UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Institut für Informatik
Paderborn Center for Parallel Computing

Risk Aware Overbooking for SLA Based Scheduling Systems

Dissertation

von

Georg Birkenheuer

Schriftliche Arbeit zur Erlangung des Grades
eines Doktors der Naturwissenschaften

PADERBORN
CENTER FOR
PARALLEL
COMPUTING

Paderborn, Mai 2012

Gutachter
Prof. Dr. André Brinkmann, Johannes Gutenberg-Universität Mainz
Prof. Dr. Friedhelm Meyer auf der Heide, Universität Paderborn

Erklärung zum Antrag auf Eröffnung des Promotionsverfahrens vom 23. Mai 2012

In Ergänzung meines Antrags auf Zulassung zur Promotion in der Fakultät für Elektrotechnik, Informatik und Mathematik der Universität Paderborn erkläre ich gemäß §7 Absatz 4 der Promotionsordnung:

Die gültige Promotionsordnung der Fakultät für Elektrotechnik, Informatik und Mathematik der Universität Paderborn (Fassung vom 26.10.2010) ist mir bekannt.

Die Dissertation wurde von mir unter der Betreuung von Prof. Dr. André Brinkmann an der Universität Paderborn erarbeitet.

Die von mir vorgelegte Dissertation habe ich selbständig verfasst. Die benutzten Quellen und Hilfsmittel sind vollständig angegeben.

Zum Aufzeigen von Promotionsmöglichkeiten ist kein Vermittler gegen Entgelt in Anspruch genommen worden.

Die Dissertation wurde von mir in dieser oder ähnlicher Form an keiner anderen Stelle im Rahmen eines Promotions- oder anderen Püfungsverfahrens vorgelegt.

Ein Promotionsverfahren an einer anderen Hochschule oder an einer anderen Fakultät habe ich weder früher noch gleichzeitig beantragt.

Paderborn, den 31. Mai 2012

Georg Birkenheuer

Vorwort und Danksagung

Augustinus hat einmal geschrieben, *Die Welt ist wie ein Buch. Wer nicht reist, liest nur die erste Seite*. Umgekehrt denke ich, wer ein Buch schreiben will, muss sich auf eine Reise begeben, sonst kommt er nicht über Seite eins hinaus. Auch wenn ich im Rahmen meiner Projekte und den Konferenzen, die ich besuchen durfte, viele schöne Orte sah, ist die Reise, die ich meine, geistiger Natur.

Rückblickend betrachtet, gleicht meine Reise zur Fertigstellung dieser Dissertation einer sehr langen Wanderung. Am Anfang konnte ich den Weg nicht überblicken, ich hatte eine Idee aber keinen Kompass der die Richtung wies.

Auf meinem Weg gab es gute Tage, Wochen und Monate, wenn ich etwas geschafft hatte, eine neue Lösung gefunden oder ein Papier akzeptiert wurde.

Auf dem Weg gab es aber auch schwierige Passagen. Wenn die Projekte alle Zeit brauchten und meine eigene Forschung ruhte. Wenn die Suche nach Fehlern in der Simulation der Suche nach Nadeln im Heuhaufen glich.

Auf dem Weg gab es auch Tage, Wochen und Monate, von denen ich dachte, dass sie gut wären und ich weite Strecken gelaufen sei. Am Ende aber war ich falsch abgebogen, die erdachte Lösung nicht publizierbar und ich fand mich am Ende einer langen Schlucht wieder.

Belohnt wird, wer alle Hürden genommen hat und am Ziel angekommen ist.

Wie bei jeder guten Reise hatte auch ich Gefährten, die mich ein Stück des Weges begleiteten und mir an schwieriger Stelle auch das eine oder andere Stück Ballast abnahmen. Deshalb habe auch ich vielen Menschen zu danken.

Dazu zählen: Matthias Hovestadt, weil er mir die Idee für diese Arbeit auf einer langen Fahrt nach Brüssel in den Kopf gesetzt hat. André Brinkmann, weil er mit seinem Feedback und seinen Ideen viel zur gelungenen Arbeit beitrug. Odej Kao, weil er mich motivieren konnte wie kaum ein Anderer. Holger Karl, für seine Ideen und die Hinweise für eine gute wissenschaftliche Arbeit. Holger Nitsche und Andreas Krawinkel, die technische Probleme immer zu lösen wussten. Inga Poste, die der Aufgabe Herr werden musste, meine englischen Sprachkünste in einen lesbaren Text zu verwandeln. Barbara Kempkes, Axel Keller, Matthias Keller, Markus Autenrieth, Tobias Kenter, Tobias Beisel und Lars Schäfers für Feedback und Korrekturen. Außerdem meine Kollegen Kerstin Voss, Dominik Battré, Matthias Grawinkel, André Höing, Jürgen Kaiser, Dirk Meister, Tobias Schumacher und Jens Simon, die mich immer unterstützt haben. Zu guter Letzt Jens Lischka, der unsere gemeinsame Reise leider viel zu früh verlassen musste.

Abstract

Academia as well as the economy profit from the abilities of today's computer-aided design processes and simulations. When the need for compute resources is not continuous but occurs in peak loads, it is more efficient for a large number of research groups to share an infrastructure or to rent the compute power on demand instead of maintaining an own infrastructure.

Compute resource providers cover the resulting demand. They trade compute power on the basis of automatically negotiable contracts including estimated resources, fees and penalties, runtime, and deadlines. Statistics show that users lack the ability to accurately estimate a job's runtime and tend to overestimate. This leads to low utilization of the provider's infrastructure. However, a high utilization is important to be competitive and profitable.

This thesis introduces two overbooking approaches for the scheduling and negotiation mechanisms of a compute provider. Overbooking exploits the runtime overestimations by using statistics on the user estimation quality to calculate the probability of failure and success when a job is planned with less runtime. Accepting additional, promising jobs increases the utilization of the underlying compute infrastructure, increases the provider's profit, and minimizes the risk of job failures due to overload.

The potential of the presented overbooking approaches is shown based on simulations with real-world job traces. The evaluation demonstrates that academic and commercial resource providers can benefit from overbooking. Careful overbooking allows to successfully execute more jobs even if a few of the additional accepted jobs fail. It is possible to double a provider's utilization and profit by overbooking.

Zusammenfassung

Wissenschaft und Wirtschaft profitieren von modernen computergestützten Designprozessen und Simulationen. Wenn dabei die maximale Rechenlast nicht kontinuierlich sondern nur in seltenen Fällen benötigt wird, ist es effizienter Rechenleistung zu teilen oder zu mieten, als eine eigene Rechnerinfrastruktur zu unterhalten.

Die dadurch entstehende Nachfrage wird von Ressourcenanbietern gedeckt. Sie verkaufen Rechenleistung auf Basis von automatisch verhandelbaren Dienstgütevereinbarungen. Diese enthalten die benötigten Ressourcen, Entgelte und Strafzahlungen, die Laufzeit und den Endtermin. Statistiken zeigen, dass die Kunden die Laufzeit nicht präzise schätzen können und sie deshalb häufig überschätzen. Die Folge sind brachliegende Ressourcen. Eine hohe Auslastung ist aber wichtig, um konkurrenzfähig und profitabel zu arbeiten.

Diese Dissertation stellt zwei Überbuchungsmechanismen vor, die es erlauben Laufzeitüberschätzungen auszunutzen. Die beiden Ansätze verwenden Statistiken über die Nutzerlaufzeitschätzungsgenauigkeit, um die Fehlschlagwahrscheinlichkeit bei einer Überbuchung zu ermitteln. Die Überbuchung erfolgsversprechender Aufträge steigert die Auslastung der Ressourcen, den Gewinn des Anbieters und vermeidet Strafzahlungen wegen Jobausfällen.

Das Leistungsvermögen der Überbuchungsansätze wird auf Basis von realen Jobabläufen evaluiert. Die Simulationen zeigten, dass Ressourcenanbieter von Überbuchungsmechanismen profitierten. Auch wenn einige der zusätzlichen Jobs fehlschlagen, erlaubt ein vorsichtiges Überbuchen mehr Jobs erfolgreich zu Ende zu führen. Es ist möglich die Auslastung und den Gewinn eines Anbieters durch Überbuchen zu verdoppeln.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Scenarios . . . 6
  - 1.2.1 Risk Assessment in Distributed Compute Infrastructures 6
  - 1.2.2 Inter Scheduler Negotiation . . . 8
  - 1.2.3 Cloud Computing . . . 10
- 1.3 Summary . . . 11

2 Related Work **15**
- 2.1 Negotiation and Market Mechanisms . . . 15
  - 2.1.1 Service Level Agreements . . . 16
  - 2.1.2 Market Mechanisms . . . 18
- 2.2 Risk Assessment and Management . . . 20
  - 2.2.1 Failure Data Analysis and Modeling . . . 20
  - 2.2.2 Risk Calculation . . . 20
  - 2.2.3 Risk Assessment . . . 21
  - 2.2.4 Risk Management . . . 22
- 2.3 Scheduling and Planning . . . 22
  - 2.3.1 Packing Theories . . . 23
  - 2.3.2 Cluster Scheduling Strategies . . . 26
  - 2.3.3 Scheduling and Planning Systems . . . 27
- 2.4 Overbooking . . . 30
  - 2.4.1 Overbooking in General . . . 30
  - 2.4.2 Overbooking of IT Systems . . . 31

3 Overbooking **33**
- 3.1 Accepting and Placing Jobs . . . 34
  - 3.1.1 Placing Algorithm for Overbooking . . . 34
  - 3.1.2 Placing Strategy . . . 35
  - 3.1.3 Acceptance Test . . . 35
- 3.2 Introduction of Statistics . . . 38
  - 3.2.1 Selecting Attributes for PDF Creation . . . 39
  - 3.2.2 Building a Joint PDF . . . 40
  - 3.2.3 Resulting Combined PDF for PoF Calculation . . . 41
- 3.3 Overbooking Approaches . . . 42
  - 3.3.1 Comprehensive Approach . . . 42
  - 3.3.2 Heuristic Overbooking . . . 48
- 3.4 Runtime Analysis . . . 51
  - 3.4.1 Comprehensive Approach . . . 51

3.4.2 Heuristic Approach . . . 52
3.4.3 Practical Issues . . . 52
3.5 Market Mechanisms and Overbooking . . . 53
3.6 Evaluated Strategies . . . 55
3.6.1 Conservative Backfilling . . . 55
3.6.2 Comprehensive Overbooking . . . 55
3.6.3 Heuristic Planning . . . 55
3.6.4 Heuristic Overbooking . . . 55

4 Determination of PDF Classes 57
4.1 Correlating Attributes . . . 57
4.2 Analysis . . . 58
4.2.1 Runtime . . . 60
4.2.2 Resources . . . 63
4.2.3 Applications . . . 65
4.2.4 Users . . . 67
4.2.5 Conclusion . . . 67
4.3 User Survey . . . 69

5 Simulation Environment 73
5.1 Simulation Parameters . . . 74
5.1.1 Simulation Input . . . 74
5.1.2 Parallel Workload Archive . . . 76
5.1.3 Arminius . . . 77
5.1.4 Acceptance Tests . . . 77
5.2 Implementation . . . 78
5.2.1 Used Data Structures . . . 78
5.2.2 Events . . . 78
5.2.3 Placing Routines . . . 81

6 Evaluation with PWA Traces 85
6.1 CTC . . . 86
6.1.1 PoF Acceptance Test . . . 88
6.1.2 Risk Acceptance Test . . . 96
6.2 HPC2n . . . 103
6.2.1 PoF Acceptance Test . . . 104
6.2.2 Risk Acceptance Test . . . 106
6.3 LANL . . . 109
6.3.1 PoF Acceptance Test . . . 110
6.3.2 Risk Acceptance Test . . . 112
6.4 SDSC-BLUE . . . 115
6.4.1 PoF Acceptance Test . . . 116
6.4.2 Risk Acceptance Test . . . 118
6.5 SDSC-DataStar . . . 121
6.5.1 PoF Acceptance Test . . . 122
6.5.2 Risk Acceptance Test . . . 124
6.6 SDSC-SP2 . . . 127
6.6.1 PoF Acceptance Test . . . 128

6.6.2 Risk Acceptance Test 130
6.7 Summary 133
6.7.1 PoF Results 133
6.7.2 Risk Results 135
6.7.3 Runtime 138

7 Evaluation of the Arminius Cluster 143
7.1 Runtime-Estimation Analysis 144
7.1.1 PoF Acceptance Test 144
7.1.2 Risk Acceptance Test 146
7.2 Resources Analysis 149
7.2.1 PoF Acceptance Test 149
7.2.2 Risk Acceptance Test 152
7.3 Application Analysis 157
7.3.1 PoF Acceptance Test 157
7.3.2 Risk Acceptance Test 160
7.4 Submission-User Analysis 163
7.4.1 PoF Acceptance Test 163
7.4.2 Risk Acceptance Test 165
7.5 Summary 169
7.5.1 PoF Results 169
7.5.2 Risk Results 170
7.5.3 Runtime 171

8 Conclusion 175
8.1 Discussion 175
8.2 Future Work 178

List of Figures I

List of Tables V

Bibliography VII

Glossary XXVII

1 Introduction

Overbooking is an interesting instrument for fostering the utilization of compute clusters in academia and should also pave the way for a commercial market for selling computing resources. Workload traces show that users cannot accurately estimate the runtime of their compute activities [Feit 10]. The resulting overestimation of runtime can be exploited to overbook compute resources. This work shows that overbooking is a promising method to strengthen a compute provider's ability to fully utilize its own compute resources while fulfilling the guarantees given to the users.

