIME media types.

do not attempt to replace MIME media types, but rather complement them and broaden the usage model.

The category type system is more feature-rich than a system using MIME media types alone: Categories are self-descriptive, discoverable through a query interface, and self-sufficient. We allow a resource in the OCCI model to have multiple categories assigned, thereby exposing several facets simultaneously. In combination with MIME media types, categories deliver a powerful system for resource metadata exposure that supports different renderings of the same information for any given resource type. This way, clients can expect a coherent type system that is standardised independently from the data representation while at the same time exploit the benefits of different rendering formats for different purposes, e.g. JSON for rich internet applications, HTML for human-readable representations of OCCI resources, or XML for service-to-service communication.

10.4 Impact of OCCI

The quality of a newly defined standard is measured through its impact to the market. We therefore focus on reviewing the adoption of OCCI in cloud management software products and on discussing application areas beyond infrastructure management.

10.4.1 Application in the Infrastructure

Currently, a number of OCCI implementations—many of them open source—are available, including *OpenNebula* (`http://opennebula.org`) and *libvirt* (`http://libvirt.org`). In production deployments, OCCI is mainly found in academic environments. For example the *SARA HPC Cloud* system, see `http://www.sara.nl/services/cloud-computing`, offers high-performance computing resources to scientists from areas such as geography, ecology bioinformatics, and computer science and comprises 608 cores and 4.75 Tbytes of RAM.

In the commercial domain, the *OpenStack* (`http://openstack.org`) infrastructure management framework shares several of OCCI's ideals and has many early adopters[12]. With the OCCI implementation for the "Nova" release, interoperability is provided not only for the various OpenStack deployments world-wide, but also towards other infrastructure management frameworks as mentioned above. An example of such an environment is the *EGI Federated Cloud* testbed with currently more than 1,400 cores, which uses different infrastructure management frameworks at the participating resource centres, but provides coherent access to them using OCCI.

[12] Including Dell, Rackspace, AT&T, and Hewlett-Packard.

10.4.2 Adoption in other Areas

The SLA@SOI project (Happe et al, 2011) shows the applicability of OCCI in the context of SLA monitoring and enforcement as part of its *Service Manager* component. There, the provisioning of arbitrary services on the basis of a consumer-supplied SLA instance is conducted using the functionality of OCCI's query interface: The enforcing component, the *SLA Manager*, discovers the corresponding Service Manager's capabilities regarding the agreed SLA terms. This way, it can tell whether a client's request can be satisfied or not and ensure that usage is within the limits of the electronic contract. The key benefit of OCCI in this context is its flexibility: New services with new capabilities can be easily exposed and discovered without changing the general management interface.

Another proof-of-concept application of OCCI has been proposed by Edmonds et al (2011b). They present a non-relational database store over HTTP using the OCCI interface. Each entry is represented by a single REST resource, and the resource is described by a simple `keyvalue` category that exposes two OCCI attributes: `key` and `value`. With the fundamental CRUD operations provided by each OCCI resource and an additional `flip` action[13] on the category, it is possible to expose NoSQL databases over a very lightweight interface.

A very interesting use of OCCI is the *CompatibleOne Cloud Service Broker* (`http://compatibleone.org`). There, OCCI is used as the core of its architecture, and extended beyond infrastructure provisioning: Multi-tenant aware federation, integration, and distribution of services is realised through an OCCI-based runtime system. OCCI is used as a message bus for service management and can be considered as the first standards-based Platform as a Service (PaaS) platform.

The most tangible application area in the context of DCIs however is the management of traditional queueing systems, where OCCI has been adopted as a viable remote interface as well: The DRMAA consortium (`http://drmaa.org`) has released a rendering of the second version of their interface specification (Tröger et al, 2012) over OCCI which exposes a batch system's capabilities via a RESTful interface. This work is insofar interesting as it shows a practical example of transforming a generic Interface Description Language (IDL)-based interface description to the OCCI core model.

[13] Which switches key and value content.

11

Conclusion

To see what is in front of one's nose
needs a constant struggle.
—George Orwell

With the commoditisation of compute resources both on the infrastructural and the operational level over the last years, it became easy to build DCI environments and cater modern science and industry's ever-increasing demand for computing power. The management of such environments, however, turned out to be a challenging problem, especially with respect to capacity planning. This is due to the problem of SRM being twofold: Efficiently utilising resources while at the same time satisfying customers is mainly an algorithmic problem, whereas the construction of DCIs that support a federated approach to this problem is a technical challenge. In this work, we approached the problem from both angles.

11.1 Synopsis

On the one hand, we discussed a number of SRM algorithms for different architectural models, namely centralised and distributed DCI federations and evaluated the different options for analysing the performance of SRM algorithms for such environments. We decided for using a simulative approach, as theoretical methods are not well-enough developed and real-world analysis is not feasible due to the necessity of long-term evaluation. Lacking appropriate simulation tools, we designed and build tGSF, a framework for the assessment of SRM algorithms in federated DCIs and created reference results for the non-federated case.

We then reviewed the centralised model regarding passive and active interaction and looked at hierarchical federations as a specialisation of centralised architectures. There, we focused on the relevance of autonomy disclosure for the participating resource centres. For the distributed model, we analysed the benefit of active workload delegation with simple heuristics, introduced mechanisms for robustness against changing workload patterns and malicious behaviour, and reviewed the applicability of resource-centric capacity planning with the "Infrastructure-as-a-Service" approach in mind. Finally, we compared

our results regarding qualitative and quantitative aspects and approached the question of further possible performance gain in DCI environments for the model used in this work.

On the other hand, we discussed technical blueprints for SRM in federated DCIs, starting from a comprehensive federation model for SRM in DCIs. We weighed its assets and drawbacks induced by the simplifications done, and formulated concretisations for interaction regarding centralised and distributed flavours. By reviewing two real-world deployments, we showed the lessons learned from their design, identified two main architectural obstacles for transferring the results from the algorithmic part of our work to production systems, and proposed a general-purpose protocol, fedSRM, for federated capacity planning in DCI environments.

On the basis of fedSRM, we propose solutions that bridge the remaining gaps. First, we reviewed the problem of negotiating and agreeing on SLAs and built AoR, which follows the ROA paradigm and allows handling of electronic contracts between SRM systems in DCI federations. Second, we introduced OCCI, a standardised protocol and API for the provisioning of resources with respect to both the delegation and the leasing use case and highlighted its advantages to proxy-based approaches.

11.2 Results

With two angles approached in this work, we also have two kinds of results: algorithmic ones that allow for more efficient capacity planning, and technical ones that allow for new approaches to DCI building.

Regarding algorithms, we find that, regardless of the architecture, i.e. centralised or distributed, or approach, i.e. workload delegation or resource leasing, the participation of a resource centre in a federation is always beneficial for its performance both from the user and the operator perspective. This is true for both completely deterministic and partly randomised heuristics. Moreover, it turns out that the often-cited issue of secrecy regarding parameters of day to day operation (such as utilisation, queue length, and others) do not have to be considered a decisive factor. On the contrary, we have shown that even though dynamic performance information is not available, it is still possible to make good SRM decisions. Also, we show that new developments[1] in data centre management can already be handled by SRM in an efficient way. Overall, however, it turns out that the room for further improvement is presumably marginal, as current SRM algorithms are close to reaching the approximated Pareto front.

Regarding technology, we show that, although requiring significant implementation effort on the operational side, the transfer of our algorithms

[1] Such as the shift of paradigm from workload centricity to resource centricity, e.g. providing users with virtual machines rather than workload submission services.

to real-world systems can be achieved with a rather simple architecture and only requires minimally invasive changes to the currently used middleware toolkits. This is mainly due to the ROA paradigm that we used in our design: Expressing all relevant entities as independent resources allows us to take full advantage of the HATEOAS principle without abandoning the loosely coupled model. In addition, we stress the importance of standards-compliant solutions for the current resource market and consequently build our protocols and APIs on this requirement. For SLAs, this is achieved by picking up a widely adopted standard, WS-Agreement, and translating it to the REST architectural style. For provisioning of resources, we built a new protocol[2], successfully pursued its standardisation through an Standards Development Organisation (SDO), and promoted its implementation into main infrastructure management middleware.

11.3 Contribution

With these results, we contribute to the state of the art in research regarding federated SRM in DCIs in a number of ways:

11.3.1 Model Definition

The modeling of DCI environments has been subject to research since the early days of Grid computing, when Foster and Kesselman (1998) formulated their vision of a future distributed resource space. Naturally, the complexity of modeling requires to put the focus on a certain level: It is practically impossible to produce a comprehensive model that covers all aspects of DCI federations.

So far, literature has focused on the two extremal points of modeling: Either, DCIs were described on a very high level by building reference models on how interaction generally works; one of the most mature works in this area has been done by Riedel et al (2009). Or the model was defined implicitly by the design of a new middleware, where the implementation imposed the structure of components and their interaction directly; this can be seen for the different middleware toolkits discussed in the previous chapters.

In this work, we addressed the gap between them, focusing on the federation itself and interactions by modeling along a certain functionality, namely SRM. Being (mostly) architecture-agnostic, we contribute participant and interaction models for the specific functionality of federated capacity planning in DCIs throughout the different layers of the system and reach our first research goal, answering the model-related part of Question 1.

[2] Due to the current lack of anything comparable on the market.

11.3.2 Algorithm Development

Regarding algorithm development, the situation is much more diversified; this could be already seen from the plethora of related work regarding each SRM strategy discussed in the second part. Other than for highly specialised use cases, it therefore can be well assumed that breakthrough improvements in capacity planning for DCIs are unlikely to emerge.

That said, most of the research results contributed are either theoretical[3] or lacking long-term evaluation with realistic workload input data. For making statements on the performance of scheduling algorithms that empirically hold, it is however necessary to analyse their behaviour for periods that approach the lifetime of the participating systems.

In this work, we address this aspect by combining the design of new SRM strategies with a thorough, extensive evaluation using real-world input traces from resource centres that were in service for long periods in time. On this basis, we developed, analysed, and compared novel algorithms for federated DCI capacity planning, addressing aspects of autonomy, disclosure, resilience, and changes of paradigm and answering to Question 2, Question 3, and Question 4. In addition, we made an attempt to approximating the potential limits of federation for the very first time, thereby getting closer to answering Question 5. Both aspects together allow us to reach or second research goal.

11.3.3 System Design

Technological gaps are the most common impediments to transferability of scientific advances in the area of SRM strategies. Although other aspects such as reluctance to implement largely untested algorithms in production use, the lack of integration between the different systems poses the main obstacle.

This problem of interoperability naturally leads to the area of standardisation, and especially with respect to DCIs, several SDOs have approached the problem. The most prevalent one in this area, the Open Grid Forum, aimed to address this problem with a comprehensive service model called Open Grid Services Architecture (OGSA) (Foster et al, 2006), which found some adoption during the earlier days of Grid computing, but never had full-fledged implementations (mainly due to the sheer number of services it defines).

In this work, we take a more practical approach to the problem of interoperability by designing a lightweight abstract protocol for handling the specific problem of interaction between participants in a federation for SRM. We further provide implementations using state-of-the-art technology, reusing proven components from OGSA and contributing novel interfaces where necessary, reaching our third and last research goal and answering the techology-related part of Question 1.

[3] Focusing on performance bounds of certain algorithms for certain setups, see Ramírez-Alcaraz et al (2011), or metrics for their analysis, see Schwiegelshohn (2011).

11.4 Outlook

Although we have reached our goal of providing a solid architectural, algorithmic, and technical basis for constructing federations of Distributed Computing Infrastructures on the basis of currently available technology with a potential for real world adoption, a number of issues are still open.

First, the model that was assumed in the first part of this work is, albeit sufficiently sophisticated on the structural level, still very simple with respect to the resources within a data centre. In order to improve this, it is necessary to further investigate the question of workload modelling, regarding two aspects. On the one hand, better workload models are needed in order to create sufficiently long traces for algorithmic analysis, including federation aspects (such as heterogeneity) in the best case. A viable alternative would be the extension of the available archives, but admittedly, this is more a political than a research challenge. On the other hand, the transformation of workload related characteristics between non-uniform systems is unsolved, especially in the light of modern applications more and more leveraging heterogeneous hardware for better performance[4]. But although preliminary work has been done by Gruber et al (2006), this is still a largely open problem.

Second, all algorithms that have been proposed in the second part of this work do not self-adapt to changes in the environment. Even the EFS-based approach will only adapt to changes in the federation structure, but not in the participants' behaviour. While it is arguable whether this is frequently the case, it is necessary to solve this problem on the medium term, as the construction of DCIs will become more and more a commodity, and consequently the number of ill-behaving participants—whether careless or deliberate—will increase. As such, it is necessary to move from the method of offline optimisation to online adaptability. A possible approach to this could lie in the investigation of probabilistic methods such as Reinforcement Learning or variants of swarm intelligence[5] regarding their fitness for the problem.

Third, the protocols and APIs discussed in the third part of this work are bound to the HTTP protocol. While this is not a problem by itself, as HTTP is widely available and currently well-viewed by most vendors, a more general approach would allow for different protocol renderings. This, however, requires a general meta model for the operational primitives and types for both protocols developed in the context of this work. For OCCI, a first step into this direction has been made, but it is yet to be proven whether its core is flexible enough to handle completely different interaction models. To this end, both the meta model and its realisations should be reviewed in matters of

[4] Lately, this can be seen with graphics hardware and specialised programming languages such as OpenGL and CUDA, which nowadays offer primitives not only for rendering tasks, but also for general number-crunching problems.