The introduction of this thesis is structured as follows. Beginning with a brief motivation for overbooking, three scenarios are given, which present possible applications of overbooking. The first scenario is based on risk assessment in distributed compute infrastructures, the second one on inter-scheduler negotiation, and the third scenario is an outlook on overbooking in cloud computing. At last, a summary of the contents of this dissertation is given.

Contents

1.1 Motivation 1

1.2 Scenarios 6

1.2.1 Risk Assessment in Distributed Compute Infrastructures 6

1.2.2 Inter Scheduler Negotiation 8

1.2.3 Cloud Computing 10

1.3 Summary 11

1.1 Motivation

The powerful abilities of today's computer driven design processes and simulations are aiding modern research and development tasks. These processes allow to evaluate a broader range of items and mechanisms, which was impossible a few years ago due to the high manual efforts that the underlying experiments would have taken. Exemplary limiting factors are,

- the time expenditures (checking a million chemicals for a certain ability needs time),
- the costs of material (for example, costs of gold or other, very rare, materials),
- the duration of observations (for instance plate tectonics),

- the experiments themselves, which are sometimes very dangerous (due to highly toxic materials, unknown effects within drug design, or radiation in high energy physics),
- hardly reachable locations (long term zero gravity experiments need space laboratories), or
- not reachable locations (e.g. measurements of inner sun processes).

Given sophisticated theorems, all of these experiments can sometimes be replaced but certainly be supported by computer simulations.

However, the powerful abilities and high precision of actual simulations do not come for free but need a huge amount of computing power. While large companies are able to buy and maintain the needed cluster infrastructures, small and medium enterprises (SMEs) or academic working groups lack the abilities to finance and maintain their own clusters.

Academia typically finances and maintains its computing resources by gathering the computing power in university or state wide computing centers. In these centers, all interested researchers can query for resources to serve their research task. While this sharing of resources has the positive effect that it allows access to compute clusters for otherwise underfunded researchers, it has the disadvantage that there is a competition for the given resources.

For scientists it is important to get the needed resources in time for instance, to write a paper. However, the academic computing provider typically offers best effort services. This means, a job is, without guarantees, eventually done.

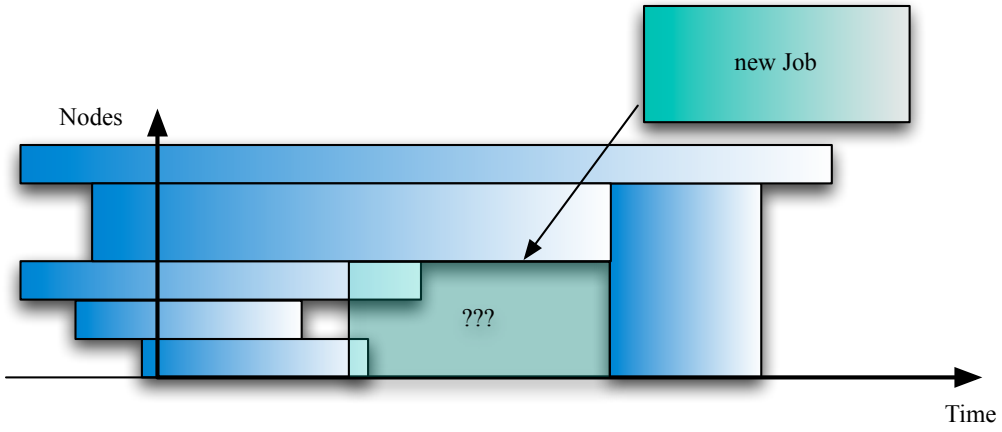
To overcome this drawback, in the last years several ideas came up to establish a certain quality of service (QoS) in academia. An important QoS demand is holding deadlines. A typical deadline is the point in time until when a job definitely has to be finished.

For negotiating and recording such deadlines, service level agreements (SLAs) are an important instrument. SLAs define the quality and quantity of services in a contract. In addition to the deadline, SLAs contain an execution-time window for the negotiated job and a description of the computing resources.

While SLAs in the industry are static, offline negotiated, and paper bound, research in computer science focuses on automated negotiation processes. Automated SLA negotiation is, for example, defined in the WS-Agreement protocol from the Open Grid Forum OGF [Zieg 08] and already used by several projects, like AssessGrid [Batt 07b] or DGSI [Birk 11b].

Within automated negotiations of compute activities, the jobs are defined with a user-estimated execution time, the time from which the job is allowed to start, and the corresponding deadline. The time from which the job is allowed to start is user given and in most of the cases as soon as possible. Normally, the users have no release-time restrictions. However, it is possible that the release (first possible start time) is in the future because the job cannot start until related tasks are fulfilled. A release time restriction can be caused by the stage-in of massive simulation data or because the job has to wait for the finish of related tasks in workflows. In this work, the notion for the job's first allowed start time in the

Figure 1.1 Overbooking example:
How likely is it that the new job fits into the free time-slot?



future is *release time* similar to [Carl 87, Li 97]. In other related work, the term for a job that is released and ready to start often is eligible [Dean 08].

An exemplary SLA might define the execution of a chemical simulation with the application Gromacs [Van 05] on 48 nodes with 12 CPUs each and at least 576 GB RAM. The simulation will run for 26 days and has to be finished within the next month.

To fulfill the SLA, the scheduler creates a plan of all the jobs. The plan defines when and in which order the jobs will be executed. Based on user-estimated execution times, the start-points and deadlines for all jobs can be calculated.

To really keep the deadlines and to provide a service that is better than best effort, jobs have to be killed when their estimated time has elapsed. This allows the following jobs to start and, thereby, to keep their deadlines.

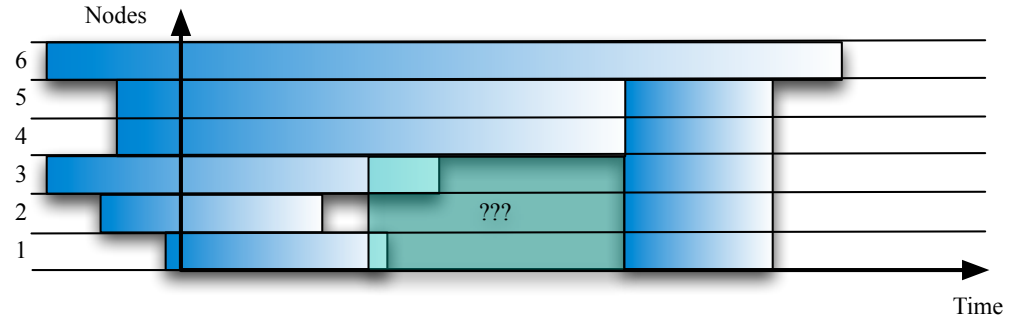
As a result, users are cautious to not lose jobs and tend to overestimate their job's duration and block resources accordingly.

Today's automated SLA negotiation processes do not consider these overestimations. This leads to underutilized resources because, in practice, jobs finish much earlier than planned.

In this work, *overbooking* is proposed to increase the utilization and competitiveness of a resource provider in such a situation. The example in Figure 1.1 shows a new job that does not fit into the schedule without overbooking. Overbooking is widely used in areas like flight ticket sales or hotel room reservations where more people buy tickets or reserve rooms than actually use it. To work as profitable as possible, seats or rooms are assigned more than once. The number of reservations that will lapse is estimated a priori and used in the planning process. However, the estimated number is a value that is only correct with a specific probability.

Consequently, if more hotel passengers or guests appear than estimated, not enough seats in the aircraft or rooms in the hotel are available, and a penalty is imposed.

Figure 1.2 Comprehensive Overbooking:
Calculating the probability by using node and other jobs' information.



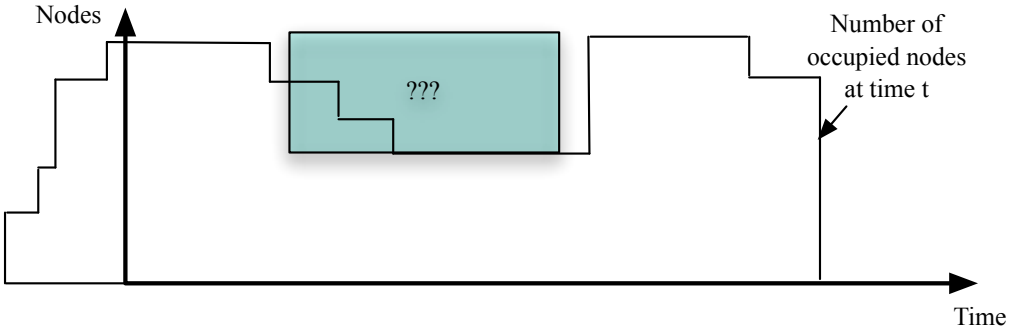
Obviously, the objective of using overbooking is to increase the expected profit. Instead of selling each seat/room once, profit can be increased by selling them several times. This opportunity has to be compared to the risk implied by overbooking, i.e., the compensation/penalty for the buyer if no seat/room is free combined with the probability of that event. The probabilistic best choice of the risk and opportunity will provide the most profit.

In this work, overbooking can be applied when the resource provider would otherwise not be able to accept a job with its required resources and user-estimated duration in the user defined execution window. With conservative scheduling strategies, it is impossible to accept jobs where the maximum estimated job duration is longer than any gap in the schedule. However with overbooking, the scheduler can assess the probability of failure when placing the job in a gap that is smaller than the estimated runtime. For such an *overbooked job*, the probability of failure (PoF) no longer only depends on machine failures like in conservative scheduling [Djem 06]. It also depends on the probability of the job's real runtime being longer than the gap length.

In this dissertation, the SLA negotiation process is extended by the ability to overbook the underlying resources. This instrument increases system utilization and allows to estimate the PoF for a job execution. Developed instruments measure how likely it is that a job is able to finish successfully in a gap that does not provide the full estimated runtime. They then combine the result with an estimation of the probability that the underlying resources do not crash during the execution. Two implemented strategies, a comprehensive approach and a heuristic one, apply overbooking based on the mentioned calculation.

The comprehensive overbooking algorithm and the resulting investigations have, at first, been proposed for overbooking with focus on a single resource [Birk 08a, Birk 09a] and then were extended for parallel resources in [Birk 10]. The comprehensive algorithm was designed to map jobs on resources at SLA negotiation time. The numbered resources in Figure 1.2 show this. It is known a priori on which resource(s) a job will be executed. First, this allows to apply node specific failure assumptions. More importantly, placing jobs on resources at negotiation time allows us to include the probability that a job's direct predecessor(s) finishes early giving the following job more or even its full runtime before the start of

Figure 1.3 Heuristic Overbooking:
Other jobs are not considered, only the free nodes are counted.



the probability of success (PoS) calculation. Consequently, the comprehensive approach is very accurate in its PoS calculation.

Simulations showed that, in practice, this approach is very time consuming due to the many convolutions of the underlying statistics. The combined statistics show how likely it is that a combination of a job and its predecessor(s) ends in a given gap. A further drawback of the comprehensive approach is that it is not flexible. Until the planned job-start time, other jobs end and resources in the system become free that can provide the needed resources much earlier.

Therefore, a heuristic overbooking approach was developed. It no longer decides where the job should run at SLA negotiation time [Birk 11a]. Figure 1.3 shows nearly the same schedule. Only the resource numbers are missing as well as the shape of the single jobs. The heuristic counts the overall resources and calculates how likely the job will be successful. This approach is more flexible because a job might run on any node. The drawback is that it does not know which job(s) will be the predecessor(s). The plan holds all jobs with their estimated runtime. Therefore, in many cases it is possible to guess which jobs end before a job. Nevertheless, if other, longer estimated jobs with later deadlines end earlier it is possible to start a new job on their resources. Therefore, a job does not necessarily start according to the plan's order. Thus, only a general assumption about the resources and their stability can be made. In addition, the application of the heuristic is only possible if all nodes can execute all jobs with the same performance.

For both approaches, an acceptance test decides if the system can accept an additional job and, thus, can help to improve a provider's utilization by overbooking resources.

If in addition to the academic case the commercial resource providers are observed, money comes into play. Actual commercial resource providers like Amazon EC2[1], charge fees for their services. However, they only offer best effort and do not pay penalties to their customers because they do not want to take the risk of violating an SLA.

[1] Amazon Elastic Compute Cloud (Amazon EC2) `http://aws.amazon.com/de/ec2/`

However, with the proposed algorithms in this thesis, it is possible to calculate the risk for an SLA violation based on a possible penalty and the job's PoF. Additionally, a provider can calculate the opportunity of accepting a job, which is the probability of success multiplied by the fee.

Thus, the mechanisms of this thesis give the commercial resource providers the means to calculate whether or not it is profitable to take a job. The customer would profit from the SLA given QoS guarantees and the penalties paid for a lost job.

To investigate overbooking approaches' abilities to support commercial providers, the simulations applied a second risk-based acceptance test. This test showed that, given a commercial environment, a resource provider would also profit from an overbooking process.

1.2 Scenarios

This section presents three scenarios about possible applications of overbooking. The first scenario bases on a research project on machine stability, risk assessment, and risk management. The next scenario is inspired by a project that aims to create interoperability between different distributed compute infrastructures (DCIs). This allows a broader exchange of jobs and, thus, the application of overbooking. The last scenario is about overbooking and cloud computing. This scenario was chosen because state of the art research for compute providers heavily focuses on virtualization.

While overbooking can be applied to the three described topics, it is not limited to them.

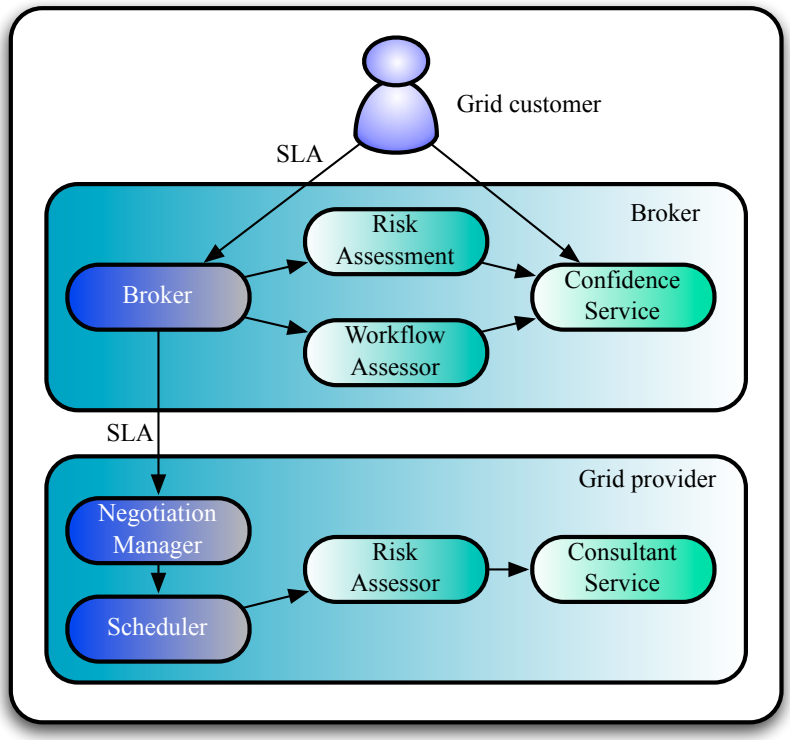
1.2.1 Risk Assessment in Distributed Compute Infrastructures

The first scenario is motivated by the AssessGrid project (*Advanced Risk Assessment and Management for Trustable Grids*). AssessGrid was funded by the EC in the 6th Framework Program [Asse 08].

Even the best maintained cluster could not avoid crashes of the software or failures of the underlying hardware. This means that 100% guarantees for a successful job execution are never possible. SLA bound jobs compensate the violation of a missed deadline with a penalty paid by the resource provider. However in most cases, the penalty will not fully compensate the loss caused by the missed deadline [Mitc 05]. Instead, the risk for a job crash is distributed between customer and provider.

The risk for a job execution is the probability of a node outage multiplied by the impact of the job [Birk 07]. If a job is very important, the risk is high even when the probability of failure of the underlying resources is low [Asse 08].

Figure 1.4 AssessGrid architecture.



Consequently, both the provider and the customer want to know the risk of the job execution in advance [Batt 08c, Birk 06a].

Therefore, AssessGrid developed measures for risk assessment. They allow an estimation of the probability of resource outages. In addition, AssessGrid developed risk management instruments. They cope with resource failures and should prevent SLAs from being violated [Brin 10]. See Figure 1.4 for a visualization of AssessGrid's architecture.

For grid customers, the risk estimations are the basis for a provider reliability measure. This measure is created by a so-called *Confidence Service* and is used by the broker, a trader service that procures SLAs between provider and customers. This allows the customer to choose the most reliable provider when he wants to be sure to get their results in time.

For a grid provider, the estimations of a *Consultant Service* are also the starting point to detect weak points in the infrastructure and accordingly plan risk management instruments like checkpointing or migration [HPC4 08, Birk 06b]. These fault tolerance instruments prevent SLA violations in case of resource failures. This reduces the penalties and, following, the impact of failures. A result is an increased reliability of the grid provider [Djem 06].

Given the possibility to estimate and manage the risk for an SLA violation, providers are eager to increase their profit by selling more SLAs. However, an SLA contains a guarantee for the job's deadline, the time a job must definitely

be finished. To give a guarantee on the deadline, the provider needs to know the runtime of the job. The runtime has to be guessed by the customer.

Given the guessed runtime, modern resource management systems offer the ability to plan resources with an advance reservation. It is a reservation for an amount of r resources for a job n with a duration ω in the time window $[t_{release}, t_{deadline}]$. The advance reservation exactly maps the requirements of SLAs.

If a job exceeds its estimated runtime ω, it will be killed because the resource is planned for other jobs. Thus, the users do not want to loose their jobs and overestimate the runtime. Consequently, the jobs are planned with an overestimated runtime.

Due to the user's overestimations, the jobs tend to end earlier than estimated. This leads to a fragmented schedule and less advance reservations are possible on the resources. Following, reservation based resource scheduling is not optimal for utilization.

Overbooking can cope with the overestimated runtime and exploit the fragmented schedule. A well-done statistical analysis of the ratio of job runtimes to user estimations allows assessing the PoF of running a job in a gap in the schedule. As a result, the overbooking approach allows to sell more SLAs.

The PoF estimation process of AssessGrid is the basis for overbooking in this dissertation. In cooperation with the fee and penalty defined in the SLA, it allows us to estimate the risk for accepting jobs in an overbooked schedule. Thus, when only jobs with a low PoF are overbooked, the number of successfully agreed SLAs will increase as well as the profit of the resource provider [Batt 08d].

1.2.2 Inter Scheduler Negotiation

The second scenario where overbooking is applicable is described by DGSI (*D-Grid Scheduler Interoperability*). DGSI connects DCI Meta-Schedulers through the creation of an interoperability layer [Birk 09b].

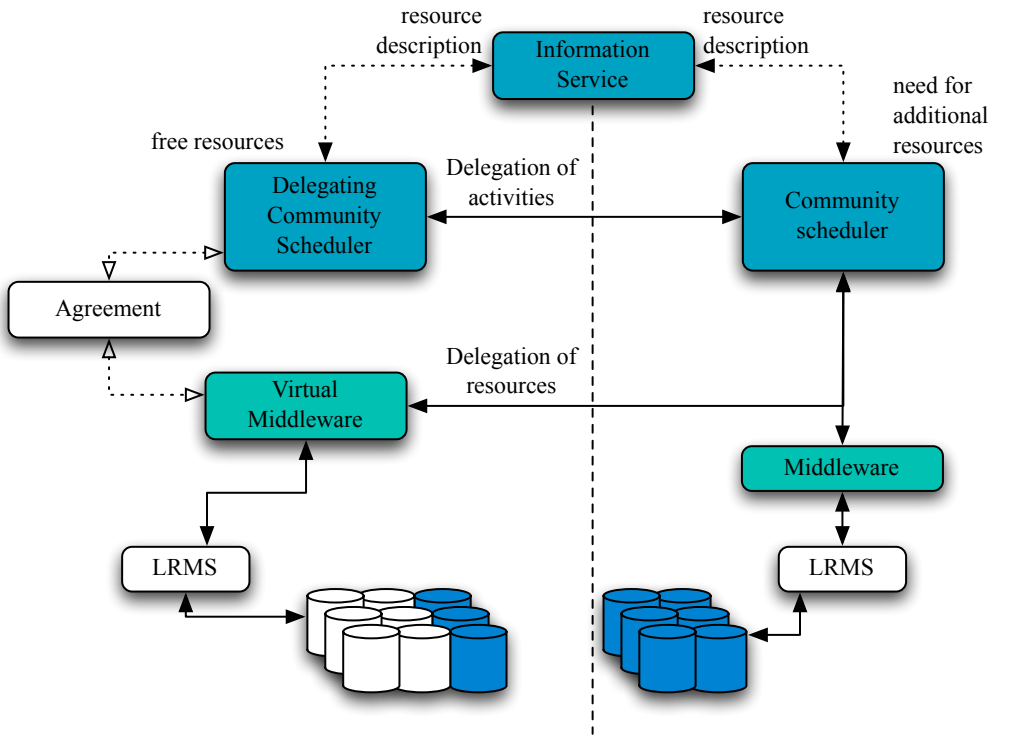
Resource providers generally have instruments to efficiently distribute workload on their resources. This issue is usually described as meta scheduling.

Scheduling is already very complex within one community because the submitted jobs and the available resources can differ. The scheduler has to apply knowledge about usage scenarios and the underlying cluster. This leads to very different, community-specific approaches for the development of cluster scheduling services. The resulting incompatibility of the meta-schedulers, however, is a major drawback for the coordinated cooperation of different DCIs. Nevertheless, if the overall resources should be utilized better, a coordinated cooperation is necessary.

Therefore, two use cases for including foreign resources into the infrastructure arise: Firstly, the need to cover peak demand, and secondly, the usage of specialized resources. These can be vector-based computing systems, astronomical telescopes, or CAVE[2] environments.

[2]Cave Automatic Virtual Environment

Figure 1.5 The DGSI delegation scenarios. Grid schedulers from different domains can cooperate using activity and resource delegation.



DGSI targets these use cases with the conception and development of an interoperability layer for grid level scheduling in DCIs. The aim is to allow the users of a community to distribute their workload among resources within the management domain of another community while keeping their individual, specialized scheduling. DGSI offers new perspectives for community collaboration, resource sharing, efficient utilization, and load balancing.

The two scenarios foreseen within the framework of the project are the delegation of activities and the delegation of resources. The delegation of activities and resources is depicted in Figure 1.5. In the following, the delegation of activities is described in more detail. For the delegation of resources, have a look at [Birk 11b].

Activity delegation means, a meta scheduler hands over a job and the management of its execution to the domain of the scheduler of another community.

The DGSI interoperability layer allows meta-schedulers to exchange single or parallel jobs or workflows. To be interoperable, each meta-scheduler first has to register at the information service. This includes a description of its own resources and execution capabilities.

If a community has an overload of jobs, it can query the capabilities of other communities. The other DCIs can directly take activities and execute them on their resources with their own local resource management system (LRMS).

The other way is to publish a single set of multiple jobs on the interoperability layer. In case of underutilization, another meta-scheduler can query for offered jobs and execute them on its infrastructure.

The activity delegation allows the exchange of workloads for scientific computing centers. This delegation is used when a scientific DCI is overloaded in such a way that it cannot execute the jobs of the working scientists with the estimated runtime before reaching, for example, the deadline of a conference.

The application of overbooking either allows the local meta scheduler to decide that the jobs can, due to the users statistical runtime-overestimation, be finished in time. Alternatively, this meta-scheduler can find another meta scheduler to accept the jobs. With overbooking, the foreign meta scheduler can decide that it is able to accept jobs from other communities, most likely without violating deadlines of the own community members even when the job schedule is overloaded [Birk 11b].

1.2.3 Cloud Computing

The third scenario describes the application of overbooking to cloud computing. Before the emergence of compute clouds, the field of managed distributed heterogeneous resources was lead by the grid.

The idea behind the grids was very interesting, but the decentralized organization and heterogeneity of the cluster infrastructures often had drawbacks to its usability. Further, the organization of the users in several decentralized communities, called virtual organizations (VOs), complicates the rollout and maintenance of the different applications. Some programs had to be licensed and some license models did not allow the usage of resources. Some codes are, of course, open source but even in this case the maintenance is complicated. The codes had to be installed on the different heterogeneous cluster systems. When the codes were installed, the underlying libraries could vary or the installed codes had different versions. While maintenance of the codes even within a project infrastructure is complicated, using other infrastructures further increases the complexity of the mission to know which simulation can be assigned to which resource(s).