[5] Such as Particle Swarm Optimisation or Ant Colony Optimisation.

this aspect, and good candidate for doing so is AMQP[6]. For OCCI, this would require a full decoupling of protocol transport and data representation. For AoR, a corresponding extension of the OCCI core model needs to be developed. A rendering of the OCCI meta model over Advanced Message Queuing Protocol (AMQP) would then cater to both methods.

[6] The Advanced Message Queuing Protocol is a standardised and widely adopted protocol for message-oriented systems (see Vinoski, 2006).

A

The teikoku Grid Scheduling Framework

What I cannot create, I do not understand.
—Richard Phillips Feynman

THE PHYSICAL ASSESSMENT of a system requires a set of tools that perform the evaluation itself. This includes the generation (or usage) of the input data sets, logic that runs the evaluation process, and means for extracting probes of the metrics used for analysis. tGSF is designed to address these issues using a layer architecture, see Figure 3.1. In the following, we introduce those layers and detail their functionality.

A.1 Foundation

The foundation layer of tGSF provides core functionality for event-based execution, see Kelton and Law (1992), with different event types, notification models, and time advancements.

A.1.1 Component Structure

The main classes in this context are `Kernel` and `Event`. While the latter describes events in the system, gives them order, and allows for tagging and classification, the former takes care of managing event objects themselves, including handling at a certain moment in time as well as a *Observer* pattern-based notification of potential subscribers, see Gamma et al (1994).

Figure A.1 depicts these relationships in detail. The `Kernel` manages all `Event` objects that need handling, and new ones can be dispatched to it from the outside. On triggering through its `Awakeable` interface, it dequeues all events relevant to the current simulation time and notifies all registered `Listener`s about their before, during, and after happening (depending on their `ModeOfConcern`). The `Clock` provides the current simulation time and manages the pace of the system by controlling the ratio between real and simulation time.

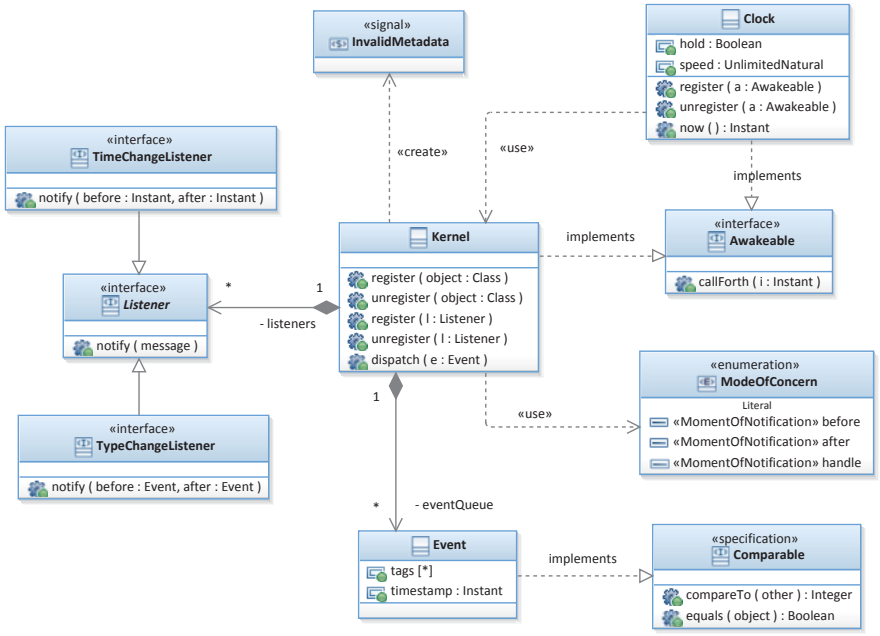


Fig. A.1 *Relationships between the different parts of the simulation engine within tGSF.* `Kernel` *and* `Event` *comprise the main functionality within the component. Registrations of classes with missing or faulty annotations are signalled by the* `InvalidMetadata` *type. The* `Clock` *keeps the pace of the system and provides a general notion of time to the simulation engine.*

A.1.2 Layer Interfacing

The interfacing towards the model of the executed environment evolved over time. Initial designs used strict interfacing to provide events to the consumers, but during implementation of the different scenarios discussed in Part II it turned out that event consumers usually listen to a specific event type or have a single mode of concern, leaving the implementors with cluttered code in the local and federated resource management layer[1].

The second iteration featured an Aspect Oriented Programming (AOP)-based implementation, see Kiczales et al (1997). Here, code cleanness could be achieved easily (due to the fact that consumer code was not touched at all), but user acceptance was low again: The invisible code weaving of *joinpoints* that add functionality to certain parts of the code, and *pointcuts* that provide

[1] More specifically, it showed that there is no good tradeoff between the number and size of interfaces for the given use case: Either users had to implement many small, only slightly different interfaces with a single method only or few, but large interfaces leaving most methods empty.

the metadata for selecting these parts based on certain criteria made it very hard for the users to understand the overall data flow in the framework.

Eventually, it was decided to replace the AOP-based model by an stereotype-based approach. Here, users of the framework annotate the model of their simulated environments according to the need for notification by the simulation engine. To this end, three annotation types were introduced. Classes declared as `@EventSink`s are eligible for registering themselves as objects with the `Kernel`, and are notified on event occurrence. Methods carrying an `@AcceptedEventType` indicate that they wish to be called only for certain types of events, letting the `Kernel` invoke them during every `callForth(Instant)` cycle, but only if a matching event has occurred. Complementing this, additional interfaces for general, non-event specific notifications are available, which derive from `Listener`.

A.1.3 The Notion of Time

An important feature of the execution engine is that the `Clock` does not depend on the system clock of the machine tGSF is running on, but rather provides a configurable pace for the system.

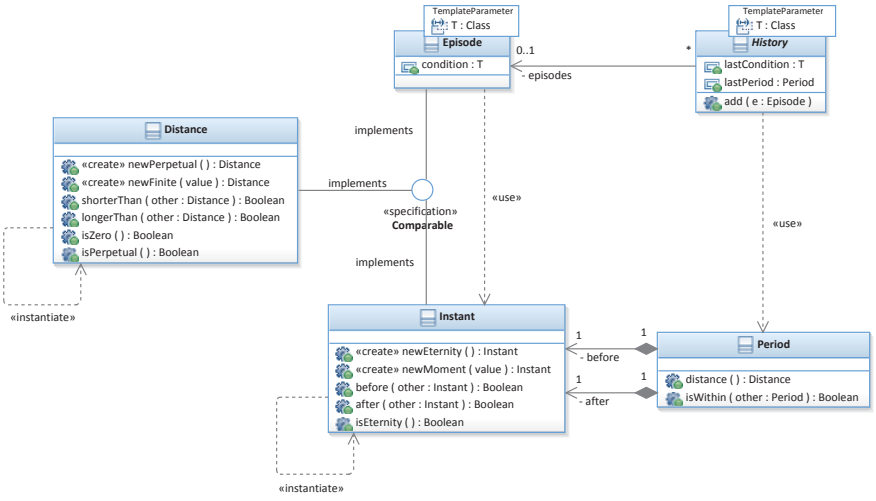


Fig. A.2 *Structure of the time system within tGSF. The main types are* `Clock`*,* `Instant`*, and* `Distance`*, which allow for an independent view on the pace within the system. Additionally, abstract classes for constructing* `History` *records comprising one or more* `Episode`*s are provided.*

To this end, it introduces its own notion of time, see `Instant` and `Distance` in Figure A.2, which can advance at the same as or a different speed than real

time[2], or in discrete mode (i.e. moving forward to the next event instantly, essentially allowing event-discrete simulations). This way, it is possible to evaluate the performance of SRM algorithms also under real-world timing constraints, as shown by Staszkiewicz (2008). In addition, it allows for keeping a `History` of `Episode`s that couple arbitrary objects to a certain `Instant` in time. These relationships are depicted in Figure A.2.

In the `Kernel`, events are queued and dispatched to all registered consumers as soon as they become relevant[3]. At that point, all consumers are triggered depending on their `ModeOfConcern`, which is inspired from *pointcut* definitions in AOP. If an `@MomentOfNotification` annotation is present on a receiving method in an `@EventSink`, the consumer registers for different notification modes, and depending on the selected mode, it will be notified

- before the event is being handled by any consumer,
- for handling of the event by this consumer, and
- after the event has been handled by all consumers.

The rationale behind this model is mainly audit: For evaluation purposes, it is necessary that the situation before and after a certain instant in time has passed can be analysed. Also, during execution it was used to handle persistency management.

A.1.4 Parallelisation

For the simulation case, tGSF supports both sequential and parallel event execution as described by Fujimoto (2000). While the model for sequential execution follows a straightforward approach of tight coupling between event queue and virtual clock, see Figure A.3a, the parallel approach needs more attention.

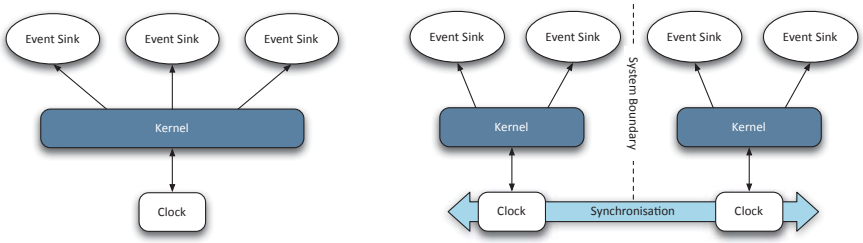
In literature, a number of different approaches to parallel event-discrete simulation methods are discussed, with *Global Time* (Steinman et al, 1995), *Link Time* (Peacock et al, 1979), and *Time Warp* (Jefferson, 1985), being the most popular ones. Marth (2010) has shown for tGSF that *Global Time* is a good compromise between efficiency of parallelisation and implementation effort: while *Link Time* is not applicable to the problem at hand[4], *Time Warp* is very hard to implement for the online scheduling case and cannot leverage its main advantage of concurrent speculative event processing due to the high cost of rollbacks because of the tight coupling of the federation regarding the impact of scheduling decisions.

Technically, the parallelisation requires only three small modifications to the sequential implementation, see Figure A.3b:

[2] I.e., it can be configured to run slower or faster than the system clock.

[3] An event becomes *relevant* when the `Clock` reaches an instant at or after its dispatch time.

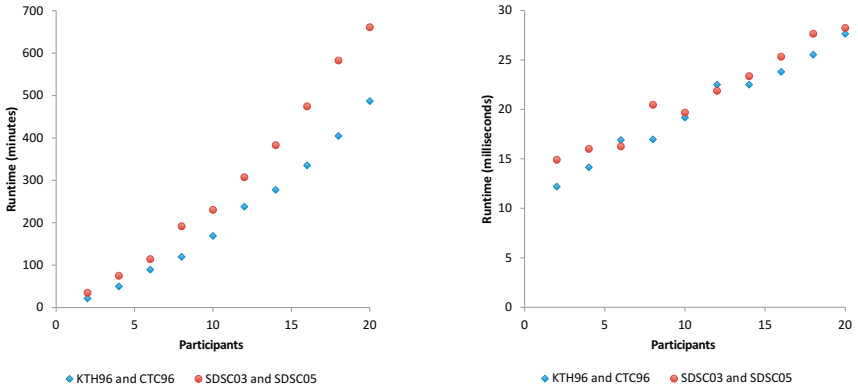
[4] It is specifically designed for offline cases where a-priori knowledge about the simulation is available.



(a) *Sequential Processing.* **(b)** *Parallel Processing.*

Fig. A.3 *Structural overview of the execution engine for different event-discrete approaches within tGSF.*

1. A basic communication module that hides the Java Remote Method Invocation (RMI) technicalities from the simulation kernel and clock,
2. A simple synchronisation mechanism to keep the current simulation time coherent on all machines, and
3. A means for holding the simulation on a machine until the other machines have reached the synchronisation point.



(a) *Runtime per simulation.* **(b)** *Runtime per job.*

Fig. A.4 *Performance of tGSF for distributed parallel simulations using* Global Time.

Figure A.4 shows performance results for the *Global Time* approach with respect to scaling the number of participants of a federation. It shows that the increasing factor for simulation runtime is hidden in the communication cost of the Java RMI-based implementation, see Figure A.4a. Looking at the

normalised simulation time for each job as shown in Figure A.4b, it shows that the parallelisation itself yields almost linear scalability of the simulation.

During non-parallelised simulations, the simulation time drops to 0.7 ms/job, which translates to a factor of approximately 15. The reason behind the large increase with respect to sequential execution lies in the use of Java RMI, which provides a simple, yet costly means of remote object management. This leaves ample room for improvement in future developments, but given the fact that for the work at hand, sequential execution turned out to be still feasible, this was not further pursued.

Overall, the use of parallelised simulations with the current simulation engine can be considered as justified only when evaluating very large setups with many participants.

A.2 Runtime Environment

The Runtime Environment layer in tGSF provides cross-concern functionality for executing an DCI model for assessment. It provides general components for application management, probing, and persistence. Figure A.5 gives an overview of the component's structure.

The entry point for application management is `Bootstrap`, which provides the Command Line Interface (CLI) for tGSF and takes care of importing the configuration. Initially, this was handled through simple key/value property files, but with the growth of the framework, it was necessary to replace this mechanism with an Inversion of Control (IoC)-based approach[5], see Johnson and Foote (1988) or Sweet (1985), to properly handle the increased complexity. Depending on the CLI options selected by the user, tGSF starts as a system service (effectively providing a research LRM running in real-time mode) or as a standalone application. In either case, the environment under assessment needs to be set up, workload sources must be initialised, and the outcome needs to be exported to a storage.

These tasks are orchestrated by the `RuntimeManager` type. It takes care of loading trace input using `EventSource`s (whether from a real data source, a recorded trace, or a generator), creates the simulated environment, and starts the simulation by instantiating `Clock` via a separate thread while attaching itself as an observer. After that, it essentially waits for a termination signal: In service mode, a process shutdown request is expected, and in application mode, all data sources need to indicate that no more workload will be coming. In either case, it terminates the application gracefully.

Every execution of tGSF provides a `RuntimeEnvironment` which takes care of the configuration of utility components. On the one hand, it initialises and tracks all `Probe`s in the runtime and acts as a service for measuring different parts of the system under assessment. Each probe makes use of the `Kernel`'s

[5] Using the Spring Framework (`http://www.springsource.org/`).

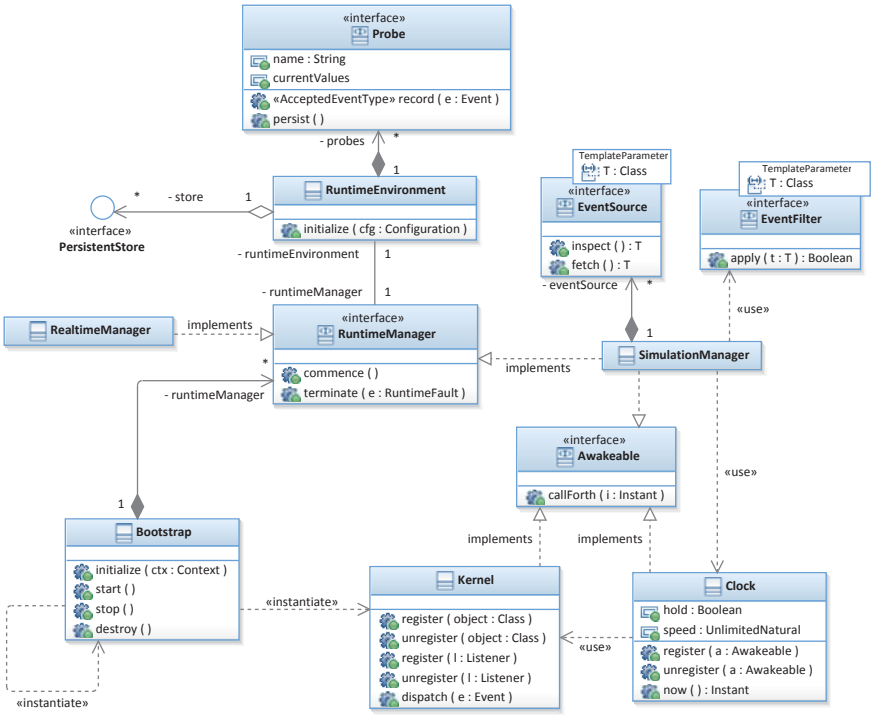


Fig. A.5 *Relationships between the different parts of the runtime environment within tGSF. The main types here are* `Bootstrap`*,* `RuntimeManager`*, and* `ResourceCenter`*, which act as the overall "glue" between all other components of the system when executing an experiment.*

`ModeOfConcern` in Foundation, see Section A.1, to measure the current system state at the correct moment in time. About 30 different probes have been implemented, and among them are all metrics introduced in Section 3.3. On the other hand, it comprises the `PersistentStore` that exposes services for write-out of data to a storage system. Two implementations have been provided here: one exports text-based files, pre-formatted for further analysis in spreadsheet applications (using the CSV format) and MATLAB®; the other, as shown by Wiggerich (2009), provides database-based storage[6] using SQL.

A.3 Local Resource Management

So far, we focused on the infrastructure of the simulation engine itself. Starting from this layer, we introduce business functionality and begin with the

[6] Technically, this has been realised by using the embedded database engine SQLite, see `http://www.sqlite.org`.

components for describing a single participant with respect to its SRM capabilities. The foundation for this is laid by a data model for workload, which provides a holistic view to units of work within a DCI, see Figure A.6.

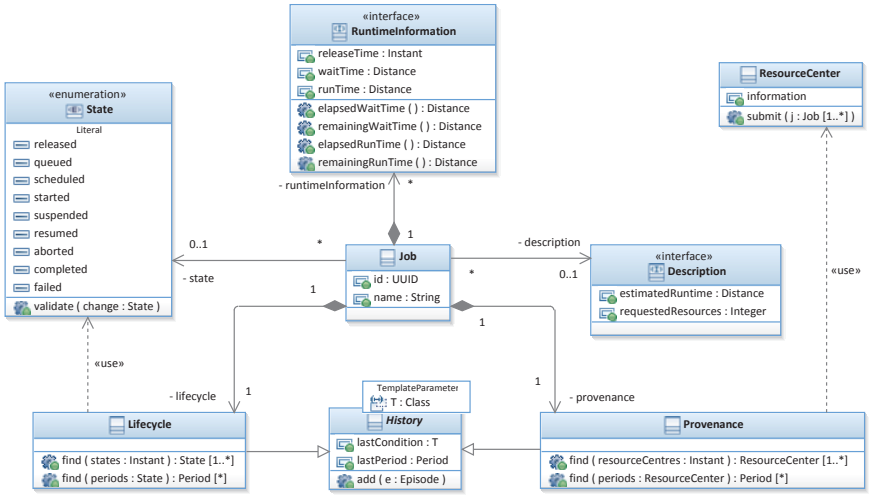


Fig. A.6 *Data model for workload description and tracking within the runtime environment of tGSF.* `Job` *denotes the main type here, with* `Lifecycle` *and* `Provenance` *accompanying it.*

This data model component consists of

- a description of static job characteristics, see Section 3.2.2, and resource requirements. These are modelled by the `Job` and `Description` types;
- the `RuntimeInformation` of an active job, exposing its current wait time and runtime, and its `State`;
- a lifecycle memory representing past and current states of the job, modelled by the `Lifecycle` type; and
- a provenance record which tracks the route a job has taken from its initial submission until final execution in the DCI, modelled by the `Provenance` type.

On the Local Resource Manger layer, each `ResourceCenter` represents a single provider in the federation. On the LRM level, it comprises the underlying `QueueingSystem`. It manages a `Queue` and a `Schedule`[7], and maintains a number of `Strategy` instances responsible for making local scheduling decisions. Besides this, it uses an `ExecutionSPI` towards the `Machine`s' operating

[7] Technically, the framework allows for multiple queues and schedules, but following the model in Section 2.4.1, we only assume one.

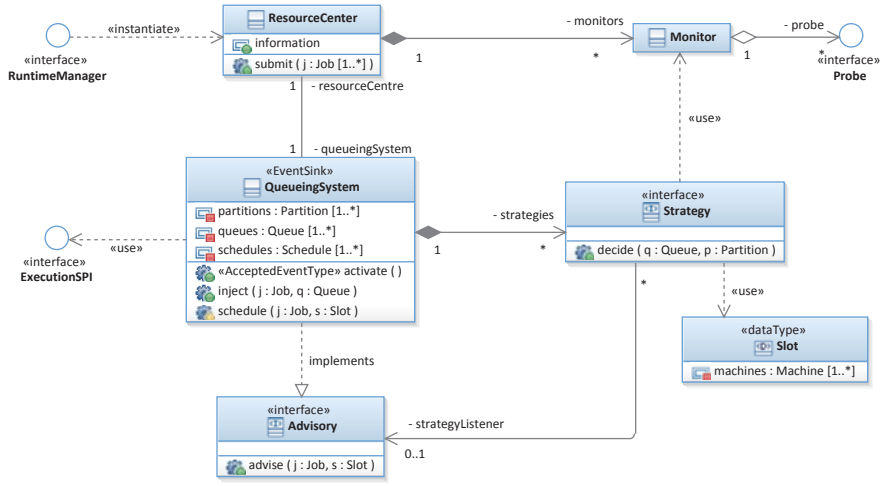


Fig. A.7 *Relationships between the different parts of* Local Resource Management *within* tGSF*. Main types in this context are* `QueueingSystem` *and* `Strategy`*, which in turn operates on the* `Schedule`*, using* `Slots`*.*

systems and an information provider abstraction (the `Monitors`) to query the current state of the resource landscape.

The strategies implemented within this component are of advisory nature: Although they calculate the assignment of workload to resources independently, only using data queried from the information provider, their decision is not binding to the encapsulating `QueueingSystem` implementation. Instead, they only "propose" their solutions, and the scheduler decides on how to act. This offers additional flexibility, since more than one strategy can be consulted at any time, leaving room for additional methods to choose the most appropriate solution. For this work, the FCFS, EASY, and LIST local strategies have been implemented.

With respect to the `ExecutionSPI`, several options are available. The simplest one is an implementation for pure simulation use cases: as soon as the scheduler deems a job ready for execution, it takes its description and, depending on the processing time provided in the recorded trace, injects corresponding completion events into the simulation system. More elaborate implementations allow the execution through operating system processes or via popular LRM interfaces such as Globus GRAM, see Foster (2006).

A.4 Federated Resource Management

The highest layer in tGSF is responsible for *Federated Resource Management*. It takes care of mediating between the different participants (i.e.

ResourceCenters) in the federation. Figure A.8 shows the different components within the layer.

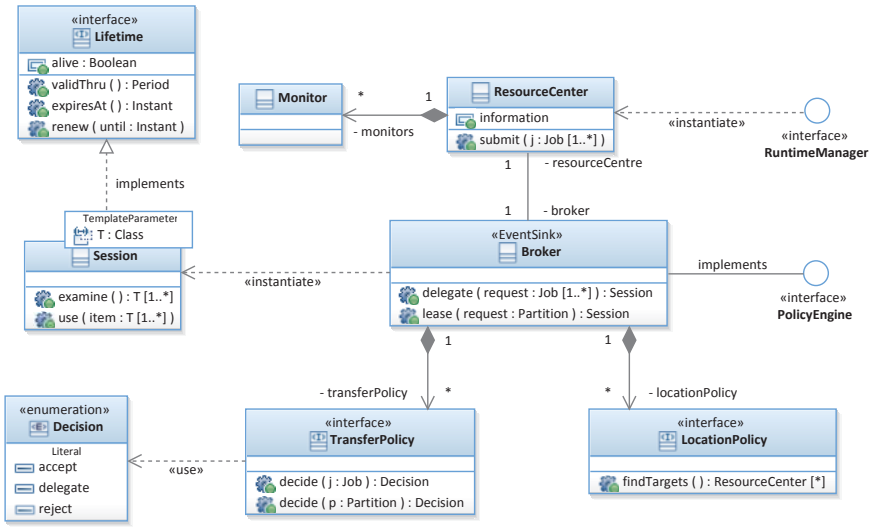


Fig. A.8 *Relationships between the different parts of* Federated Resource Management *within tGSF. Main types in this context are* `Broker`*,* `Session`*, and the policy interfaces.*

The main component is the `Broker`, which exposes interfaces for the management of resources towards participants of the federation and the submission of workload for the local user community, adopting the model discussed in Section 2.4.2. It provides two different mechanisms for workload management, namely delegating workload to other participants or leasing capacity from other participants. This immediately raises three questions each `Broker` must be able to answer:

1. Given an offer from remote, should it (or a subset of it) be accepted?
2. Given the current local situation, should a request be made, and if so, regarding what?
3. Provided that an interaction should be done, which partners in the federation should be contacted?

To make corresponding decisions, each `Broker` acts upon a pluggable policy framework that provides support for codifying rules of procedure regarding workload management with federated partners. Following the questions from above, the policy framework consists of two general rulesets:

1. For an incoming request (whether submitted from local or remote), decide whether to accept or reject it. This is handled by `TransferPolicy` implementations.
2. For an outgoing request, choose appropriate candidates from given set of participants in the federation. This is handled by `LocationPolicy` implementations

Every `Broker` can (and usually will) have multiple implementations of each policy, and—in addition—provide the general behaviour for incoming and outgoing requests. Naturally, the rules are intertwined with each other: a certain offer might be accepted for a certain participant, but not for another. That said, different implementations of each policy can be combined independently to vary the behaviour of the `Broker`, and they can rely on dynamic system information through a number of `Monitor`s.

Since it is technically possible (and sometimes even reasonable, e.g. in negotiation scenarios) to offer different, potentially overlapping sets of jobs to more than one participant at the same time, the concept of `Session`s was introduced into the broker. Here, a two-phase commit protocol, see Raz (1995), is used to ensure that a clean overall system state can be maintained at all times.

In order to keep track of the flow of workload within the system, the *Federated Resource Management* layer makes use of the afore described common components for provenance and lifecycle tracking, and the monitoring component. Depending on the policy implementation and possibly the local strategy, this information is also incorporated into the SRM decisions. With that, tGSF provides the framework for assessing the model for federated DCIs as defined in Section 2.4.