Compared to compute grids, clouds have various advantages. Compute clouds add the idea of additionally applying virtualization to the resource management. Clouds are centralized; a single institution is responsible for the underling software and ensures interoperability [Vaqu 08]. Clouds can deliver different levels of services to customers outside the cloud and cloud services can be dynamically configured and delivered on demand [Fost 08, Nurm 09, Soto 08b].

The use of virtualization hides the heterogeneity of the cluster hardware. Following, virtual machines run independently of underlying hardware. As a result, virtual machines are compatible with standard x86 computers. The use of virtualization isolates virtual machines from each other as if they were physically separated. Virtualization encapsulates a complete computing environment in one virtual machine [Barh 03].

There are, of course, several different virtualization technologies available, but interoperability layers, like libvirt [Bolt 10], allow a transparent use of the different virtualization stacks. The system virtualization allows installing software stacks, libraries, and applications encapsulated in virtual machine (VM) images. The user can configure simple images according to his needs and distribute them on every node in the cloud he uses. As a consequence, compute clouds can scale in the number of users, because everyone distributes his or her own software within their own images.

This extension has fostered the usability of clouds so far that the economical impact is enormous. There are IT business providers that offer own clouds, cloud software, or cloud services in between Software as a Service (SaaS), Platform as a Service (PaaS), or Infrastructure as a Service (IaaS) [Nieh 09]. Consequently, the ability to overbook cloud resources is an interesting scenario. Providers like Amazon already applied an overloading of the virtualized resources. Overloading, however, is no challenge for Amazon because no quality agreements are given[3].

Applications in clouds do not always fully utilize their host systems; therefore, overloading on clouds is beneficial. However, applying overbooking to clouds has other constraints as overbooking for not virtualized clusters. Cloud jobs are quite frequently not short running single activities but long running applications (like web services). SLAs are not always bound to a deadline but to requests answered per second [Batt 08a].

Virtual machines are malleable, which means that physical resources can be added or removed. Depending on the kind of the virtualization technology the virtual machine has to be restarted after changing its resources or change of the vm's can be done online. This allows a speedup or slowdown of the single jobs [Wang 10].

In addition, horizontal and vertical scaling of the VMs is possible. Here, vertical scaling means that a collection of independent virtual machines can run parallel on the same host and the number of VMs can vary. Horizontal scaling means that a job can run across several VMs on multiple physical hosts [Rima 09].

In DCIs or high performance computing (HPC) clusters without the possibility to use malleable machines, the overbooking decision is done at occurrence time because a started job will run on this machine. The ability of application checkpointing is not always possible. In clouds, the overbooking can be monitored online and when a deadline is in danger to be violated, the virtual machines can be provisioned with more resources. With virtualization, the overbooking algorithm can intercept the execution to guarantee the success of a specific SLA and assign more resources to this VM or migrate it to a free host [Birk 08b].

1.3 Summary

The remainder of this chapter contains the summary of this thesis. The introduction already motivated the thesis and provided three scenarios for the possible application of overbooking.

[3] Amazon EC2 Cloud Pricing: `http://aws.amazon.com/en/ec2/#pricing`

In the related work in Chapter 2, the commercial fundaments for overbooking are discussed. This includes the construction of service level agreements and a possible electronic market design. Following, related work on machine failures and risk assessment is summarized. Thereafter, theoretical approaches for planning and scheduling are shown and resource management systems and meta schedulers are listed. At last, related work on overbooking and its impact on planning and scheduling is examined.

Chapter 3 defines the overbooking algorithms. The chapter firstly presents placing strategies for new jobs. A conservative, a first fit, a best fit, and a last fit approach are discussed. Following, the overbooking chapter introduces the PoF, a risk based acceptance test, and the two overbooking algorithms.

Then, Probability Density Functions (PDFs) and Cumulative Distribution Functions (CDFs) are introduced because accurate predictions about the runtime estimations are key factors to be profitable. A PDF describes the probability that a job ends after exactly $x\%$ of its estimated runtime. A CDF describes the probability that the real runtime of a job with an assigned PDF will be less or equal to $x\%$ of its estimated runtime. A joint probability density function combines the PDFs of several jobs. It is defined how the joint PDF is calculated and how it can be used to estimate the probability of failure for overbooked jobs.

The *comprehensive overbooking* was designed to support a very accurate PoF calculation. It is able to calculate a job's PoF based on node specific failure predictions and the previously scheduled jobs. Three failure models based on a Weibull Failure model, a Hyper-Exponential Failure model, and a Poisson process are discussed. They allow a very accurate failure prediction. Resulting, a reliable overbooking process is supported. The disadvantage of the comprehensive overbooking is that the algorithm has to map the jobs on resources at SLA negotiation time. Until the start time, a better choice of resources is often possible. Additionally, the calculations are very complex and require a long runtime.

The *heuristic overbooking* should overcome this disadvantage. It does not decide where the job should run at SLA negotiation time. Instead, the heuristic can start jobs on any free resources. Additionally due to the shorter runtime, replanning is possible after a job ended. The heuristic overbooking uses an overall resource count to calculate the probability of a job's success. This approach is designed to be more flexible because a job can run on every node, but this strategy does not remember the jobs that ran before. Therefore, the heuristic cannot include this possibility into the PoF calculation and its PoF estimations are not as accurate as those of the comprehensive approach.

Following to the definition of the probability of success calculation for overbooking, the applied planning strategies are described. The four strategies are the heuristic planning approach, the heuristic overbooking approach, the comprehensive backfilling approach, and the comprehensive overbooking approach. The conservative backfilling plans jobs on their resources with the full estimated runtime. The comprehensive overbooking works on the same basis but the probability of an earlier job end can be included and jobs can be overbooked. Heuristic planning counts resources and checks whether or not a job can be executed with its

requested resources and full estimated runtime. The heuristic overbooking works similar but can assign less runtime to a job according to the heuristic overbooking calculation.

Finally, market mechanisms are described where overbooking could be applied. The considered market mechanisms are auctions, bidding or tendering, bartering models, and commodity markets.

In Chapter 4, the statistical analyses about the runtime estimations are shown. They provide the basis for estimating the probability of a successful overbooking approach. An exemplary statistical analysis of user estimations of traces is shown. At first, a runtime analysis can be used to reflect that users often submit the same application with similar input again and again. Secondly, the analysis of correlations between the resource consumption of jobs and the estimation accuracy is the basis of another statistic that can be beneficial for overbooking. Thirdly, a runtime analysis is done on a user basis. At last, an application-oriented statistical analysis is shown.

The chapter is completed by a user survey revealing how the users of the Paderborn Center for Parallel Computing (PC^2) estimate their job runtimes.

The implementation of the simulation environment is shown in Chapter 5. First, the used parameters are discussed. Basis for the simulation is the job information that was retrieved from the job-traces. Most attributes and parameters were directly extracted and used. However, some attributes had to be adjusted because the simulations need an overload of jobs. In addition, some attributes had to be created because they are missing in academic environments. This is information about the fee and penalty for a job, and the job's deadline.

Then, the architecture of the simulation environment is shown. The description includes the used data structures and the event system. It covers tasks to handle the job occurrence, job start, and job end, resource outages, and repair. The last discussed aspects are the implemented placing routines first fit and best fit.

Chapter 6 contains the results of the evaluation. It analyzes the abilities of the overbooking strategies based on job traces from the Parallel Workload Archive. Six traces of the archive were selected for the evaluation. To reveal the benefit of the overbooking approach, the four scheduling strategies heuristic planning, heuristic planning with overbooking, comprehensive backfilling, and comprehensive backfilling with overbooking were applied.

To get hints about the abilities of the underlying statistics, two different statistical sources were defined for the simulation. One source is a statistical analysis about the ratio of the estimated to real duration of a job. The other source is a statistical analysis about the quality of the runtime estimation for different groups of jobs. The jobs are assembled according to their amount of requested resources.

To evaluate whether the overbooking would be profitable for computational markets and/or just resource exchange, two different acceptance tests were applied. The first acceptance test uses the PoF threshold directly. This allows having an acceptance test for academic environments. In academic environments, the researchers do not have to pay of using a cluster. The second acceptance test bases

on a risk assessment and simulates the impact of overbooking to computational markets.

In Chapter 7, the simulation is extended to two further kinds of statistical input functions based on the used applications and individual users. This allows getting a clue about applicability of a broader area of statistical analyses. The simulations show that the comprehensive overbooking is more trustworthy and reliable. The additional gain of this overbooking mechanism compared to the underlying, not overbooking, strategy is higher. However, the simulations also show a very time consuming runtime for the comprehensive overbooking. The heuristic overbooking has a higher utilization and profit than the comprehensive overbooking and is much faster to calculate. The higher profit is a result of the better performance of the underlying not overbooking heuristic scheduling strategy that results in a higher utilization than the comprehensive backfilling.

The minimum additional gain with overbooking in the simulations was 10%, and the maximum additional gain was up to 94% in the simulations with the PoF acceptance test and up to 234% with the risk acceptance test.

The quality of the underlying statistics is dependent on the quality of the selection of the jobs that are gathered as source. However, due to the different PoF thresholds that were applied by the simulations, a similar peak performance was achieved with many different statistics.

The last part of this thesis considers future work. Its subjects are improved monitoring of jobs, the use of commercial job traces for more appropriate statistical inputs, the application of advanced reservations, the search for correlations between user estimation accuracy and other job parameters, the application of system generated runtime predictions, the introduction of checkpointing and migration, and the extension of overbooking to virtualization and cloud computing.

2 Related Work

The related work presents the technical basics in the research field of negotiation, risk assessment, scheduling, and overbooking. First, it discusses the commercial fundaments for overbooking. This includes the construction of contracts, called service level agreements for a possible electronic market. Second, related work on machine failures and risk assessment for compute clusters and resource providers is introduced. Third, theoretical approaches for planning, followed by scheduling strategies, and scheduling systems are described. At last, related work on overbooking and its impact on planning and scheduling is discussed.

Contents

2.1 Negotiation and Market Mechanisms 15

2.1.1 Service Level Agreements 16

2.1.2 Market Mechanisms 18

2.2 Risk Assessment and Management 20

2.2.1 Failure Data Analysis and Modeling 20

2.2.2 Risk Calculation 20

2.2.3 Risk Assessment 21

2.2.4 Risk Management 22

2.3 Scheduling and Planning 22

2.3.1 Packing Theories 23

2.3.2 Cluster Scheduling Strategies 26

2.3.3 Scheduling and Planning Systems 27

2.4 Overbooking 30

2.4.1 Overbooking in General 30

2.4.2 Overbooking of IT Systems 31

2.1 Negotiation and Market Mechanisms

The related work begins with a summary of related work on the construction and negotiation of SLAs and market mechanisms.

2.1.1 Service Level Agreements

A key requirement for applying overbooking are contracts that contain the negotiated service quantity and quality in combination with agreed guarantees. Commonly, such contracts are known as service level agreements (SLA)s. The additional requirement for SLAs is the ability to be electronically negotiable.

Jobs without guaranteed deadlines and without negotiated minimal service quality are operated in a best effort manner. Here, overbooking can be applied without any risk for the provider. The provider will do its best to execute the job if no other more important task has to be done. Eventually and probably in the far future, the jobs will be completed. Therefore, this related work focuses on the interesting part, SLAs with assigned deadlines and QoS demands.

Typically contents of SLAs are defined as *Purpose, Parties, Validity Period, Scope, Service Level Objectives SLOs, Service Level Indicators, Penalties and Operational Services* in [Saha 03].

SLAs are well-established means in the industry. Industry standards like the IT Infrastructure Library (ITIL) define SLAs and the handling for commercial contracts [Bock 08]. However, the SLAs used by the industry are often based on paper contracts. For on demand SLA negotiation, the assumed scenario for this dissertation, standards like ITIL are not sufficient. Instead, automatically negotiable electronic contract standards, like the WS-Agreement specification from the Open Grid Forum (OGF), have to be used [Andr 04b].

The WS-Agreement defines how an SLA is structured. A WS-Agreement based SLA consists of the agreement context describing the agreement terms, the agreement template, and the creation constraints. The agreement context describes the handling actors and who the service provider or customer is. The QoS guarantees are defined by two kinds of agreement terms. First, the service description terms (SDT) describe the guaranteed contents, like provided resources, start time, or deadline. Secondly, the guarantee terms (GT) are connected to the service description terms and define the fee for a contract conform service provision and the penalty for violating an SDT.

The agreement template and creation constraints describe the constraints that have to be redeemed during negotiation.

While the WS-Agreement standard describes how an SLA is structured, the WS-Agreement extension called WS-Negotiation defines the negotiation procedures [Zieg 08].

Other related work on SLAs for distributed computing concerns, for instance, the SNAP protocol for negotiating service level agreements and coordinating resource management [Czaj 02], or the IBM defined Web Service Level Agreement (WSLA) specification [Ludw 03].

For the WS-Agreement specification, several implementations, like WSAG4J, exist [Wael 11]. Two surveys located SLA implementations in the following projects [Park 08, Wied 08].