Acronyms

DUNA.
—Internet Jargon

ACL	Access Control List ... 179
AJAX	Asynchronous JavaScript and XML
AMQP	Advanced Message Queuing Protocol ... 204
AOP	Aspect Oriented Programming ... 206
AoR	Agreements over REST ... 167
AWWT	Average Weighted Wait Time ... 79
API	Application Programming Interface ... 25
AWRT	Average Weighted Response Time ... 49
AWS	Amazon Web Services ... 32
BES	Basic Execution Service ... 161
BOINC	Berkeley Open Infrastructure for Network Computing ... 31
C3Grid	Collaborative Climate Community Data & Processing Grid ... 155
CDMI	Cloud Data Management Interface ... 191
CERN	Conseil Européen pour la Recherche Nucléaire
CFD	Computational Fluid Dynamics ... 18
CIMI	Cloud Infrastructure Management Interface ... 187
CLI	Command Line Interface ... 210
CRUD	Create/Retrieve/Update/Delete ... 171
CSV	Comma Separated Values
DCI	Distributed Computing Infrastructure ... 2
DLT	Divisible Load Theory ... 31
DMTF	Distributed Management Task Force
DRMAA	Distributed Resource Management Application API ... 25
DSL	Domain Specific Language ... 169
DUNA	Don't Use No Acronyms
EA	Evolutionary Algorithm ... 142
EASY	Extensible Argonne Scheduling sYstem ... 52
EC2	Elastic Compute Cloud ... 32
ECT	Estimated Completion Time ... 85
EFS	Evolutionary Fuzzy System ... 106
EGI	European Grid Infrastructure

ES	Evolution Strategy ... 110
ESGF	Earth Science Grid Federation ... 157
FCFS	First-Come-First-Serve ... 31
fSRM	Federated SRM ... 133
GMF	Gaussian Membership Function ... 108
HATEOAS	Hypertext As The Engine Of Application State ... 170
HPC	High Performance Computing ... 15
HTC	High Throughput Computing ... 19
HTML	Hypertext Markup Language ... 177
HTTP	Hypertext Transfer Protocol ... 165
IaaS	Infrastructure as a Service ... 123
IDL	Interface Description Language ... 197
IETF	Internet Engineering Task Force ... 161
IoC	Inversion of Control ... 210
IPCC	Intergovernmental Panel of Climate Change ... 22
JSDL	Job Submission Definition Language
JSON	Java Script Object Notation ... 179
LHC	Large Hadron Collider ... 77
LIST	List Scheduling ... 53
LRF	Largest Resource First ... 78
LRM	Local Resource Manager ... 20
MIME	Multipurpose Internet Mail Extensions ... 176
MOP	Multi-Objective Problem ... 142
MPI	Message Passing Interface ... 32
MPP	Massively Parallel Processing ... 17
NIST	National Institute of Standards and Technology
NSGA-II	Nondominated Sorting Genetic Algorithm, Version 2 ... 143
NPX	n-Point Crossover ... 147
OCA	OpenNebula Cloud API
OCCI	Open Cloud Computing Interface ... 187
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture ... 202
OPC	Operation Planning and Control ... 20
OSI	Open Systems Interconnection Reference Model ... 191
P2P	Peer-To-Peer ... 26
PaaS	Platform as a Service ... 197
POSIX	Portable Operating System Interface
PTGrid	Plasma Technology Grid ... 155
QoS	Quality of Service ... 26
RDBMS	Relational Data Base Management System ... 16
REST	Representational State Transfer ... 165
RFC	Request For Comments ... 177
RMI	Remote Method Invocation ... 209
ROA	Resource Oriented Architecture ... 164
S3	Simple Storage Service ... 103

SA	Squashed Area 70
SDO	Standards Development Organisation 201
SERF	Smallest Expected Response First 79
SLA	Service Level Agreement 159
SME	Small and Medium Enterprise 160
SOA	Service Oriented Architecture 164
SOAP	Simple Object Access Protocol 164
SPMD	Single Process, Multiple Data 124
SQL	Structured Query Language
SRM	Scheduling and Resource Management 3
SWF	Standard Workload Format 41
tGSF	teikoku Grid Scheduling Framework 35
ToS	Terms of Service 167
TPX	Two-Point Crossover 147
TSK	Takagi–Sugeno–Kang 106
UCX	Uniform Crossover 147
UML	Unified Modeling Language
UR	Usage Record 41
URI	Uniform Resource Identifier 169
URL	Uniform Resource Locator 170
U	Utilisation 53
VO	Virtual Organisation 21
WLCG	Worldwide LHC Computing Grid 32
WSRF	Web Service Resource Framework 170
XML	Extensible Markup Language 169

References

Thou mayest as well expect to grow stronger
by always eating, as wiser by always reading.
—Thomas Fuller

Abraham A, Liu H, Grosan C, Xhafa F (2008) Metaheuristics for Scheduling in Distributed Computing Environments, Studies in Computational Intelligence, vol 146, Springer, Berlin/Heidelberg, chap 9: Nature Inspired Meta-heuristics for Grid Scheduling – Single and Multi-objective Optimization Approaches, pp 247–272. DOI 10.1007/978-3-540-69277-5_9

Aida K (2000) Effect of Job Size Characteristics on Job Scheduling Performance. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1911, Springer, Cancun, pp 1–17, DOI 10.1007/3-540-39997-6_1

Altunay M, Avery P, Blackburn K, Bockelman B, Ernst M, Fraser D, Quick R, Gardner R, Goasguen S, Levshina T, Livny M, McGee J, Olson D, Pordes R, Potekhin M, Rana A, Roy A, Sehgal C, Sfiligoi I, Wuerthwein Fa (2011) A Science Driven Production Cyberinfrastructure—the Open Science Grid. Journal of Grid Computing 9(2):201–218, DOI 10.1007/s10723-010-9176-6

Anderson DP (2004) BOINC: A System for Public-Resource Computing and Storage. In: Buyya R (ed) Proceedings of the 5th International Workshop on Grid Computing, IEEE/ACM, IEEE Press, Pittsburgh (PA), Grid, pp 4–10, DOI 10.1109/GRID.2004.14

Andreeva J, Campana S, Fanzago F, Herrala J (2008) High-Energy Physics on the Grid: the ATLAS and CMS Experience. Journal of Grid Computing 6(1):3–13, DOI 10.1007/s10723-007-9087-3

Andrieux A, Czajkowski K, Dan A, Keahey K, Ludwig H, Nakata T, Pruyne J, Rofrano J, Tuecke S, Xu M (2007) Web Services Agreement Specification (WS-Agreement). In: OGF Document Series, no. 107 in Recommendation Track, Open Grid Forum, Muncie (IN), United States of America, URL `http://www.ogf.org/documents/GFD.107.pdf`

Andronico G, Ardizzone V, Barbera R, Becker B, Bruno R, Calanducci A, Carvalho D, Ciuffo L, Fargetta M, Giorgio E, La Rocca G, Masoni A, Paganoni M, Ruggieri F, Scardaci D (2011) e-Infrastructures for e-Science: A Global View. Journal of Grid Computing 9(2):155–184, DOI 10.1007/s10723-011-9187-y

Anedda P, Leo S, Manca S, Gaggero M, Zanetti G (2010) Suspending, migrating and resuming HPC virtual clusters. Future Generation Computer Systems 26(8):1063–1072, DOI 10.1016/j.future.2010.05.007

Bäck T (1996) Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford

Bailey Lee C, Schwartzman Y, Hardy J, Snavely A (2005) Are User Runtime Estimates Inherently Inaccurate? In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3277, Springer, Cambridge (MA), pp 253–263, DOI 10.1007/11407522_14

Balakrishnan H, Kaashoek MF, Karger D, Morris R, Stoica I (2003) Looking up data in p2p systems. Communications of the ACM 46(2):43–48, DOI 10.1145/606272.606299

Barkstrom BR, Hinke TH, Gavali S, Smith W, Seufzer WJ, Hu C, Cordner DE (2003) Distributed Generation of NASA Earth Science Data Products. Journal of Grid Computing 1(2):101–116, DOI 10.1023/B:GRID.0000024069.33399.ee

Behrens M, Carlson M, Edmonds A, Johnston S, Mazzafero G, Metsch T, Nyrén R, Papaspyrou A, Richardson A, Swidler S (2011a) Open Cloud Computing Interface – Core. In: Nyrén R, Edmonds A, Papaspyrou A, Metsch T (eds) OGF Document Series, no. 183 in Recommendation Track, Open Grid Forum, Muncie (IN), United States, URL `http://www.ogf.org/documents/GFD.183.pdf`

Behrens M, Carlson M, Edmonds A, Johnston S, Mazzafero G, Metsch T, Nyrén R, Papaspyrou A, Richardson A, Swidler S (2011b) Open Cloud Computing Interface – Infrastructure. In: Metsch T, Edmonds A (eds) OGF Document Series, no. 184 in Recommendation Track, Open Grid Forum, Muncie (IN), United States, URL `http://www.ogf.org/documents/GFD.184.pdf`

Bell WH, Cameron DG, Millar AP, Capozza L, Stockinger K, Zini F (2003) OptorSim: A Grid Simulator for Studying Dynamic Data Replication Strategies. International Journal of High Performance Computing Applications 17(4):403–416, DOI 10.1177/10943420030174005

Berman F, Wolski R, Casanova H, Cirne W, Dail H, Faerman M, Figueira S, Hayes J, Obertelli G, Schopf J, Shao G, Smallen S, Spring N, Su A, Zagorodnov D (2003) Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Systems 14(4):369–382, DOI 10.1109/TPDS.2003.1195409

Beyer HG, Schwefel HP (2002) Evolution Strategies – A Comprehensive Introduction. Natural Computing 1(1):3–52, DOI 10.1023/A:1015059928466

Bharadwaj V, Ghose D, Robertazzi TG (2003) Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. Cluster Computing 6(1):7–17, DOI 10.1023/A:1020958815308

Birkenheuer G, Carlson A, Fölling A, Högqvist M, Hoheisel A, Papaspyrou A, Rieger K, Schott B, Ziegler W (2009) Connecting Communities on the Meta-Scheduling Level: The DGSI Approach. In: Bubak M (ed) Proceedings of the 9th Cracow Grid Workshop, Cyfronet, Cracow, Poland, CGW, pp 96–103

Birkenheuer G, Brinkmann A, Högqvist M, Papaspyrou A, Schott B, Sommerfeld D, Ziegler W (2011) Infrastructure Federation through Virtualized Delegation of Resources and Services. Journal of Grid Computing 9(3):355–377, DOI 10.1007/s10723-011-9192-1

Blümel F, Metsch T, Papaspyrou A (2011) A RESTful Approach to Service Level Agreements for Cloud Environments. In: Chen J, Dou W, Liu J, Yang LT, Ma J (eds) Proceedings of the 9th International Conference on Dependable, Autonomic, and Computing, IEEE, Sydney, Australia, DASC, pp 650–657, DOI 10.1109/DASC.2011.116

Blumofe RD, Park DS (1994) Scheduling Large-scale Parallel Computations on Networks of Workstations. In: Proceedings of the Third International Symposium on High Performance Distributed Computing, IEEE, IEEE Press, San Francisco (CA), HPDC, pp 96–105, DOI 10.1109/HPDC.1994.340255

Bonacorsi D, Ferrari T (2007) WLCG Service Challenges and Tiered Architecture in the LHC Era. In: Montagna G, Nicrosini O, Vercesi V (eds) Inconti di Fisica delle Alte Energie, IFAE, Springer, Pavia, pp 365–368, DOI 10.1007/978-88-470-0530-3_68

Braun TD, Siegel HJ, Beck NB, Bölöni L, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B (1998) A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems. In: Proceedings of the 17th Symposium on Reliable Distributed Systems, IEEE Computer Society, West Lafayette (IN), SRDS, pp 330–335, DOI 10.1109/RELDIS.1998.740518

Bray T (2009) REST Casuistry. Online (last accessed on May 21, 2012), URL `http://www.tbray.org/ongoing/When/200x/2009/03/20/Rest-Casuistry`

Brucker P (2007) Scheduling Algorithms, 5th edn. Springer, Berlin/Heidelberg

Buyya R, Murshed M (2002) GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. Concurrency and Computation: Practice and Experience 14:1175–1220, DOI 10.1002/cpe.710

Buyya R, Abramson D, Venugopal S (2005) The Grid Economy. Proceedings of the IEEE 93(3):698–714, DOI 10.1109/JPROC.2004.842784

Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems 25(6):599–616, DOI 10.1016/j.future.2008.12.001

Cabinet Office HM (ed) (2011) ITIL Service Design, 3rd edn. IT Infrastructure Library, The Stationery Office, London

Cai Y, Natarajan A, Wong J (2007) On Scheduling of Peer-to-Peer Video Services. IEEE Journal on Selected Areas in Communications 25(1):140–145, DOI 10.1109/JSAC.2007.070114

Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience 41(1):23–50, DOI 10.1002/spe.995

Carretero J, Xhafa F, Abraham A (2007) Genetic Algorithm Based Schedulers for Grid Computing Systems. International Journal of Innovative Computing, Information and Control 3(5):1053–1071

Casanova H, Legrand A, Quinson M (2008) SimGrid: A Generic Framework for Large-Scale Distributed Experiments. In: Proceedings of the 10th International Conference on Computer Modeling and Simulation, Cambridge, UKSIM, pp 126–131, DOI 10.1109/UKSIM.2008.28

Casavant TL, Kuhl JG (1988) A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. IEEE Transactions on Software Engineering 14(2):141–154, DOI 10.1109/32.4634

Chapin S, Cirne W, Feitelson D, Jones J, Leutenegger S, Schwiegelshohn U, Smith W, Talby D (1999) Benchmarks and Standards for the Evaluation of Parallel Job Schedulers. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for

Parallel Processing, Lecture Notes in Computer Science, vol 1659, Springer, San Juan, pp 67–90, DOI 10.1007/3-540-47954-6_4

Chen HY, Hsiung M, Lee HC, Yen E, Lin S, Wu YT (2010) GVSS: A High Throughput Drug Discovery Service of Avian Flu and Dengue Fever for EGEE and EU-AsiaGrid. Journal of Grid Computing 8(4):529–541, DOI 10.1007/s10723-010-9159-7

Chen S, Zhang W, Ma F, Shen J, Li M (2004) The Design of a Grid Computing System for Drug Discovery and Design. In: Jin H, Pan Y, Xiao N, Sun J (eds) Grid and Cooperative Computing, Lecture Notes in Computer Science, vol 3251, Springer, Wuhan, pp 799–802, DOI 10.1007/978-3-540-30208-7_108

Chiba T, den Burger M, Kielmann T, Matsuoka S (2010) Dynamic Load-Balanced Multicast for Data-Intensive Applications on Clouds. In: Proceedings of the 10th International Conference on Cluster, Cloud and Grid Computing, IEEE/ACM, IEEE Press, Melbourne, CCGrid, pp 5–14, DOI 10.1109/CCGRID.2010.63

Chun G, Dail H, Casanova H, Snavely A (2004) Benchmark Probes for Grid Assessment. In: Proceedings of the 18th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, Santa Fe (NM), IPDPS, pp 276–283, DOI 10.1109/IPDPS.2004.1303355

Cirne W, Berman F (2001) A Model for Moldable Supercomputer Jobs. In: Proceedings of the 15th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, San Francisco (CA), IPDPS, pp 8–16, DOI 10.1109/IPDPS.2001.925004

Cirne W, Brasileiro F, Andrade N, Costa L, Andrade A, Novaes R, Mowbray M (2006) Labs of the World, Unite!!! Journal of Grid Computing 4(3):225–246, DOI 10.1007/s10723-006-9040-x

Coello Coello CA, Lamont GB (eds) (2004) Applications of Multi-Objective Evolutionary Algorithms, Advances in Natural Computation, vol 1. World Scientific, Singapore

Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Genetic and Evolutionary Computation Series, Springer, Berlin/Heidelberg

Cook A, Gray AJG, Ma L, Nutt W, Magowan J, Oevers M, Taylor P, Byrom R, Field L, Hicks S, Leake J, Soni M, Wilson A, Cordenonsi R, Cornwall L, Djaoui A, Fisher S, Podhorszki N, Coghlan B, Kenny S, O'Callaghan D (2003) R-GMA: An Information Integration System for Grid Monitoring. In: Meersman R, Tari Z, Schmidt DC (eds) On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science, vol 2888, Springer, Catania, pp 462–481, DOI 10.1007/978-3-540-39964-3_29

Cordon O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic Fuzzy Systems – Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. Advances in Fuzzy Systems – Applications & Theory, World Scientific, Singapore

Czajkowski K, Fitzgerald S, Foster I, Kesselmann C (2001) Grid Information Services for Distributed Resource Sharing. In: Proceedings of the 10th International Symposium on High Performance Distributed Computing, IEEE, San Francisco (CA), HPDC, pp 181–194, DOI 10.1109/HPDC.2001.945188

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2):182–197, DOI 10.1109/4235.996017

Deelman E, Singh G, Livny M, Berriman B, Good J (2008) The Cost of Doing Science on the Cloud: The Montage Example. In: Proceedings of the 21st International Conference for High Performance Computing, Networking, Storage and Analysis, 2008., IEEE/ACM, IEEE Press, SC, pp 1 –12, DOI 10.1109/SC.2008.5217932

Dimitrakos T, Mac Randal D, Yuan F, Gaeta M, Laria G, Ritrovato P, Bassem S, Wesner S, Wulf K (2003) An Emerging Architecture Enabling Grid Based Application Service Provision. In: Proceedings of the 7th International Enterprise Distributed Object Computing Conference, IEEE Computer Society, Brisbane, EDOC, pp 240–251, DOI 10.1109/EDOC.2003.1233853

Dong F, Akl SG (2006) Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report 2006-504, School of Computing, Queens University, Kingston (ON)

Dooley R, Milfeld K, Guiang C, Pamidighantam S, Allen G (2006) From Proposal to Production: Lessons Learned Developing the Computational Chemistry Grid Cyberinfrastructure. Journal of Grid Computing 4(2):195–208, DOI 10.1007/s10723-006-9043-7

Dou W, Jia Y, Wang HM, Song WQ, Zou P (2003) A P2P approach for Global Computing. In: Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, IEEE Press, Nice, France, IPDPS, DOI 10.1109/IPDPS.2003.1213451

Downey AB, Feitelson DG (1999) The Elusive Goal of Workload Characterization. ACM SIGMETRICS Performance Evaluation Review 26(4):14–29, DOI 10.1145/309746.309750

Dumitrescu CL, Foster I (2005) GangSim: A Simulator for Grid Scheduling Studies. In: Proceedings of the 5th International Symposium on Cluster Computing and the Grid, IEEE, Cardiff, CCGrid, vol 2, pp 1151–1158, DOI 10.1109/CCGRID.2005.1558689

Edmonds A, Metsch T, Luster E (2011a) An Open, Interoperable Cloud. Online (last accessed on May 21, 2012), URL `http://www.infoq.com/articles/open-interoperable-cloud`

Edmonds A, Metsch T, Papaspyrou A (2011b) Grid and Cloud Database Management, Springer, Berlin/Heidelberg, Germany, chap 1: Open Cloud Computing Interface in Data Management-related Setups, pp 23–48. DOI 10.1007/978-3-642-20045-8_2

Edmonds A, Metsch T, Papaspyrou A, Richardson A (2012) Toward an Open Cloud Standard. IEEE Internet Computing 16(4):15–25, DOI 10.1109/MIC.2012.65

Eernisse M (2006) Build Your Own Ajax Web Applications. SitePoint, Melbourne

Eickermann T, Westphal L, Wäldrich O, Ziegler W, Barz C, Pilz M (2007) Towards Next Generation Grids, Springer, Heidelberg, chap 18: Co-Allocating Compute and Network Resources, pp 193–202. CoreGRID, DOI 10.1007/978-0-387-72498-0_18

Elmroth E, Tordsson J (2008) Grid Resource Brokering Algorithms Enabling Advance Reservations and Resource Selection based on Performance Predictions. Future Generation Computer Systems 24(6):585–593, DOI 10.1016/j.future.2007.06.001

Engelbrecht AP (2007) Computational Intelligence: An Introduction, 2nd edn. John Wiley & Sons, Inc., Hoboken (NJ), United States of America, DOI 10.1002/9780470512517

England D, Weissman J (2005) Costs and Benefits of Load Sharing in the Computational Grid. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3277, Springer, New York (NY), pp 14–32, DOI 10.1007/11407522_9

Ernemann C, Hamscher V, Schwiegelshohn U, Yahyapour R, Streit A (2002a) On Advantages of Grid Computing for Parallel Job Scheduling. In: Proceedings of the 2nd International Symposium on Cluster Computing and the Grid, IEEE/ACM, IEEE Press, Berlin, CCGrid, pp 39–46, DOI 10.1109/CCGRID.2002.1017110

Ernemann C, Hamscher V, Streit A, Yahyapour R (2002b) Enhanced Algorithms for Multi-site Scheduling. In: Parashar M (ed) Grid Computing, Lecture Notes in Computer Science, vol 2536, Springer, Baltimore (MD), pp 219–231, DOI 10.1007/3-540-36133-2_20

Ernemann C, Song B, Yahyapour R (2003) Scaling of workload traces. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 2862, Springer, Seattle (WA), pp 166–182, DOI 10.1007/10968987_9

Ernemann C, Hamscher V, Yahyapour R (2004) Benefits of Global Grid Computing for Job Scheduling. In: Buyya R, Baker M (eds) Proceedings of the 5th International Workshop on Grid Computing, IEEE/ACM, IEEE Press, Pittsburgh (PA), Grid, pp 374–379, DOI 10.1109/GRID.2004.13

Ernemann C, Krogmann M, Lepping J, Yahyapour R (2005) Scheduling on the Top 50 Machines. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3277, Springer, Cambridge (MA), pp 11–16, DOI 10.1007/11407522_2

Fayad C, Garibaldi JM, Ouelhadj D (2007) Fuzzy Grid Scheduling Using Tabu Search. In: Proceedings of the 16th International Fuzzy Systems Conference, IEEE, London, FuzzIEEE, pp 1–6, DOI 10.1109/FUZZY.2007.4295513

Feitelson D (1996) Packing Schemes for Gang Scheduling. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1162, Springer, Honolulu (HI), pp 89–110, DOI 10.1007/BFb0022289

Feitelson D (2001) Metrics for Parallel Job Scheduling and Their Convergence. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 2221, Springer, Cambridge (MA), pp 188–205, DOI 10.1007/3-540-45540-X_11

Feitelson D (2002a) The Forgotten Factor: Facts on Performance Evaluation and Its Dependence on Workloads. In: Monien B, Feldmann R (eds) Euro-Par 2002 – Parallel Processing, Lecture Notes in Computer Science, vol 2400, Springer, Paderborn, pp 81–116, DOI 10.1007/3-540-45706-2_4

Feitelson D (2002b) Workload Modeling for Performance Evaluation. In: Calzarossa M, Tucci S (eds) Performance Evaluation of Complex Systems: Techniques and Tools, Lecture Notes in Computer Science, vol 2459, Springer, Rome, pp 114–141, DOI 10.1007/3-540-45798-4_6

Feitelson D, Rudolph L, Schwiegelshohn U, Sevcik K, Wong P (1997) Theory and Practice in Parallel Job Scheduling. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1291, Springer, Geneva, pp 1–34, DOI 10.1007/3-540-63574-2_14

Feitelson D, Rudolph L, Schwiegelshohn U (2005) Parallel Job Scheduling — A Status Report. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3277, Springer, Cambridge (MA), pp 1–16, DOI 10.1007/11407522_1

Feitelson DG, Tsafrir D (2006) Workload Sanitation for Performance Evaluation. In: Proceedings of 6th International Symposium on Performance Analysis of Systems and Software, IEEE, Austin (TX), ISPASS, pp 221–230, DOI 10.1109/ISPASS.2006.1620806

Feitelson DG, Weil AM (1998) Utilization and Predictability in Scheduling the IBM SP2 with Backfilling. In: Werner B, Torres A (eds) Proceedings of the International Parallel Processing Symposium, IEEE, Orlando (FL), IPPS, pp 542–546, DOI 10.1109/IPPS.1998.669970

Ferrari T (2011) Annual Report on the EGI Production Infrastructure. Deliverable to the European Commission INFSO-RI-261323/D4.2, European Grid Infrastructure, Amsterdam, URL `http://go.egi.eu/1059`

Fidanova S, Durchova M (2006) Ant Algorithm for Grid Scheduling Problem. In: Lirkov I, Margenov S, Wasniewski J (eds) Large-Scale Scientific Computing, Lecture Notes in Computer Science, vol 3743, Springer, Sozopol, pp 405–412, DOI 10.1007/11666806_46

Fielding RT (2000) Architectural styles and the design of network-based software architectures. Doctoral thesis, University of California, Irvine, Irvine (CA)

Fölling A, Grimme C, Lepping J, Papaspyrou A, Schwiegelshohn U (2009) Competitive Co-evolutionary Learning of Fuzzy Systems for Job Exchange in Computational Grids. Evolutionary Computation 17(4):545–560, DOI 10.1162/evco.2009.17.4.17406

Fölling A, Grimme C, Lepping J, Papaspyrou A (2010a) Connecting Community-Grids by Supporting Job Negotiation with Co-evolutionary Fuzzy-Systems. Soft Computing – A Fusion of Foundations, Methodologies and Applications 15(12):2375–2387, DOI 10.1007/s00500-010-0667-y

Fölling A, Grimme C, Lepping J, Papaspyrou A (2010b) Robust Load Delegation in Service Grid Environments. IEEE Transactions on Parallel and Distributed Systems 21(9):1304–1316, DOI 10.1109/TPDS.2010.16

Fölling A, Grimme C, Lepping J, Papaspyrou A (2010c) The Gain of Resource Delegation in Distributed Computing Environments. In: Schwiegelshohn U, Frachtenberg E (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 6253, Springer, Atlanta (GA), United States, pp 77–92, DOI 10.1007/978-3-642-16505-4_5

Foster I (2006) Globus Toolkit Version 4: Software for Service-Oriented Systems. Journal of Computer Science and Technology 21(4):513–520, DOI 10.1007/s11390-006-0513-y

Foster I, Iamnitchi A (2003) On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In: Kaashoek MF, Stoica I (eds) Peer-to-Peer Systems II, Lecture Notes in Computer Science, vol 2735, Springer, Berkeley (CA), pp 118–128, DOI 10.1007/978-3-540-45172-3_11

Foster I, Kesselman C (eds) (1998) The Grid: Blueprint for a Future Computing Infrastructure. Elsevier Series in Grid Computing, Morgan Kaufman, San Mateo (CA)

Foster I, Czajkowski K, Ferguson DF, Frey J, Graham S, Maguire T, Snelling D, Tuecke S (2005) Modeling and Managing State in Distributed Systems:

The Role of OGSI and WSRF. Proceedings of the IEEE 93(3):604–612, DOI 10.1109/JPROC.2004.842766