Table 2.1: *SLA implementations by different projects*

Project	Description
AgentScape	AgentScape provided mobile agents to access Internet based heterogeneous compute resources. The WS-Agreement based SLA negotiation layer allowed QoS definition for virtual domains [Moba 06].
Akrogrimo	Akrogrimo defined a mobile grid architecture for connecting businesses with costumers. WS-Agreement was used to define the QoS aspects in the SLAs [D An 06].
ASKALON	ASKALON was designed to simplify the development and optimization of grid applications, based on SOA concepts. ASKALON used the GridARM package for SLA negotiation [Fahr 07].
AssessGrid	AssessGrid implemented a risk assessment and management layer for grids and added PoF estimated values to SLAs (based on WS-Agreement) [Batt 07b].
Bazaar	Bazaar created a tool for transparent SLA-based resource allocation. The SLA layer was able to filter appropriate resources and identified missing resources [Baza 10].
BEinGRID	BEinGRID was an EC founded IP project that included several sub-projects with individual SLA implementations [Dimi 07].
BREIN	BREIN integrated multi-agent and semantic web concepts for eBusiness. The project developed a framework to provide and sell services with different resource types [Park 08].
CATNETS	CATNETS allowed a decentralized, self-organized mechanism for resource allocation. The project developed an own bidding language for the SLA negotiation [Eyma 07].
DGSI	The D-Grid project DGSI used the WS-Agreement to implement an interoperability layer in between grid meta schedulers [Birk 09b].
JSS	The Job Submission Service (JSS) is a grid broker that minimized the total time of delivery for an individual job submission. The WS-Agreement based SLA negotiation could directly be connected to Maui [Elmr 05].
NextGRID	NextGRID aims to enable new businesses in a grid. Therefore, SLAs in between the actors were a key-concept [Hass 07].
PHOSPHORUS	PHOSPHORUS implemented co-allocation for computational resources and network resources with a required QoS on Service Level Agreements [PHOS 10].
SLA4DGrid	SLA4DGrid used the WS-Agreement to realize a Service Level Agreement layer for the German grid infrastructure [SLA4 11].

Table 2.1: *SLA implementations by different projects (con.)*

Project	Description
SLA@SOI	SLA@SOI developed a framework for SLAs in Service Oriented Infrastructures (SOI). Focus lay on research and development for entire SLA architectures for SOIs. This included a technical framework [SLAS 11].
SmartLM	SmartLM focused on negotiation and co-allocation of software licenses and grid resources with respect to job submission and execution for license-protected applications [smartlm 10].
SORMA	SORMA investigated the application of computational and related resources for a market infrastructure. Aim was to bring potential consumers to fitting resource providers. In addition, SORMA allows asynchronous auctions [sorma 10].
TrustCoM	TrustCoM developed a framework for trust, security, and contract management in between the dynamic evolvement of virtual organizations (VOs). The WS-Agreement based SLAs allowed negotiation, monitoring, and application of different performance levels [Wils 07].
VIOLA	The VIOLA Meta Scheduling Service (MSS) was designed as testbed for multiple partners. It included an own WS-Agreement based SLA framework for job negation [Barz 07].

2.1.2 Market Mechanisms

Pricing in DCI environments is a topic of many research papers. The paper *The Grid economy* provides a good summary [Buyy 05]. It shows the use of different economic models for trading resources in different application domains. Other interesting approaches define general strategies for the service market place [McKe 07] or create a prediction-based enforcement of performance contracts [Sand 07]. Typical trade goods are CPU cycles, storage space, database queries, or distributed computing.

The pricing strategies for computational markets follow several different strategies. Some approaches offer fixed prices for the services, other approaches have different prices for resources, based on environmental attributes such as the availability of larger main memory or higher CPU speed, and again other approaches adapt the prices depending on the specific demands [Buyy 05].

The most applied models are:

- Auctions,
- Bidding and Tendering,
- (Cooperative) Bartering Models, and
- Commodity Models (Markets).

The most frequently investigated topics of research in the field of DCI commercialization are auctions, biddings, and commodity models. *Auctions*, like on

eBay[1], are the most frequently used instruments. They fit best to the needs of supply and demand. There are different approaches: *English-, Dutch-, Double-*, or *Vickrey* auctions. The customers can offer money for a job execution, and the highest money or first offer wins the auction.
English auctions are open auctions where the participants have to raise bids to outbid other parties. After each round, the highest bid is shown, and then a new round is started. End of the auction is when no higher bid is given [Nisa 98].
In *Dutch auctions*, the auctioneer starts with a high price for the item and lowers it continually until the first bidder accepts the price [Comm 91].
Double auction is a system of asks and bids. Sellers ask a price for their resources and buyers make bids for the same. The auctioneer matches up corresponding asks and bids [Pada 03].
In the above-explained *normal* auctions, every participant knows the value of a resource and tries to maximize the income for himself. This means he might bid lower than the estimated value. Thus, lying about the estimated value of an item is profitable.
In *Vickrey auctions*, the winner of the auction pays the value of the second highest bid. Thus, everybody can bid the price he or she estimates for the item and can still hope to get an advantage. Lying is not profitable [Wald 92].
Bidding or *Tendering are reverse auctions. Buyers are defining the kind of service they want to have and the price they want to pay. A price offer is called a bid and the providers may accept the bids* [Ston 94]. Bidding often is a competitive process of setting a price one is willing to pay for something. The term is used in context of auctions, stock exchange or card games. The approaches have to seek for prices and terms for a particular job in common, which are carried out under a contract [Lali 00, Reed 99, Bred 98, Eyma 07].
In *(Cooperative) Bartering Models*, resources are traded; for example, storage space is exchanged for CPU time [Buyy 05].
Commodity Markets are markets where the prices for goods traded are given by a formula. Factors like production of goods, consumption, and transfer costs influence the price. The formula is a function over $f(j,t)$ where the resource owner receives $f(j,t)$ award for completing job j in time t [Amir 98].
Brooke et al. describe standardized functions where the cost for a job or service is based on the quality or quantity of the service [Broo 00]. For instance, a fast resource might have a high price, and a slow resource has a low price. The function then increases the price when the resources are running low or decreases the prize when the resources are unused.
Commodity markets are used in several grid projects, like:

- Mungi [Heis 98]
- Enhanced MOSIX [Amir 00]
- Nimrod/G [Buyy 00]
- Grid Sim [Buyy 02]
- G-Commerce [Wols 01]
- Gridbus [Buyy 04]

[1] eBay: `http://www.ebay.de/`

Overbooking might be used in every market mechanism. In auctions, more resources could be offered than available, bidding could be exploited with overbooking, or the functions of commodity markets could include overbooking.

2.2 Risk Assessment and Management

Here, related work on risk assessment and risk management is presented. A risk assessment process bases on high-quality knowledge of the underlying failure vulnerabilities. This section starts with an explanation how risk can be defined and transfers this knowledge to be used as input for the related work on risk assessment and management.

2.2.1 Failure Data Analysis and Modeling

Failures of job-executions have many different possible causes. Jobs die due to bugs in their source code, because the input is missing or does not fit to the program/version specifications, because of wrong parameters, or because of crashes of the underlying hardware.

Therefore, many research papers have focused on failure data analysis and methods to model resource stability estimations. Related work stated that many problems that lead to a job failure are not detectable from logs (36% - 58 %). The detected problems that lead to outages are often software related or hardware failures (6% - 16%) [Kaly 99]. In addition, rebooting a machine does not always help, and failures are propagated and correlated [Xu 99]. For large cluster systems Schroeder and Sahoo claimed that crashes are correlated and bursty [Schr 06, Saho 04]. Amrit et al. developed a methodology to describe software reliability [Goel 85].

Several papers worked on modeling failure curves. Iosup et al. and Nurmi et al. stated that failure-rates of large clusters follow Weibull-distributions [Iosu 07, Nurm 05]. The same result was shown for Windows NT clusters [Xu 99] and Internet services [Heat 01]. In contrast to the papers mentioned before, evaluations on other kinds of computing infrastructures found hyperexponential distributions to model failure-rates best. For example, hyperexponential distributions were best for grid or wide area computing environments [Nurm 05], for an evaluation of workstations [Mutk 87], and operating systems [Lee 93]. Lee also found that Markov chains are a good description of the failure nature [Lee 93].

2.2.2 Risk Calculation

The risk for computing providers consists of hazardous events that potentially adversely affect a provider's ability to ensure that an SLA is fulfilled. Risk can be characterized using two key parameters: the probability of occurrence and the impact of occurrence [Birk 07].

Consider a node outage affecting a compute resource on which a job is running. In order to evaluate the PoF, the provider must take the possible causes and their probability into account. A node outage, for example, could be caused by a power cut, a system crash, or a hardware failure. Each of these events must be taken into consideration in order to enable a calculation of the probability of occurrence [Djem 06]. The resulting PoF is a real number between 0 and 1. This PoF has to be multiplied with the SLA's penalty to know the risk. However, risk can also be a positive force. The opposite of risk is opportunity; a chance of winning at an event. It is defined as the probability of success multiplied by the value of the event. The PoS is always 1 - PoF.

2.2.3 Risk Assessment

AssessGrid [Asse 08] introduced the main instruments for assessing the probability of a job failure. It addresses the issue of risk assessment and risk management [Hove 05] at all DCI layers. This includes risk awareness and consideration in SLA negotiation [Batt 07b] and self-organization of fault-tolerant actions [Djem 08].

Risk assessment is usually defined as the process of finding possible vulnerabilities. When vulnerabilities are found, the probability that they occur and the associated risk is calculated [Benn 96, Stew 04]. Lichtenstein has identified relevant requirements and factors for risk assessment methods like costs, influences, structures, and levels of risk [Lich 96].

Very important sources of vulnerabilities are machine failures. For instance, Schroeder [Schr 06] and Sahoo [Saho 04] showed that machine crashes in cluster systems are typically busted and correlated.

Risk assessment can be either quantitative (i.e., producing a numerical assessment) or qualitative [Majl 06] (i.e., producing a verbal assessment or showing traffic lights). Some quantitative risk assessment techniques are Monte-Carlo simulations, fault and event tree analysis, sensitivity analysis [Whit 95], annual loss expectancy [Rain 91], risk exposure [Boeh 89], and effects analysis [Whit 95]. Qualitative techniques are scenario analysis [Rain 91] and the fuzzy set theory (FST) [Rain 91].

In the area of software development, research on risk assessment methods has been introduced [Benn 96]. Used methods are probabilistic risk analysis, Fault Tree Analysis (FTA), and Failure Mode and Effect Analysis (FMEA). They are used in design and testing of software products as well as in the identification of unsafe states [Leve 86]. Känsälä [Kans 02] presents a quantitative method for risk assessment in software project development. The method supplements traditional software cost models with risk contingency capabilities. Ngai and Wat [Ngai 05] used FST and developed a fuzzy decision support system for risk assessment in eBusiness development [Voss 08, Carl 08].

The results allow resource providers to assess risk and end-users to know the probability of an SLA violation in order to accurately compare provider's SLA offers. With such knowledge, end-users can make appropriate decisions in relation

to acceptable costs and penalty fees since these also have a potential impact on their own business. The provider can plan jobs on its resources and assess the PoF of these jobs. To reduce this basic PoF, appropriate fault tolerance mechanisms can be used or the provider can decide to not accept an SLA that is too risky.

In this dissertation, a risk assessment process close to the probabilistic or risk exposure is chosen [Boeh 89]. Here the risk can be assessed as PoF of the job multiplied by the penalty of the failed SLA.

2.2.4 Risk Management

The last point, following to the risk estimation is the application of measures to reduce the risk calculated by the risk assessment. This is the task of the risk management. It plans fault tolerance (FT) mechanisms [Birk 06b].

Risk management in this scope means that for a job with a high risk, a simple FT mechanism might be to execute the job twice on different machines. When one execution fails, the other execution can still be successful and provide the results in time. Both executions have to fail to violate the SLA. Following, a double execution reduces the probability of failure of the job. However, this procedure is resource intensive.

Therefore, an improved FT mechanism is the use of checkpointing. This allows restarting a job, after a node failure, not from the beginning but from the last checkpoint. Therefore, provider internal checkpointing reduces the risk. However, when complete cluster systems fail, local checkpoints do not help, at least until the resources within the provider are available again.

Further, checkpointing and migration of jobs to other foreign providers is a sophisticated means to reduce the risk of a resource provider. This requires a fee that has to be paid to the foreign scheduler for the execution of the migrated job, but the migration costs may be lower than the penalty for the SLA violation [HPC4 08].

To summarize, a risk management layer reduces the probabilities of failure for a job or the penalties for an SLA and, following, the impact of failures [Djem 06].