Foster I, Kishimoto H, Savva A, Berry D, Djaoui A, Grimshaw AS, Horn B, Maciel F, Siebenlist F, Subramanian R, Treadwell J, Von Reich J (2006) The Open Grid Services Architecture, Version 1.5. In: OGF Document Series, no. 80 in Informational Track, Open Grid Forum, Muncie (IN), URL `http://www.ogf.org/documents/GFD.80.pdf`

Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud Computing and Grid Computing 360-Degree Compared. In: Proceedings of the 4th Grid Computing Environments Workshop, IEEE Computer Society, Austin (TX), GCE, pp 1–10, DOI 10.1109/GCE.2008.4738445

Frachtenberg E, Feitelson D (2005) Pitfalls in Parallel Job Scheduling Evaluation. In: Feitelson D, Frachtenberg E, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3834, Springer, Cambridge (MA), pp 257–282, DOI 10.1007/11605300_13

Franke C, Lepping J, Schwiegelshohn U (2006a) On Advantages of Scheduling using Genetic Fuzzy Systems. In: Feitelson DG, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 4376, Springer, Seattle (WA), pp 68–93, DOI 10.1007/978-3-540-71035-6_4

Franke C, Schwiegelshohn U, Yahyapour R (2006b) Job Scheduling for Computational Grids. Forschungsbericht 0206, Technische Universität Dortmund, Dortmund

Franke C, Lepping J, Schwiegelshohn U (2010) Greedy Scheduling with Custom-made Objectives. Annals of Operations Research 180(1):145–164, DOI 10.1007/s10479-008-0491-2

Freitag S, Wieder P (2011) Guide to e-Science, Springer, Berlin/Heidelberg, chap 2: The German Grid Initiative D-Grid: Current State and Future Perspectives, pp 29–52. Computer Communications and Networks, DOI 10.1007/978-0-85729-439-5_2

Frumkin M, Van der Wijngaart RF (2001) NAS Grid Benchmarks: A Tool for Grid Space Exploration. In: Proceedings of the 10th International Symposium on High Performance Distributed Computing, IEEE Computer Society, San Francisco (CA), HPDC, pp 315–322, DOI 10.1109/HPDC.2001.945199

Fujimoto N, Hagihara K (2003) Near-Optimal Dynamic Task Scheduling of Independent Coarse-Grained Tasks onto a Computational Grid. In: Sadayappan P, Yang CS (eds) Proceedings of the 32nd International Conference on Parallel Processing, IEEE, Kaohsiung, ICPP, pp 391–398, DOI 10.1109/ICPP.2003.1240603

Fujimoto RM (2000) Parallel and Distributed Simulation Systems. Parallel and Distributed Computing, Wiley, New York (NY)

Gagliardi F, Jones R, Grey F, Bégin ME, Heikkurinen M (2005) Building an Infrastructure for Scientific Grid Computing: Status and Goals of the EGEE Project. Philosophical Transactions, Series A: Mathematical, Physical, and Engineering Sciences 363(1833):1729–1742, DOI 10.1098/rsta.2005.1603

Gamma E, Helm R, Johnson RE, Vlissides J (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series, Addison-Wesley, Amsterdam

Garey MR, Graham RL (1975) Bounds for Multiprocessor Scheduling with Resource Constraints. SIAM Journal on Computing 4(2):187–200, DOI 10.1137/0204015

Gehring J, Preiss T (1999) Scheduling a Metacomputer with Uncooperative Subschedulers. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1659, Springer, San Juan, pp 179–201, DOI 10.1007/3-540-47954-6_10

Gentzsch W, Girou D, Kennedy A, Lederer H, Reetz J, Riedel M, Schott A, Vanni A, Vazquez M, Wolfrat J (2011) DEISA—Distributed European Infrastructure for Supercomputing Applications. Journal of Grid Computing 9(2):259–277, DOI 10.1007/s10723-011-9183-2

Gombás G, Balaton Z (2004) A Flexible Multi-level Grid Monitoring Architecture. In: Fernández Rivera F, Bubak M, Gómez Tato A, Doallo R (eds) Grid Computing, Lecture Notes in Computer Science, vol 2970, Springer, Santiago de Compostela, pp 214–221, DOI 10.1007/978-3-540-24689-3_27

Graham RL (1969) Bounds on Multiprocessing Timing Anomalies. SIAM Journal on Applied Mathematics 17(2):416–429, DOI 10.1137/0117039

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AH (1979) Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In: Hammer PL, Johnson EL, Korte BH (eds) Discrete Optimization II, Annals of Discrete Mathematics, vol 5, Elsevier, pp 287–326, DOI 10.1016/S0167-5060(08)70356-X

Grama A, Gupta A, Karypis G, Kumar V (2003) Introduction to Parallel Computing, 2nd edn. Pearson Education, Harlow

Greenberg MS, Byington JC, Harper DG (1998) Mobile Agents and Security. IEEE Communications Magazine 36(7):76–85, DOI 10.1109/35.689634

Grimme C, Papaspyrou A (2009) Cooperative Negotiation and Scheduling of Scientific Workflows in the Collaborative Climate Community Data and Processing Grid. Future Generation Computer Systems 25(3):301–307, DOI 10.1016/j.future.2008.05.002

Grimme C, Lepping J, Papaspyrou A (2007) Identifying Job Migration Characteristics in Decentralized Grid Scheduling Scenarios. In: Zheng SQ (ed) Proceedings of the 19th International Conference on Parallel and Distributed Computing and Systems, IASTED, ACTA Press, Cambridge (MA), United States, PDCS, pp 124–129

Grimme C, Lepping J, Papaspyrou A (2008a) Benefits of Job Exchange Between Autonomous Sites in Decentralized Computational Grids. In: Priol T, Lefevre L, Buyya R (eds) Proccedings of the 8th IEEE International Symposium on Cluster Computing and the Grid, IEEE, Lyon, France, CCGrid, pp 25–32, DOI 10.1109/CCGRID.2008.55

Grimme C, Lepping J, Papaspyrou A (2008b) Discovering Performance Bounds for Grid Scheduling by using Evolutionary Multiobjective Optimization. In: Keijzer M (ed) Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, ACM, Atlanta (GA), United States, GECCO, pp 1491–1498, DOI 10.1145/1389095.1389385

Grimme C, Lepping J, Papaspyrou A (2008c) Prospects of Collaboration between Compute Providers by means of Job Interchange. In: Frachtenberg E, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 4942, Springer, Seattle (WA), United States, pp 132–151, DOI 10.1007/978-3-540-78699-3_8

Grimme C, Lepping J, Papaspyrou A (2008d) The Parallel Predator-Prey Model: A Step Towards Practical Application. In: Rudolph G, Jansen T, Lucas S, Beume N (eds) Parallel Problem Solving From Nature, Lecture Notes in Computer Science,

vol 5199, Springer, Dortmund, Germany, pp 681–690, DOI 10.1007/978-3-540-87700-4_68

Grimme C, Lepping J, Moreno Picon J, Papaspyrou A (2010) Applying P2P Strategies to Scheduling in Decentralized Grid Computing Infrastructures. In: Lee WC, Yuan X (eds) Proceedings of the 39th International Conference on Parallel Processing, IEEE Computer Society, San Diego (CA), United States, ICPP, pp 295–302, DOI 10.1109/ICPPW.2010.47

Gruber R, Keller V, Kuonen P, Sawley MC, Schaeli B, Tolou A, Torruella M, Tran TM (2006) Towards an Intelligent Grid Scheduling System. In: Wyrzykowski R, Dongarra J, Meyer N, Wasniewski J (eds) Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, vol 3911, Springer, Poznan, pp 751–757, DOI 10.1007/11752578_90

Guim F, Corbalan J, Labarta J (2007) Modeling the Impact of Resource Sharing in Backfilling Policies using the Alvio Simulator. In: Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, IEEE Computer Society, Istanbul, MASCOTS, pp 145–150, DOI 10.1109/MASCOTS.2007.40

Hagerup T (1997) Allocating Independent Tasks to Parallel Processors: An Experimental Study. Journal of Parallel and Distributed Computing 47(2):185–197, DOI 10.1006/jpdc.1997.1411

Hamscher V, Schwiegelshohn U, Streit A, Yahyapour R (2000) Evaluation of Job-Scheduling Strategies for Grid Computing. In: Buyya R, Baker M (eds) Grid Computing, Lecture Notes in Computer Science, vol 1971, Springer, Bangalore, pp 191–202, DOI 10.1007/3-540-44444-0_18

Happe J, Theilmann W, Edmonds A, Kearney KT (2011) Service Level Agreements for Cloud Computing, Springer, Berlin/Heidelberg, chap 2: A Reference Architecture for Multi-Level SLA Management, pp 13–26. DOI 10.1007/978-1-4614-1614-2_2

Hey T, Trefethen AE (2005) Cyberinfrastructure for e-Science. Science 308(5723):817–821, DOI 10.1126/science.1110410

Hill Z, Humphrey M (2009) A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster? In: Proceedings of the 10th International Workshop on Grid Computing, IEEE/ACM, IEEE Press, Banff (AB), Grid, pp 26–33, DOI 10.1109/GRID.2009.5353067

Holl S, Riedel M, Demuth B, Romberg M, Streit A, Kasam V (2009) Life Science Application Support in an Interoperable E-Science Environment. In: Proceedings of the 22nd International Symposium on Computer-Based Medical Systems, IEEE, Albuquerque (NM), CBMS, pp 1–8, DOI 10.1109/CBMS.2009.5255322

Hong B, Prasanna VK (2003) Bandwidth-Aware Resource Allocation for Heterogeneous Computing Systems to Maximize Throughput. In: Proceedings of the 32nd International Conference on Parallel Processing, IEEE Computer Society, Kaohsiung, ICPP, pp 539–546, DOI 10.1109/ICPP.2003.1240621

Hotovy S (1996) Workload Evolution on the Cornell Theory Center IBM SP2. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1162, Springer, Honolulu (HI), pp 27–40, DOI 10.1007/BFb0022285

Huang J, Jin H, Xie X, Zhang Q (2005) An Approach to Grid Scheduling Optimization Based on Fuzzy Association Rule Mining. In: Stockinger H, Buyya R, Perrott R (eds) Proceedings of the International Conference on e-Science and

Grid Computing, IEEE Computer Society, Melbourne, e-Science, pp 195–201, DOI 10.1109/E-SCIENCE.2005.16

Humphrey M, Thompson MR (2002) Security Implications of Typical Grid Computing Usage Scenarios. Cluster Computing 5(3):257–264, DOI 10.1023/A:1015621120332

Hüsing T (2010) The Role of e-Infrastructures in the Creation of Global Virtual Research Communities. European commission study, empirica GmbH, Bonn

Iosup A, Epema D (2006) GrenchMark: A Framework for Analyzing, Testing, and Comparing Grids. In: Turner SJ, Lee BS, Cai W (eds) Proceedings of the 6th International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Singapore, CCGrid, pp 313–320, DOI 10.1109/CCGRID.2006.49

Iosup A, Dumitrescu C, Epema D, Epema DHJ, Li H, Wolters L (2006) How are Real Grids Used? The Analysis of Four Grid Traces and Its Implications. In: Gannon D, Badia RM, Buyya R (eds) Proceedings of the 7th International Workshop on Grid Computing, IEEE/ACM, IEEE Press, Grid, pp 262–269, DOI 10.1109/IC-GRID.2006.311024

Iosup A, Epema DHJ, Tannenbaum T, Farrellee M, Livny M (2007) Inter-operating Grids through Delegated Matchmaking. In: Proceedings of the 20th International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE/ACM, IEEE Press, Reno (NV), SC, pp 1–12, DOI 10.1145/1362622.1362640

Iverson MA, Özgüner F, Follen GJ (1996) Run-time Statistical Estimation of Task Execution Times for Heterogeneous Distributed Computing. In: Proceedings of the 5th International Symposium on High Performance Distributed Computing, IEEE, Syracuse (NY), HPDC, pp 263–270, DOI 10.1109/HPDC.1996.546196

Jackson D, Snell Q, Clement M (2001) Core Algorithms of the Maui Scheduler. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 2221, Springer, Cambridge (MA), pp 87–102, DOI 10.1007/3-540-45540-X_6

Jagerman DL, Melamed B, Willinger W (1997) Frontiers in Queueing: Models and Applications in Science and Engineering, CRC Press, Inc., Boca Raton (FL), chap 10: Stochastic Modeling of Traffic Processes, pp 271–320

Jakob W, Quinte A, Stucky KU, Süß W (2005) Optimised Scheduling of Grid Resources Using Hybrid Evolutionary Algorithms. In: Wyrzykowski R, Dongarra J, Meyer N, Wasniewski J (eds) Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, vol 3911, Springer, Poznan, pp 406–413, DOI 10.1007/11752578_49

Jann J, Pattnaik P, Franke H, Wang F, Skovira J, Riordan J (1997) Modeling of Workload in MPPs. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1291, Springer, Geneva, pp 95–116, DOI 10.1007/3-540-63574-2_18

Jefferson DR (1985) Virtual Time. ACM Transactions on Programming Languages and Systems 7(3):404–425, DOI 10.1145/3916.3988

Jennings N, Faratin P, Lomuscio AR, Parsons S, Wooldridge MJ, Sierra C (2001) Automated Negotiation: Prospects, Methods and Challenges. Group Decision and Negotiation 10(2):199–215, DOI 10.1023/A:1008746126376

Jin Y, Von Seelen W, Sendhoff B (1999) On Generating FC^3 Fuzzy Rule Systems from Data using Evolution Strategies. IEEE Transactions on Systems, Man, and Cybernetics 29(6):829–845, DOI 10.1109/3477.809036

Johnson RE, Foote B (1988) Designing Reusable Classes. Journal of Object-Oriented Programming 1(2):22–35

Jones B (2005) An Overview of the EGEE Project. In: Türker C, Agosti M, Schek HJ (eds) Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures, Lecture Notes in Computer Science, vol 3664, Springer, Cagliari, pp 904–904, DOI 10.1007/11549819_1

Juang CF, Lin JY, Lin CT (2000) Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design. IEEE Transactions on Systems, Man, and Cybernetics 30(2):290–302, DOI 10.1109/3477.836377

Kacsuk P, Kovacs J, Farkas Z, Marosi A, Gombas G, Balaton Z (2009) SZTAKI Desktop Grid (SZDG): A Flexible and Scalable Desktop Grid System. Journal of Grid Computing 7(4):439–461, DOI 10.1007/s10723-009-9139-y

Kalé LV, Kumar S, DeSouza J (2002) A Malleable-Job System for Timeshared Parallel Machines. In: Proceedings of the 2nd International Symposium on Cluster Computing and the Grid, IEEE/ACM, IEEE Press, Berlin, CCGrid, pp 230–237, DOI 10.1109/CCGRID.2002.1017131

Kee YS, Casanova H, Chien AA (2004) Realistic Modeling and Synthesis of Resources for Computational Grids. In: Proceedings of the 17th International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE/ACM, ACM Press, Pittsburgh (PA), SC, pp 54–63, DOI 10.1109/SC.2004.48

Kelton WD, Law AM (1992) Simulation Modeling and Analysis, 2nd edn. McGraw-Hill, New York (NY)

Kertész A, Kacsuk P (2007) Distributed and Parallel Systems, Springer, Berlin/Heidelberg, chap 20: A Taxonomy of Grid Resource Brokers, pp 201–210. DOI 10.1007/978-0-387-69858-8_20

Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C, Loingtier JM, Irwin J (1997) Aspect-Oriented Programming. In: Aksit M, Matsuoka S (eds) ECOOP'97 – Object-Oriented Programming, Lecture Notes in Computer Science, vol 1241, Springer, Jyväskylä, pp 220–242, DOI 10.1007/BFb0053381

Kielmann T, Bal HE, Gorlatch S (2000) Bandwidth-efficient Collective Communication for Clustered Wide Area Systems. In: Proceedings of the 14th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, Cancun, IPDPS, pp 492–499, DOI 10.1109/IPDPS.2000.846026

Kishimoto Y, Ichikawa S (2004) An Execution-time Estimation Model for Heterogeneous Clusters. In: Proceedings of the 18th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, Santa Fe (NM), IPDPS, pp 105–115, DOI 10.1109/IPDPS.2004.1303053

Krampe A, Lepping J, Sieben W (2010) A Hybrid Markov Chain Model for Workload on Parallel Computers. In: Proceedings of the 19th International Symposium on High Performance Distributed Computing, ACM, Chicago (IL), HPDC, pp 589–596, DOI 10.1145/1851476.1851563

Krauter K, Buyya R, Maheswaran M (2002) A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. Software: Practice and Experience 32:135–164, DOI 10.1002/spe.432

Kübert R, Katsaros G, Wang T (2011) A RESTful implementation of the WS-Agreement specification. In: Pautasso C, Wilde E (eds) Proceedings of the 2nd International Workshop on RESTful Design, ACM, Hyderabad, India, WS-REST, pp 67–72, DOI 10.1145/1967428.1967444

Kurowski K, Nabrzyski J, Oleksiak A, Weglarz J (2007) Grid Scheduling Simulations with GSSIM. In: Proceedings of the 13th International Conference on Parallel and Distributed Systems, IEEE, Hsinchu, ICPADS, vol 2, pp 1–8, DOI 10.1109/ICPADS.2007.4447835

Lamanna DD, Skene J, Emmerich W (2003) SLAng: A Language for Service Level Agreements. In: Proceedings of the 9th Workshop on Future Trends in Distributed Computing Systems, IEEE, San Juan, FTDCS, pp 100–106, DOI 10.1109/FTDCS.2003.1204317

Lei D (2008) A Pareto Archive Particle Swarm Optimization for Multi-Objective Job Shop Scheduling. Computers & Industrial Engineering 54(4):960–971, DOI 10.1016/j.cie.2007.11.007

Lepping J (2011) Dezentrales Grid-Scheduling mittels Computational Intelligence. PhD thesis, Technische Universität Dortmund, Dortmund

Li H, Muskulus M (2007) Analysis and Modeling of Job Arrivals in a Production Grid. ACM SIGMETRICS Performance Evaluation Review 34(4):59–70, DOI 10.1145/1243401.1243402

Li H, Muskulus M, Wolters L (2007) Modeling Correlated Workloads by Combining Model based Clustering and a Localized Sampling Algorithm. In: Proceedings of the 21st Annual International Conference on Supercomputing, ACM, Seattle (WA), ICS, pp 64–72, DOI 10.1145/1274971.1274983

Lifka D (1995) The ANL/IBM SP scheduling system. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 949, Springer, Santa Barbara (CA), pp 295–303, DOI 10.1007/3-540-60153-8_35

Liu H, Abraham A, Hassanien AE (2010) Scheduling Jobs on Computational Grids using a Fuzzy Particle Swarm Optimization Algorithm. Future Generation Computer Systems 26(8):1336–1343, DOI 10.1016/j.future.2009.05.022

Llorente IM, Montero RS, Huedo E, Leal K (2006) A Grid Infrastructure for Utility Computing. In: Proceedings of the 15th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society, Manchester, WETICE, pp 163–168, DOI 10.1109/WETICE.2006.7

Lo V, Mache J, Windisch K (1998) A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1459, Springer, Orlando (FL), pp 25–46, DOI 10.1007/BFb0053979

Lo V, Zappala D, Zhou D, Liu Y, Zhao S (2005) Cluster computing on the fly: P2p scheduling of idle cycles in the internet. In: Voelker G, Shenker S (eds) Peer-to-Peer Systems III, Lecture Notes in Computer Science, vol 3279, Springer, San Diego (CA), pp 227–236, DOI 10.1007/978-3-540-30183-7_22

Lu K, Subrata R, Zomaya A (2006) Towards Decentralized Load Balancing in a Computational Grid Environment. In: Chung YC, Moreira J (eds) Advances in Grid and Pervasive Computing, Lecture Notes in Computer Science, vol 3947, Springer, Taichung, pp 466–477, DOI 10.1007/11745693_46

Lublin U, Feitelson DG (2003) The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. Journal of Parallel and Distributed Computing 63(11):1105–1122, DOI 10.1016/S0743-7315(03)00108-4

Ludwig H, Nakata T, Wäldrich O, Wieder P, Ziegler W (2006) Reliable Orchestration of Resources Using WS-Agreement. In: Gerndt M, Kranzlmüller D (eds)

High Performance Computing and Communications, Lecture Notes in Computer Science, vol 4208, Springer, Munich, pp 753–762, DOI 10.1007/11847366_78

Mach R, Lepro-Metz R, Jackson S (2007) Usage Record – Format Recommendation. In: McGinnis L (ed) OGF Document Series, no. 98 in Recommendation Track, Open Grid Forum, Muncie (IN), URL `http://www.ogf.org/documents/GFD.98.pdf`

Marco C, Fabio C, Alvise D, Antonia G, Francesco G, Alessandro M, Moreno M, Salvatore M, Fabrizio P, Luca P, Francesco P (2009) The gLite Workload Management System. In: Abdennadher N, Petcu D (eds) Advances in Grid and Pervasive Computing, Lecture Notes in Computer Science, vol 5529, Springer, Geneva, pp 256–268, DOI 10.1007/978-3-642-01671-4_24

Marth F (2010) Parallelisierung der Event-Behandlung im teikoku Grid Scheduling Framework. Diploma thesis, Technische Universität Dortmund, Dortmund

Massie ML, Chun BN, Culler DE (2004) The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. Parallel Computing 30(7):817–840, DOI 10.1016/j.parco.2004.04.001

Medernach E (2005) Workload Analysis of a Cluster in a Grid Environment. In: Feitelson D, Frachtenberg E, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3834, Springer, Cambridge (MA), pp 36–61, DOI 10.1007/11605300_2

Mell P, Grance T (2011) The NIST Definition of Cloud Computing. Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg (MD)

Michalewicz Z (1996) Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Berlin/Heidelberg

Mu'alem AW, Feitelson DG (2001) Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. IEEE Transactions on Parallel and Distributed Systems 12(6):529–543, DOI 10.1109/71.932708

Naedele M (2003) Standards for XML and Web services security. Computer 36(4):96–98, DOI 10.1109/MC.2003.1193234

Naroska E, Schwiegelshohn U (2002) On an Online Scheduling Problem for Parallel Jobs. Information Processing Letters 81(6):297–304, DOI 10.1016/S0020-0190(01)00241-1

Netto MAS, Buyya R (2011) Coordinated Rescheduling of Bag-of-Tasks for Executions on Multiple Resource Providers. Concurrency and Computation: Practice and Experience (to appear), DOI 10.1002/cpe.1841

Oleksiak A, Nabrzyski J (2004) Comparison of Grid Middleware in European Grid Projects. In: Fernández Rivera F, Bubak M, Gómez Tato A, Doallo R (eds) Grid Computing, Lecture Notes in Computer Science, vol 2970, Springer, Santiago de Compostela, pp 317–325, DOI 10.1007/978-3-540-24689-3_39

Ostermann S, Prodan R, Fahringer T (2009) Extending Grids with Cloud Resource Management for Scientific Computing. In: Proceedings of the 10th International Workshop on Grid Computing, IEEE/ACM, IEEE Press, Banff (AB), Grid, pp 42–49, DOI 10.1109/GRID.2009.5353075

Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D (2010) A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing. In: Avresky DR, Diaz M, Bode A, Ciciani B, Dekel E, Akan O, Bellavista P, Cao J, Dressler F, Ferrari D, Gerla M, Kobayashi H, Palazzo S, Sahni S,

Shen XS, Stan M, Xiaohua J, Zomaya A, Coulson G (eds) Cloud Computing, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 34, Springer, Munich, pp 115–131, DOI 10.1007/978-3-642-12636-9_9

Paisios E, Zangrando L, Urbah E, Grimshaw AS, Morgan m, Crouch S, Riedel M, Smirnova O, Degtyarenko I, Saga K, Konstantinov A, Newhouse S (2011) Production Grid Infrastructure Use Case Collection. In: Riedel M, Watzl J (eds) OGF Document Series, no. 180 in Informational Track, Open Grid Forum, Muncie (IN), URL `http://www.ogf.org/documents/GFD.180.pdf`

Pareto V (1964) Cours D'Economie Politique. No. 26 in Travaux de Droit, d'Economie, de Sociologie et de Sciences Politiques, Librairie Droz, Paris

Parkin M, Badia RM, Martrat J (2008) A Comparison of SLA Use in Six of the European Commissions FP6 Projects. Technical Report TR-0129, CoreGRID Network of Excellence, Barcelona

Peacock JK, Wong JW, Manning EG (1979) Distributed Simulation using a Network of Processors. Computer Networks 3(1):44–56, DOI 10.1016/0376-5075(79)90053-9

Petcu D, Macariu G, Panica S, Crăciun C (2012) Portable Cloud Applications—From Theory to Practice. Future Generation Computer Systems (to appear), DOI 10.1016/j.future.2012.01.009

Peters I, Stock WG (2007) Folksonomy and Information Retrieval. Proceedings of the American Society for Information Science and Technology 44(1):1–28, DOI 10.1002/meet.1450440226

Pfister GF (2001) Aspects of the InfiniBand architecture. In: Proceedings of the 3rd International Conference on Cluster Computing, IEEE, Newport Beach (CA), CLUSTER, pp 369–371, DOI 10.1109/CLUSTR.2001.960002

Pinedo ML (2012) Scheduling — Theory, Algorithms, and Systems, 4th edn. Mathematics and Statistics, Springer, Berlin/Heidelberg

Pugliese A, Talia D, Yahyapour R (2008) Modeling and Supporting Grid Scheduling. Journal of Grid Computing 6(2):195–213, DOI 10.1007/s10723-007-9083-7

Ramírez-Alcaraz J, Tchernykh A, Yahyapour R, Schwiegelshohn U, Quezada-Pina A, González-García J, Hirales-Carbajal A (2011) Job allocation strategies with user run time estimates for online scheduling in hierarchical grids. Journal of Grid Computing 9(1):95–116, DOI 10.1007/s10723-011-9179-y

Rana O, Ziegler W (2010) Research Challenges in Managing and Using Service Level Agreements. In: Desprez F, Getov V, Priol T, Yahyapour R (eds) Grids, P2P and Services Computing, CoreGRID, Springer, pp 187–200, DOI 10.1007/978-1-4419-6794-7_15

Ranjan R, Harwood A, Buyya R (2008) A Case for Cooperative and Incentive-based Federation of Distributed Clusters. Future Generation Computer Systems 24(4):280–295, DOI 10.1016/j.future.2007.05.006

Ravichandran K, Lee S, Pande S (2011) Work Stealing for Multi-core HPC Clusters. In: Jeannot E, Namyst R, Roman J (eds) Euro-Par 2011 – Parallel Processing, Lecture Notes in Computer Science, vol 6852, Springer, Bordeaux, pp 205–217, DOI 10.1007/978-3-642-23400-2_20