2.3 Scheduling and Planning

This dissertation distinguishes two scheduling approaches, queuing and planning. When a resource becomes free, a queuing system selects a fitting job from its queues according to a given strategy. There can be several queues for short, medium, or long running jobs. They have different priorities and wall times. A wall time describes the maximal duration a job can run. Planning means that the waiting jobs are not held in a queue but in a plan of the cluster system. The plan of the cluster holds all jobs with requested resources and assigned start- and end-times [Hove 03].

Queuing based strategies are well examined and established. However, planning strategies are better suited when deadlines are supported because the planned jobs

are connected to a specific estimated runtime. In queuing systems, there is no job bound runtime estimation, only the queue related wall time. Thus, queuing systems allow to calculate the latest start of a job, but the results are calculated based on the more general wall times. This usually leads to more imprecise job start estimations of queuing compared to planning.

In scheduling, the makespan is the time difference between the start and finish of a sequence of jobs or tasks. A minimal makespan is a typical aim of scheduling strategies.

Scheduling distinguishes between online and offline strategies. In offline strategies, all jobs are known before the calculation. In the online case, the jobs occur during the job execution and have to be included into the schedule. Online scheduling is called x-competitive if for any sequence of jobs, it produces a schedule with a makespan of at most x times the makespan of the optimal schedule [Huri 08b].

2.3.1 Packing Theories

Overbooking of cluster resources focuses on cluster systems with several nodes. Therefore, two-dimensional (resources $\times$ time) planning algorithms were further investigated. The theoretical approaches for planning jobs base on the evaluation of two-dimensional packing. Exemplary strategies for planning jobs on a single resource are for example First-Fit (FF) or Next-Fit (NF) [John 73, John 74a]. They will not be considered further. Instead, this section will begin with a short overview of bin packing and proceed with strip packing approaches that correspond to cluster planning.

2.3.1.1 Bin Packing

In bin packing, a number of rectangles have to be placed in an unlimited number of identical rectangular k bins with a given width and height [Lodi 02a]. Goal is the minimization of the used number of bins. Packing of bins $(k \geq 2)$ is NP complete and optimal packing is NP-hard [Gare 79, Jans 10]. However, simple Bottom Left (BL) algorithms approximate the results in $O(n^2)$ [Chaz 06].

The performance of the algorithms can be given as an asymptotic performance or as an absolute performance. The asymptotic performance ratio is defined as:

$$\inf\{r \geq 1 | \text{ for some } N > 0, \frac{C^A(L)}{C^*(L)} \leq r \text{ for all lists } L \text{ with } C^*(L) \geq N\},$$

where $C^A(L)$ is the number of bins the packing of algorithm A used, and $C^*(L)$ is an optimum number of bins for a packing of L. The absolute performance ratio for A is defined as [Xia 10]:

$$\inf\{r \geq 1 | \frac{C^A(L)}{C^*(L)} \leq r \text{ for all lists } L\}.$$

For one-dimensional bin packing with best-fit (BF), Simchi-Levi shows that the best possible polynomial time approximation is $\frac{3}{2}$ [Simc 94]. Xia and Tan showed that the absolute performance ratio of the first fit (FF) is $\frac{12}{7}$ [Xia 10]. Further, Minyi showed an asymptotic performance ratio for FF of $\frac{11}{9}$ [Yue 91]. Johnson et al. show that best fit will not use more than $\frac{17}{10}$ times the optimum $+2$ bins. If the input is sorted, $\frac{11}{9}$ times the optimum $+4$ bins are sufficient for FF and BF [John 74b]. When restrictions to the dimension of rectangles are given, approximation algorithms were able to reduce the performance ratio to $(1 + \epsilon)$. Exemplary restrictions are for instance that the number of bins has to be larger than a minimum or the rectangles have bounded dimensions [Karm 08, Vega 81]. However, as long as no restrictions are given it has been shown that for all $\epsilon > 0$, bin packing is without restrictions NP-hard to approximate within $\frac{3}{2} - \epsilon$ [Vega 81].

2.3.1.2 Strip packing

Planning-based resource management systems (RMS) are special applications of packing. Contrary to bin packing, it is not the aim to reduce the number of bins. Strip packing does not have any bins but one *strip*. The width of the strip is often defined as the number of nodes generally available, and its height equals the time. The total usage time for an arbitrary number of jobs does not end. Thus, the strip has an, in principle, infinite height. Jobs are considered as rectangles having a width equal to the number of required resources and a height equal to the execution time determined by the user. The rectangles have to be positioned on the strip in such a way that the distances between rectangles are minimal, and jobs must not overlap each other. Since strip packing is an NP-hard problem [Bake 80], several algorithms have been developed that work with heuristics and are applicable in practice. Ntene and Lodi et al. give a good overview of strip packing algorithms [Nten 07, Lodi 02a]. Strip packing algorithms are either online or offline algorithms. An offline algorithm has information about all jobs to be scheduled a priori, whereas online algorithms cannot estimate which jobs arrive in the future. The approaches could be divided into several main areas: bottom-left algorithms, which try to put a new job as far to the bottom of the strip and as far left as possible, level-oriented algorithms [Coff 80], split algorithms [Coff 80], shelf algorithms [Bake 83], and hybrid algorithms that are combinations of different placing strategies.

Bottom-left algorithms
The Bottom-left algorithms place each rectangle as close to the bottom of the strip and as far to the left as possible [Jako 96]. Such algorithms do not need a sorted list of rectangles and can be used in an online manner. Baker et al. proved that bottom-left algorithms pack in a ratio of three to the optimum [Bake 80] and require $O(n^2)$ time [Asik 09].

Level-oriented algorithms
Level-oriented algorithms sort rectangles in order of decreasing height; accordingly, all jobs have to be known a priori, and such algorithms are offline mechanisms [Coff 80]. The packing is performed in a series of levels. The bottom of the first level is the bottom of the strip, and the bottom of the level $n + 1$ is the top of

the highest rectangle in the level n. Next fit decreasing height (NFDH) [Coff 80] means that the next-fit approach is used to pack the sorted list of rectangles. The rectangles are packed left-justified on a level until the next rectangle does not fit. This rectangle is used to define the next level where the packing is continued. First fit decreasing height (FFDH) [Coff 80] means that each rectangle is placed on the first (lowest) level it fits on. If no level is available, a new one is introduced on the top. Best fit decreasing height (BFDH) puts a rectangle on the level that minimizes the unused horizontal space [Lodi 02b].

Coffman, et al. proved that NFDH packs ≤ 2 times the optimum and FFDH is $\leq$ 1,7 times the optimum +1 [Coff 80]. FFDH and NFDH have a runtime in $O(n \log(n))$ [John 74a].

Shelf algorithms
Shelf algorithms are similar to the level oriented approach but do not require a sorted list of rectangles [Bake 83]. Consequently, these are qualified as online algorithms. Online algorithms deal with jobs that are not known in advance but occur with a delay called release time [Feit 05]. The height h of the levels does not depend on the highest rectangle but on a fixed parameter $r \in [0, \ldots, 1]$. The shelf size grows in the form of $r^{k+1} \leq h \leq r^k$, i.e., for small r the range between heights is large. This would be sufficient for small sets of rectangles. If r is close to one, the difference between the heights is very small. This is sufficient for huge amounts of jobs where the jobs in one shelf have similar heights. Next fit shelf (NFS) is an approach in which a job is packed on the highest fitting shelf that has the required height. If no shelf is available, a new one of the given height is introduced into the system; the job will be placed there. First fit shelf (FFS) is an approach that places a job in the lowest shelf of the fitting height. Csirik and Woeginger showed that the shelf algorithm has an asymptotic competitive ratio close to 1.691 [Csir 97].

Han et al. compared strip packing to bin packing and showed that any offline packing algorithm can be applied to strip packing with an asymptotic worst-case ratio. The upper bound of online strip packing was improved to an asymptotic competitive ratio of 1.58889 [Ye 07]. Csirik and Woeginger show a summary of online packing algorithms [Csir 98] .

Split algorithms
For split algorithms, the rectangles are sorted by the width and the strip is split vertically into smaller open ended strips depending on the width of the rectangle [Coff 80]. This approach could be used with shelf or level-oriented algorithms.

Hybrid algorithms
Combinations are known as hybrid algorithms when two or more types of the above mentioned algorithms are used. Hybrid first fit (HFF) combines a strip packing FFDH strategy with a following finite bin packing algorithm. HFF packs in $\frac{17}{8}$ times the optimum $+5$ [Chun 82]. Finite Best-Strip (FBS) [Berk 87] is a variation of HFF using BFDH.

Sleator [Slea 80] presents an approximation algorithm that packs rectangles in 2.5 times the optimum. Brown shows that the lower bound of the bottom-left algorithm is $\frac{5}{4}$ for even a sorted list [Brow 80]. Breaker et al. show an up-down

(UD) algorithm that packs rectangles to the height of $\frac{5}{4}$ to the optimum [Bake 81]. Hoyland provides an algorithm with $\frac{4}{3}$ performance ratio with an algorithm that is allowed to cut rectangles horizontally [Hoyl 88]. Steinberg [Stei 97] and Schiermeier [Schi 94] show algorithms that decrease the lower bound by up to 2 times the optimum height. Kenyon and Rémila invented an algorithm that reduces the lower bound to $(1 + \epsilon)$ to the optimum solution and runs in polynomial time [Keny 00, Keny 02]. Their work is based on Fernandez and Zissimopoulos who also show an approximation algorithm for problems where the size of the rectangles is bound to an absolute maximum size. This algorithm also gets a result of $(1 + \epsilon)$ compared to the optimum [La V 98]. Johannes analyzes a deterministic list-scheduling algorithm. He proves a lower competitive ratio of 2.25 compared to the optimal makespan for online algorithms [Joha 06]. Hurink and Paulus [Huri 08a] improved the result to an absolute competitive ratio of $\frac{7}{2} + \sqrt{10} \approx 6.6623$ if the job length is limited. Ye et al. showed that the border also holds without restriction to the job length [Ye 09]. Further, Hurink and Paulus showed that the competitive ratio has a tight lower bound of 2 for any number of machines [Huri 08b]. Chan et al. showed that the competitive ratio can be improved to $1 + \sqrt{\frac{2}{3}}$ for two machines [Chan 08]. Naroska and Schwiegelshohn proved that parallel jobs can be scheduled online with a competitive factor of $2 - \frac{1}{m}$, where m is the number of machines [Naro 02]. For further background on online scheduling see [Pruh 03].

In this dissertation, a placing strategy similar to bottom left first is applied.

2.3.2 Cluster Scheduling Strategies

In practice, compute jobs are connected with deadlines or have machine dependent constraints. Therefore, cluster scheduling has been a separate research area.

Feitelson et al. have written a survey on theory and practice of parallel job scheduling [Feit 97a]. In contrast to packing algorithms, not only the absolute or asymptotic performance ratio counts but also, depending on the tasks, the minimization of the makespan, the maximization of throughput, the minimization of the waiting time, or the minimum average response time. The makespan describes the time that elapses from the start of the first job to the finish of the last job [Joha 06]. Maximization of throughput means that as many jobs as possible should be finished in a given time [Bar 09]. The waiting time is the time the jobs are waiting from submit to start [Tsaf 07]. The average response time is the time from job submit to finish [Ture 94].

Many scheduling strategies for cluster systems are still based on first-come first-serve (FCFS) [Hams 00]. FCFS guarantees fairness but leads to a poor system utilization as it might create gaps in the schedule.

Backfilling, in contrast, was developed to increase system utilization and throughput [Feit 97b]. It does not have to schedule a new job at the end of a queue but is able to fill gaps in case a job fits in. The additional requirement for the ability to use backfilling is an estimation of the runtime of each job. The *EASY* (Extensible Argonne Scheduling sYstem) backfilling approach can be used to further improve

system utilization. Within EASY, putting a job in a gap is acceptable if the next job in the queue is not delayed [Feit 97b]. However, EASY backfilling has to be used with caution in systems guaranteeing QoS aspects since other jobs in the queue might be delayed.

Therefore, Feitelson and Weil introduced the conservative backfilling approach, which only uses free gaps if no previously accepted job is delayed [Feit 98]. Simulations show that both backfilling strategies help to increase overall system utilization and reduce the slowdown and waiting time of the scheduling system [Mual 01]. The work also shows that the effect of the described backfilling approaches is limited due to inaccurate runtime estimations.

Several papers analyze the effect of bad runtime estimations on scheduling performance. An interesting effect is that bad estimations can lead to a better performance [Zotk 99]. Tsafrir shows an approach to improve scheduling results by multiplying a fixed badness factor to the user-estimated runtimes [Tsaf 06].

Effort has been made to develop methods to cope with bad runtime estimations. Several approaches tried to automatically predict the application runtimes based on the history of similar jobs [Gibb 97, Smit 98, Tsaf 05]. Tsafrir et al. present a scheduling algorithm similar to the EASY approach (called EASY++) that uses system-generated execution time predictions and shows an improved scheduling performance for job waiting times [Tsaf 07]. The approach shows that automatic runtime prediction can improve backfilling strategies.