Raz Y (1995) The Dynamic Two Phase Commitment (D2PC) protocol. In: Gottlob G, Vardi M (eds) Database Theory, Lecture Notes in Computer Science, vol 893, Springer, Prague, pp 162–176, DOI 10.1007/3-540-58907-4_14

Riedel M, Wolf F, Kranzlmüller D, Streit A, Lippert T (2009) Research advances by using interoperable e-science infrastructures. Cluster Computing 12(4):357–372, DOI 10.1007/s10586-009-0102-2

Rimal BP, Choi E, Lumb I (2009) A Taxonomy and Survey of Cloud Computing Systems. In: Kim J, Delen D, Jinsoo P, Ko F, Rui C, Hyung Lee J, Jian W, Kou G (eds) Proceedings of the 5th International Joint Conference on Networked Computing, Advanced Information Management and Service, and Digital Content, Multimedia Technology, and its Applications, IEEE Computer Society, Seoul, NCM, pp 44–51, DOI 10.1109/NCM.2009.218

Robertazzi TG, Yu D (2006) Multi-Source Grid Scheduling for Divisible Loads. In: Calderbank R, Kobayashi H (eds) Proceedings of the 40th Annual Conference on Information Sciences and Systems, IEEE Computer Society, Princeton (NJ), CISS, pp 188–191, DOI 10.1109/CISS.2006.286459

Sabin G, Kettimuthu R, Rajan A, Sadayappan P (2003) Scheduling of Parallel Jobs in a Heterogeneous Multi-site Environment. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 2862, Springer, Seattle (WA), pp 87–104, DOI 10.1007/10968987_5

Schley L (2003) Prospects of Co-Allocation Strategies for a Lightweight Middleware in Grid Computing. In: Proceedings of the 20th International Conference on Parallel and Distributed Systems, IASTED, ACTA Press, Orlando (FL), PDCS, vol I, pp 198–205

Schubert L, Kipp A, Koller B, Wesner S (2009) Service-oriented Operating Systems: Future Workspaces. IEEE Wireless Communications 16(3):42–50, DOI 10.1109/MWC.2009.5109463

Schwefel HP (1995) Evolution and Optimum Seeking: The Sixth Generation. John Wiley & Sons, Inc., New York (NY)

Schwiegelshohn U (2010) D-Grid: A National Grid Infrastructure in Germany. Annals of Telecommunications 65(11):763–769, DOI 10.1007/s12243-010-0201-3

Schwiegelshohn U (2011) A system-centric metric for the evaluation of online job schedules. Journal of Scheduling 14(6):571–581, DOI 10.1007/s10951-010-0206-9

Schwiegelshohn U, Yahyapour R (1998a) Analysis of first-come-first-serve parallel job scheduling. In: Proceedings of the Ninth Annual Symposium on Discrete Algorithms, SIAM, Society for Industrial and Applied Mathematics, Philadelphia (PA), SODA, pp 629–638

Schwiegelshohn U, Yahyapour R (1998b) Improving First-come-first-serve Job Scheduling by Gang Scheduling. In: Feitelson D, Rudolph L (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 1459, Springer Berlin / Heidelberg, Orlando (FL), pp 180–198, DOI 10.1007/BFb0053987

Schwiegelshohn U, Yahyapour R (2000) Fairness in Parallel Job Scheduling. Journal of Scheduling 3(5):297–320, DOI 10.1002/1099-1425(200009/10)3:5<297::AID-JOS50>3.0.CO;2-D

Schwiegelshohn U, Yahyapour R (2004) Grid Resource Management – State of the Art and Future Trends, Kluwer Academic Publishers, Dordrecht, chap 4: Attributes for Communication between Grid Scheduling Instances, pp 41–52. No. 64 in Operations Research & Management Science

Schwiegelshohn U, Tchernykh A, Yahyapour R (2008) Online Scheduling in Grids. In: Proceedings of the 23th International Symposium on Paral-

lel and Distributed Processing, IEEE, Miami (FL), IPDPS, pp 1–10, DOI 10.1109/IPDPS.2008.4536273

Shvachko K, Kuang H, Radia S, Chansler R (2010) The Hadoop Distributed File System. In: Proceedings of the 26th International Symposium on Mass Storage Systems and Technologies, IEEE, Lake Tahoe (NV), MSST, pp 1–10, DOI 10.1109/MSST.2010.5496972

Singh G, Kesselman C, Deelman E (2007) A Provisioning Model and its Comparison with Best-effort for Performance-cost Optimization in Grids. In: Proceedings of the 16th International Symposium on High Performance Distributed Computing, ACM, New York (NY), HPDC, pp 117–126, DOI 10.1145/1272366.1272382

Smarr L, Catlett CE (1992) Metacomputing. Communications of the ACM 35(6):44–52, DOI 10.1145/129888.129890

Song B, Ernemann C, Yahyapour R (2004) Modeling of Parameters in Supercomputer Workloads. In: Brinkschulte U, Becker J, Fey D, Großpietsch KE, Hochberger C, Maehle E, Runkler TA (eds) ARCS 2004 – Organic and Pervasive Computing, Lecture Notes in Informatics, vol P-41, Küllen Druck+Verlag GmbH, Augsburg, pp 400–409

Song B, Ernemann C, Yahyapour R (2005a) Parallel Computer Workload Modeling with Markov Chains. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3277, Springer, Boston (MA), pp 9–13, DOI 10.1007/11407522_3

Song B, Ernemann C, Yahyapour R (2005b) User Group-based Workload Analysis and Modelling. In: Proceedings of the 5th International Symposium on Cluster Computing and the Grid, IEEE/ACM, IEEE Press, Cardiff, CCGrid, vol 2, pp 953–961, DOI 10.1109/CCGRID.2005.1558664

Staszkiewicz P (2008) Entwicklung einer Simulationsumgebung für serviceorientiertes, verteiltes Grid-Scheduling. Diploma thesis, Technische Universität Dortmund, Dortmund

Steinman JS, Lee CA, Wilson LF, Nicol DM (1995) Global Virtual Time and Distributed Synchronization. In: Proceedings of the 9th Workshop on Parallel and Distributed Simulation, IEEE, Lake Placid (NY), PADS, pp 139–148, DOI 10.1109/PADS.1995.404307

Sturm R, Morris W, Jander M (2000) Foundations of Service Level Management. Sams Publishing, Indianapolis (IN)

Subramani V, Kettimuthu R, Srinivasan S, Sadayappan P (2002) Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests. In: Proceedings of the 11th International Symposium on High Performance Distributed Computing, IEEE, Edinburgh, HPDC, pp 359–366, DOI 10.1109/HPDC.2002.1029936

Subramaniyan R, Troxel I, George AD, Smith M (2006) Simulative Analysis of Dynamic Scheduling Heuristics for Reconfigurable Computing of Parallel Applications. In: Proceedings of the 14th International Symposium on Field Programmable Gate Arrays, ACM, Monterey (CA), FPGA, pp 230–230, DOI 10.1145/1117201.1117249

Sullivan H, Bashkow TR (1977) A Large Scale, Homogeneous, Fully Distributed Parallel Machine, I. ACM SIGARCH Computer Architecture News 5(7):105–117, DOI 10.1145/633615.810659

Sweet RE (1985) The Mesa programming environment. ACM SIGPLAN Notices 20(7):216–229, DOI 10.1145/17919.806843

Takagi T, Sugeno M (1985) Fuzzy Identification of Systems and Its Applications to Modeling and Control. IEEE Transactions on Systems, Man, and Cybernetics 15(1):116–132

Talia D, Yahyapour R, Ziegler W, Wieder P, Seidel J, Wäldrich O, Ziegler W, Yahyapour R (2008) Using SLA for Resource Management and Scheduling - A Survey. In: Grid Middleware and Services – Challenges and Solutions, CoreGRID, Springer, New York (NY), pp 335–347, DOI 10.1007/978-0-387-78446-5_22

Talwar V, Basu S, Kumar R (2003) Architecture and Environment for Enabling Interactive Grids. Journal of Grid Computing 1(3):231–250, DOI 10.1023/B:GRID.0000035188.36288.39

Tavares AL, Valente MT (2008) A Gentle Introduction to OSGi. SIGSOFT Software Engineerings Notes 33(5):8:1–8:5, DOI 10.1145/1402521.1402526

Tchernykh A, Ramírez J, Avetisyan A, Kuzjurin N, Grushin D, Zhuk S (2006) Two Level Job-Scheduling Strategies for a Computational Grid. In: Wyrzykowski R, Dongarra J, Meyer N, Wasniewski J (eds) Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, vol 3911, Springer, Poznan, pp 774–781, DOI 10.1007/11752578_93

Tchernykh A, Trystram D, Brizuela C, Scherson I (2009) Idle regulation in non-clairvoyant scheduling of parallel jobs. Discrete Applied Mathematics 157(2):364–376, DOI 10.1016/j.dam.2008.03.005

Thies G, Vossen G (2009) Governance in Web-oriented Architectures. In: Proceedings of the 4th Asia-Pacific Services Computing Conference, IEEE Computer Society, Singapore, APSCC, pp 180–186, DOI 10.1109/APSCC.2009.5394126

Tonellotto N, Yahyapour R, Wieder P (2007) A Proposal for a Generic Grid Scheduling Architecture. In: Gorlatch S, Danelutto M (eds) Integrated Research in Grid Computing, CoreGRID, Springer, Pisa, pp 227–239, DOI 10.1007/978-0-387-47658-2_17

Tröger P, Rajic H, Haas A, Domagalski P (2007) Standardization of an API for Distributed Resource Management Systems. In: Proceedings of the 7th International Symposium on Cluster Computing and the Grid, IEEE, Rio de Janeiro, CCGrid, pp 619–626, DOI 10.1109/CCGRID.2007.109

Tröger P, Brobst R, Gruber D, Mamonsi M, Templeton D (2012) Distributed Resource Management Application API, Version 2.0. In: OGF Document Series, no. 194 in Recommendation Track, Open Grid Forum, Muncie (IN), URL `http://www.ogf.org/documents/GFD.194.pdf`

Tsafrir D, Etsion Y, Feitelson D (2005) Modeling User Runtime Estimates. In: Feitelson D, Frachtenberg E, Rudolph L, Schwiegelshohn U (eds) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 3834, Springer, Cambridge (MA), pp 1–35, DOI 10.1007/11605300_1

Tsouloupas G, Dikaiakos M (2005) GridBench: A Workbench for Grid Benchmarking. In: Sloot P, Hoekstra A, Priol T, Reinefeld A, Bubak M (eds) Advances in Grid Computing, Lecture Notes in Computer Science, vol 3470, Springer, Amsterdam, pp 454–456, DOI 10.1007/11508380_23

Uhlig S, Quoitin B, Lepropre J, Balon S (2006) Providing Public Intradomain Traffic Matrices to the Research Community. ACM SIGCOMM Computer Communication Review 36(1):83–86, DOI 10.1145/1111322.1111341

Ullman JD (1975) NP-complete Scheduling Problems. Journal of Computer and System Sciences 10(3):384–393, DOI 10.1016/S0022-0000(75)80008-0

Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2008) A Break in the Clouds: Towards a Cloud Definition. ACM SIGCOMM Computer Communication Review 39(1):50–55, DOI 10.1145/1496091.1496100

Venugopal S, Buyya R, Ramamohanarao K (2006) A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. ACM Computing Survey 38(1):3, DOI 10.1145/1132952.1132955

Vinoski S (2006) Advanced Message Queuing Protocol. IEEE Internet Computing 10(6):87–89, DOI 10.1109/MIC.2006.116

Wäldrich O, Battré D, Brazier FMT, Clark KP, Oey MA, Papaspyrou A, Wieder P, Ziegler W (2011) WS-Agreement Negotiation Version 1.0. In: OGF Document Series, no. 193 in Recommendation Track, Open Grid Forum, Muncie (IN), United States, URL `http://www.ogf.org/documents/GFD.193.pdf`

Wiggerich B (2009) Analyse dynamischer Gridumgebungen mittels graphischer Darstellung. Diploma thesis, Technische Universität Dortmund, Dortmund

Xhafa F, Abraham A (2008) Metaheuristics for Scheduling in Distributed Computing Environments, Studies in Computational Intelligence, vol 146. Springer, Berlin/Heidelberg

Xhafa F, Alba E, Dorronsoro B (2007) Efficient Batch Job Scheduling in Grids using Cellular Memetic Algorithms. In: Proceedings of the 21st International Symposium on Parallel and Distributed Processing, IEEE, Long Beach (CA), IPDPS, pp 1–8, DOI 10.1109/IPDPS.2007.370437

Xu D, Kulkarni S, Rosenberg C, Chai HK (2006) Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution. Multimedia Systems 11(4):383–399, DOI 10.1007/s00530-006-0015-3

Zhang Q, Cheng L, Boutaba R (2010) Cloud Computing: State-of-the-art and Research Challenges. Journal of Internet Services and Applications 1(1):7–18, DOI 10.1007/s13174-010-0007-6

Zhang W, Cheng AMK, Hu M (2006) Multisite Co-allocation Algorithms for Computational Grid. In: Proceedings of the 20th International Symposium on Parallel and Distributed Processing, IEEE, Rhodes Island, IPDPS, DOI 10.1109/IPDPS.2006.1639652

Zimmermann H (1980) OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications 28(4):425–432, DOI 10.1109/TCOM.1980.1094702