Scheduling strategies for parallel processes also need special placing strategies for parallel jobs. Placing strategies for parallel jobs are useful if network topologies provide connections with lower latency for nearer nodes. In this case, the node allocations have to be considered within scheduling. Krueger et al. [Krue 94] and Mohapatra et al. [Moha 93] developed scheduling schemes for hypercube clusters. Lo et al. show a strategy for mesh-connected clusters [Lo 97]. Subramani et al. developed a buddy algorithm [Subr 02].

This thesis focuses on scheduling approaches that try to overcome best effort and plan jobs to hold their deadlines on distributed compute infrastructures. Deadline scheduling for client server systems in computational grids was introduced by Takefusa et al. [Take 01]. The scheduler estimates the runtime and data transfer time based on traces and the current background load of the servers. Aim of the scheduler is to minimize the overall occurrences of deadline misses and the magnitude of the misses [Take 01]. Load balancing based on the current background load and fallback mechanisms for wrong decisions should further increase the performance of the system. Caron et al. improved the previous mentioned approach for the multi-client multi-server case [Caro 04].

2.3.3 Scheduling and Planning Systems

There is a number of well-developed scheduling systems available. They prove the general applicability of scheduling in practice. The next two sections describe different groups of scheduling systems. The first topic are resource management systems (RMSs) and schedulers that distribute jobs in between clusters and

the second topic are cluster management systems for virtualization and cloud software.

2.3.3.1 Resource Management Systems

The tasks of a Resource Management System (RMS) comprise the planning and distribution of the users' jobs on the resources. The ability of the local RMS (LRMS) to utilize its cluster as far as possible is a very important factor of efficiency. Consequently, there have been many projects that developed special LRMSs and many resource management systems are available. While some have specialized abilities, all have the ability to schedule jobs on a system and decide where the job should be placed. In the following, the most important RMSs are summarized.

The portable batch system (PBS) or openPBS is a frequently used resource management system [Bayu 99]. PBS Pro [Nitz 03] is the commercial version. OpenPBS is also the basis for the Torque [Ress 11] RMS. The Load Sharing Facility (LSF) is a commercial scheduling product from Platform computing, an IBM company [Zhou 92]. The computing center software (CCS) or (openCCS) is an invention of the PC^2 [Kell 01, Hove 03]. The LoadLeveler is an IBM product to manage serial and parallel jobs over server clusters [Kann 01]. SLURM is an open-source, highly scalable resource manager designed for Linux clusters of all sizes [Yoo 03]. The GridEngine, formally known as Sun Grid Engine (SGE), is another frequently used cluster management system [Sloa 03]. Maui is a scheduler extension that can be used by many different RMSs [Jack 01], for example PBS, PBS pro, Torque, SLURM, or the Grid Engine. Condor is a resource management system that has been developed to be able to schedule jobs on workstations while not disturbing the activities of people who interactively use the systems [Litz 02].

2.3.3.2 Meta Scheduler

Meta schedulers do not manage the resources of a single cluster but distribute the jobs between several clusters where each cluster can be managed by an own (independent) RMS. Some resource management systems see themselves as meta schedulers. Examples of such schedulers are LSF, Moab/Maui, SGE, or Condor. However in this section, systems are listed that were originally developed to be meta schedulers.

The Grid Workflow Execution Service (GWES) is a workflow engine with Meta scheduling capabilities [Hohe 06, Tayl 07]. The GridWay meta scheduler is part of the Globus Toolkit and provides a single point of access to all resources of a community [Mont 06]. The Meta Scheduling Service (MSS) is designed for co-allocation of computing and networking resources and supports advance reservations [Barz 07]. The Workflow Scheduling Service (WSS) is a workflow management and scheduling component, which was developed for the Climate Community in the Data and Processing Grid (C3Grid) project [Grim 09, Grim 07]. The UNICORE 6 integrated workflow engine is an environment for managing

workflow-oriented scientific and industrial applications in UNICORE infrastructures [Schu 07]. A well accepted standard for service management and orchestration in the business domain is BPEL. There are several commercial and open source workflow orchestrator implementations for BPEL [Hoin 09]. The P-GRADE or WS-PGRADE workflow engine is embedded in a workflow-oriented grid portal[Kacs 05, Kacs 07, Kacs 08]. The Grid Service Broker was developed to handle data-oriented applications [Venu 04]. The Nimrod/G is a meta scheduler based on Globus Toolkit and handles dynamic resources in geographically distributed grids [Buyy 00]. The community scheduler framework (CSF) is a meta scheduling framework from Platform computing [Smit 03]. Freefluo is the workflow enactment engine of Taverna, a workflow language for the semantic grid [Oinn 06]. At last, jBPM is the workflow engine from JBoss, a business process management solution [Koen 04].

2.3.3.3 Virtualization Solutions

Virtualization in the scope of this work means that it is possible to take one or several virtual machines (VMs) that act like an own resource with an own operating system and put them on one host resource. The management software executes these virtual machines separated from each other and the underlying hardware [Barh 03]. The advantage of virtualization in cluster computing is that several virtual machines can be packed on one resource. This allows an improved utilization of the underlying hardware and a higher throughput of jobs [Bolt 10].

Several virtualization solutions are available for cluster management. These are the LXC Linux container system [Laad 10], the OpenVZ Linux container system [Solt 07], the User Mode Linux paravirtualized kernel [Hosk 06], or the VirtualBox hypervisor [Wats 08]. They allow to handle thousands of virtual machines and their images, and users. The commercial cloud solutions typically only support their own hypervisor, while Open Source solutions try to be as general as possible. With the used of libvirt, oVirt or the Common Information Model (CIM) different technologies can be managed through a defined interface [Joha 08, Bolt 10].

2.3.3.4 Cloud Management Systems

The term Cloud computing means the delivery of computing resources as a service. A cloud abstracts from technology, resources and their location. The Cloud computing relies on sharing of resources through virtualization to achieve a high utilization and be able to scale in case of high demand [Mell 09].

In the field of Cloud computing, several software solutions are available. Some of them base on former grid projects. The Xen Based Execution Environment (XBEE) allows a user to get an advanced reservation; the VMs have to be started by hand [Petr 07]. The Xen Grid Engine (XGE) is based on the Sun Grid Engine. It allows the execution of jobs in corresponding VMs [Fall 06, Smit 09]. Another Xen-based solution with similar functionality is Maestro-VC [Kiya 06]. Nimbus [Keah 05] can query the Amazon cloud or a local resource manager (like PBS)

for resource provisioning. Assunção et al. [Assu 09] demonstrated that clouds could be used as an extension to private clusters. The Heizea [Soto 08a, Soto 09b] scheduler is integrated in the OpenNebula [Soto 09a] project and can also be used to study scheduling of virtual machines. JAWS [Grit 06] and UBIS [Walt 08] provide extensions of batch schedulers that allow the use of virtual machines. OpenStack [Open 11] is an open source cloud project by Nasa et al. that provides components for compute and object storage with the support of CloudStack. CloudStack hat two components: the management server and the compute nodes. The supported virtualization technologies are the Xen Hypervisor [Crit 08], KVM [Kivi 07], and VMware vSphere [Lowe 09, VMwa 09]. Eucalyptus *(Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems)* in an open source cloud system that supports the Amazon EC2, S3, and EBS interfaces [Nurm 09].

Conclusion The approaches found in literature are not directly applicable to this work. The scheduling approaches target queuing based systems and work on best effort basis. The aim of these scheduling strategies is to improve the system utilization and to decrease the slowdown of single jobs. Instead, this approach bases on a planning based scheduling scenario with strict deadlines given by SLAs. An acceptance test decides if an additional job can successfully be accepted, and thus, improve the provider's profit by overbooking resources.

2.4 Overbooking

Here, the related work in the field of overbooking is listed. It begins with a brief introduction of fields where overbooking was applied and then looks at related work in the field of overbooking compute clusters or IT services.

2.4.1 Overbooking in General

The examples of hotels [Libe 78] and aeronautical companies [Subr 99, Roth 85] show the overbooking-idea for the provisioning of compute resources this work pursues as well. However, overbooking in the context of compute resources differs from those fields of application since the assumption is made that fewer customers use their reservations than booked. Comparing the usage of a compute resource and a seat in an aircraft is, in this scope, not meaningful. Generally, no fixed intervals for the resource utilization of cluster nodes exist, while a seat in an aircraft will not be occupied after the aircraft has taken off. As a consequence, results and observations from overbooking in the classical fields of application cannot be reused. A job can start on a cluster system any time if enough resources are free.

2.4.2 Overbooking of IT Systems

Overbooking for Web and Internet service platforms is presented by Urgaonkar et al. [Urga 02]. It is assumed that different web applications are running concurrently on a limited set of nodes. The difference between Urgaonkar and this thesis is that the overbooking procedure in this dissertation assigns nodes exclusively. Therefore, it is impossible to share resources between different applications, while it is possible to use execution time length overestimations, which are not applicable to web hosting.

Overbooking for high-performance computing (HPC), Cloud, and Grid computing has been proposed in [Andr 04a, Hove 03]. However, the authors mention the possibility of overbooking but do not propose solutions or strategies. In the grid context, overbooking has been integrated in a three-layered negotiation protocol [Sidd 06]. The approach includes the restriction that overbooking is only used for multiple reservations for workflow sub-jobs. Chen et al. [Chen 04] use time-sharing mechanisms to provide high resource utilization for average system and application loads. At high load, they use priority-based queues to ensure the responsiveness of the applications. Sulisto et al. [Suli 08] try to compensate no-shows of jobs with the use of revenue management and overbooking. However, they do not deal with the fact that jobs can start later and run shorter than estimated.

Nissimov and Feitelson introduced a probabilistic backfilling approach. It applies user runtime estimations and a probabilistic assumption about the real end time of the job to allow to use a gap that is smaller than the estimated execution time [Niss 07]. When estimating the PoS of putting a job in a gap, the probabilistic backfilling and the overbooking scenario are similar. The difference is that Nissimov's acceptance test is applied to an already scheduled job and aims at reducing its slowdown, while the approach of this dissertation is used during the acceptance test at arrival time [Niss 07].

Overbooking in Cloud computing can be applied dynamically. Nodes can host many VMs, and if a host is currently underutilized, a new VM can be assigned to it. In the case a job needs more resources, the assignment can be adapted and VMs can be migrated or suspended. Verboven [Verb 10] analyzed the benefit of overbooking in a virtual environment with different workloads, either being best effort (Low-QoS) or combined with a defined High-QoS. The aim of the scheduling algorithm is to increase the cluster utilization while supporting QoS guarantees. High-QoS jobs get reservations for full CPU slots, while the Low-QoS jobs are used to increase cluster utilization and are suspended when the High-QoS jobs need more resources. The decisions are done online depending on the CPU load and do not require any statistical background.

Another way of load balancing is the live migration of VMs to another host. An estimation of the concrete utilization enables to determine the percentage of the cluster that could be overbooked. This approach requires the ability of job migration and an improved estimation process.

3 Overbooking

The developed overbooking algorithms are described in this chapter. It starts with a description of the acceptance tests and the placing strategies. Then, the used statistical distributions are introduced. Following, the two mechanisms for calculating the Probability of Success (PoS) of overbooking are explained. Further, the complexity of the created algorithms is discussed. The chapter is completed by a description of market mechanisms, to which overbooking could be applied to and a description of the four evaluated strategies.

Contents

3.1 Accepting and Placing Jobs 34
3.1.1 Placing Algorithm for Overbooking 34
3.1.2 Placing Strategy 35
3.1.3 Acceptance Test 35
3.2 Introduction of Statistics 38
3.2.1 Selecting Attributes for PDF Creation 39
3.2.2 Building a Joint PDF 40
3.2.3 Resulting Combined PDF for PoF Calculation 41
3.3 Overbooking Approaches 42
3.3.1 Comprehensive Approach 42
3.3.2 Heuristic Overbooking 48
3.4 Runtime Analysis 51
3.4.1 Comprehensive Approach 51
3.4.2 Heuristic Approach 52
3.4.3 Practical Issues 52
3.5 Market Mechanisms and Overbooking 53
3.6 Evaluated Strategies 55
3.6.1 Conservative Backfilling 55
3.6.2 Comprehensive Overbooking 55
3.6.3 Heuristic Planning 55
3.6.4 Heuristic Overbooking 55

In this work, the comprehensive overbooking was developed first. It covers the most important measurable parameters and allows a very accurate estimation of the PoS. This comprehensive overbooking yield promising results. However, due

to its long runtime a second overbooking strategy was developed. The idea was to create a more practical approach, based on a heuristic.

For the following work, a job j is defined as the tuple:
$[o, e, x, \omega, d, n, R, \lambda_r, \mu_r, l]$. Table 3.1 details the attributes.

Table 3.1: *Job attributes*

Attribute	Description
o	the occurrence time
e	the release time, determines the earliest possible start
x	the user-estimated runtime
ω	the real runtime
d	the deadline of the job
n	the number of requested nodes
R	the set of resources that is supposed to be assigned to the job
λ_r	the machine failure rate of one resource $r \in \mathrm{R}$
μ_r	the corresponding repair rate of the resource $r \in \mathrm{R}$
l	the length of the time slot

3.1 Accepting and Placing Jobs

This section defines a possible scheduling algorithm for backfilling and the acceptance tests. Generally, the scheduler keeps a list of all the gaps in its plan. The gaps have a start-point, when resources become available and an end-point. For each new job arriving in the system, the scheduler calculates the probability of success PoS for the execution of this job in its gaps. Based on this information and a given acceptance test, the scheduler can decide whether it accepts the job. The acceptance test depends on the calculated PoS of the job and the negotiated charge and penalty.

3.1.1 Placing Algorithm for Overbooking

When a new job occurs, the algorithm selects the possible gaps, and computes the corresponding risks and opportunities. Based on this information, the job is either accepted or rejected. The pseudo code for an example overbooking algorithm is shown in Listing 3.1. First, the algorithm selects all possible time slots in between the job's first possible start and its deadline.

For each gap, it is determined whether the length of the time-slot equals or exceeds the user-estimated runtime. In this case, the job can run without overbooking, the time-slot is set to the user-estimated runtime, and the probability of failure PoF for the job execution is calculated based on assumptions about resource outages.

If the time-slot is shorter than the user-estimated runtime, the PoF is calculated with the additional probability of failure caused by overbooking. In case the resulting PoF is lower than a maximum threshold, the job can be accepted for this time-slot; otherwise, the time-slot is deleted.

Different placing strategies can determine where a job should be placed best, and the scheduler has to apply an acceptance test to decide if it takes the job. Possible acceptance tests and placing strategies are discussed in the following.

At the end of any job execution, the resource plan can be updated. This allows following jobs to start earlier.

3.1.2 Placing Strategy

If one or more time-slots are available, the scheduler has to choose one time slot suggested by a given strategy. For the concrete implementation of the acceptance test, several policies can be applied.

- A conservative approach chooses the gap with the highest PoS.
- A best fit approach chooses the gap providing the biggest opportunity.
- A first fit approach chooses the first gap with an acceptable high PoS/opportunity.
- A last fit approach can choose the last gap with an acceptable high PoS/opportunity.

On the one hand, first fit is the fastest algorithm because it terminates after the first possible match. On the other hand, best fit might provide a higher profit. Choosing the conservative approach might prevent some SLA violations and, thus, not damage the provider's reputation. Last fit aims to distribute the utilization and to be able to accept suddenly incoming short jobs with near deadlines. The last gap denotes here a gap which offers the latest possible job-start time. As a consequence, different placing strategies might be chosen depending on the market mechanism and the market situation.

3.1.3 Acceptance Test

When the PoF is calculated, the scheduler has to decide whether accepting the job for the corresponding gap is worth it or not. If the job is accepted, the requested amount of resources is reserved for the selected time-slot and the resource plan is updated. If no acceptable time-slot was found, the job has to be rejected. Depending on the kind of SLAs, two different mechanisms can be used. If an SLA contains charges and penalties for a job, the risk can be computed with the calculated PoS and PoF. If the SLA does not contain money exchange, PoF thresholds can be applied.

Acceptance Based on Risk and Opportunity When the SLA for the job contains a charge and a penalty, it is possible to use this information in

combination with the calculated PoS and PoF to calculate risk and opportunity for the acceptance decision. The following acceptance test uses this information.

Algorithm 3.1 Simple acceptance test based on risk and opportunity.

if $PoS \times Charge > PoF \times Penalty$ **then**
  accept the SLA.
else
  reject the SLA.
end if

This term of Algorithm 3.1 simply says: *Do not accept jobs if the risk (PoF of the job multiplied with the penalty) is higher than the opportunity (PoS of the job multiplied with the charge).* This also means that the expected profit is positive.

Acceptance Based on Probabilities If the underlying SLA is not based on money exchange, for example, in case of a bartering market, the risk acceptance test cannot be applied. In this case, the acceptance test simply uses a given PoF_{max} threshold like it is shown in Algorithm 3.2. If the calculated PoF is below the threshold, the probability of failure is considered low enough and the job can be placed in the gap. As a consequence, this approach is beneficial if the utilization of the system is supposed to increase.

Algorithm 3.2 Simple acceptance test based on PoF threshold.

if $PoF < PoF_{max}$ **then**
  accept the SLA.
else
  reject the SLA.
end if

The following Listing 3.1 summarizes the previously described procedures as pseudo code for an exemplary overbooking algorithm.

```
public class OverbookingAlgorithm{
  static Plan plannedJobs;

  boolean placeJob(Job newJob){
        List possibleGaps = getPossibleGaps(newJob);
        // according to implemented decision strategy (first fit, best fit...)
        Reservation r = selectFirstReservation(possibleGaps);
        if (exists(r)){
          // reservation found
          plannedJobs.add(r)
          return true;
        }else{
          // placing job not possible
          return false;
        }
  }

  List getPossibleGaps(Job job){
   List possibleReservations = new List();
   Reservation possibleReservation;
   List timeSlots = getListOfTimeSlots(job.release, job.deadline, job.res);
   for(TimeSlot ts in timeSlots){
     if(ts.length >= job.estimatedRuntime){
      //no overbooking needed
      long [risk,opportunity] = calulateRessouceRisk(job,ts.length);
      possibleReservation = new Reservation(ts,job, risk,opportunity);
      possibleReservations.add(possibleReservation);
     }else{
      // overbooking needed
      long [risk,opportunity] = calulateOverbookingRisk(job,ts.length);
      if (risk < opportunity){
        possibleReservation = new Reservation(ts,job, risk,opportunity);
        possibleReservations.add(possibleReservation);
        }
     }
   }
   return possibleReservations;
  }

  List getListOfTimeSlots(Time release, Time deadline, Resources res){
   List timeSlots = new List();
   for(Time possibleStart = plannedJobs.getStartTimes()){
     if (possibleStart >= release && possibleStart < deadline){
      if ( plannedJobs.getFreeRessouces(possibleStart) >= res){
        // possible start
        Time start = possibleStart;
        Time end = getTimeWhereLessResAvailable(start,res);
        end = min(end,deadline);
        TimeSlot timeSlot = new TimeSlot(start,end);
        timeSlots.add(timeSlot);
      }
     }
   }
   return timeSlots;
  }
}
```

Listing 3.1: *Pseudo code for an overbooking algorithm.*

3.2 Introduction of Statistics

This section describes how the Probability Density Functions (PDFs) and Cumulative Distribution Functions (CDFs) are created. The overbooking approaches need them for the PoS calculation.

A PDF describes the probability that a job ends after exactly $x\%$ of its estimated runtime. An example for a PDF is given in Figure 3.1, which shows this probability distribution resulting from all jobs submitted to the Arminius cluster in 2007. The x-axis shows the runtime in % of the estimation and the y-axis shows the probability density that the jobs ends at exact this point.

In order to create PDFs, the ratio of real to estimated runtimes of historical job traces has to be extracted. While it is possible to model the PDFs and CDFs as continuous or discrete functions, this dissertation assigns the ratio of the real to estimated runtime of each job to corresponding integrated percent values. As a consequence, the mathematical model for the calculation of the PDFs and CDFs will be based on discrete functions. Therefore, the PDFs have $i = 0, 1, \cdots, 100$ containers with X_i ratios of 0% to 100%. The resulting PDF is defined by the discrete empirical distribution:

$$PDF(x) = \frac{|\{X_i | X_i = x\}|}{n}$$

where n is the number of samples. $PDF(x)$ returns the fraction of samples that have terminated at $x\%$ of the estimated runtime.

Figure 3.1 The PDF derived from all jobs in the examined cluster.

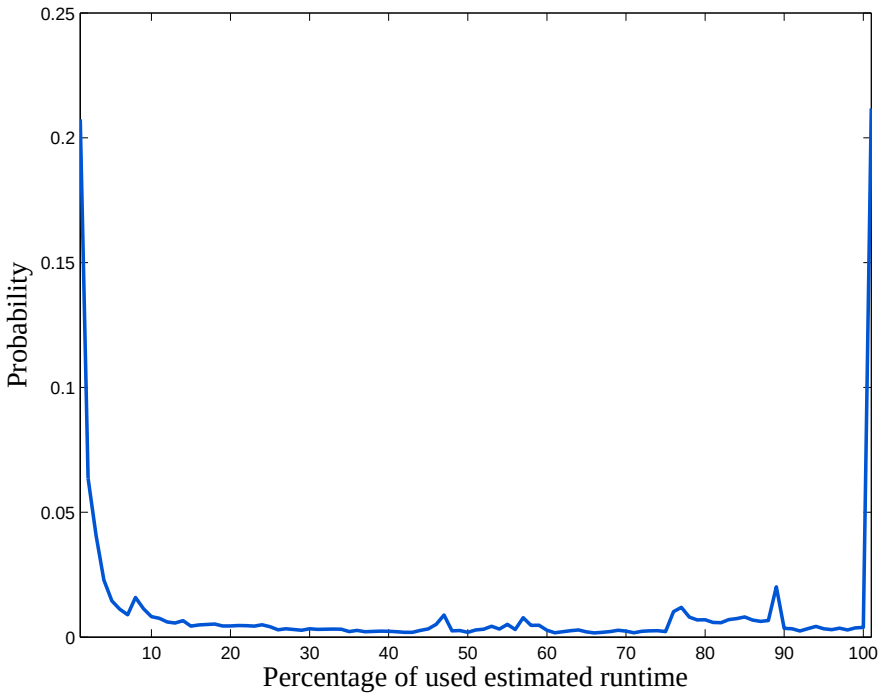
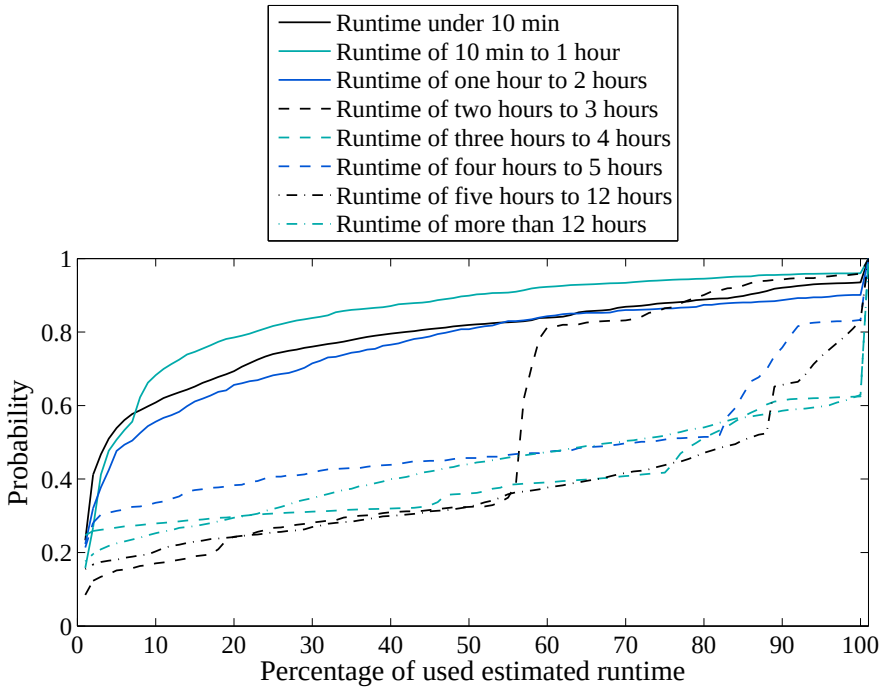


Figure 3.2 The CDF for several time slots.



A CDF is defined as:

$$CDF(x) = \frac{|\{X_i | X_i \leq x\}|}{n}$$

A CDF describes the probability that the real runtime of a job with an assigned PDF will be less than or equal to $x\%$ of its estimated runtime. Thus, the CDF is a cumulative PDF.

$$CDF(x) = \sum_{i=0}^{x} PDF(i)$$

Figure 3.2 shows eight different CDFs. Each presents a group of jobs with an estimated runtime in the same time range.

3.2.1 Selecting Attributes for PDF Creation

The PDFs can be built based on several parameters. One goal of this thesis is to create different PDFs for different job characteristics. When a new job arrives in the system, the most appropriate PDF has to be selected according to the knowledge about the user, the application, or the resources. The more jobs are available in the groups, the more accurate the PDFs can be.

The work uses the following attributes. They are discussed in detail on page 58.

- Estimated runtime
- Requested resources

- Submission user based statistics
- Selected application
- Combinations of the above mentioned

3.2.2 Building a Joint PDF

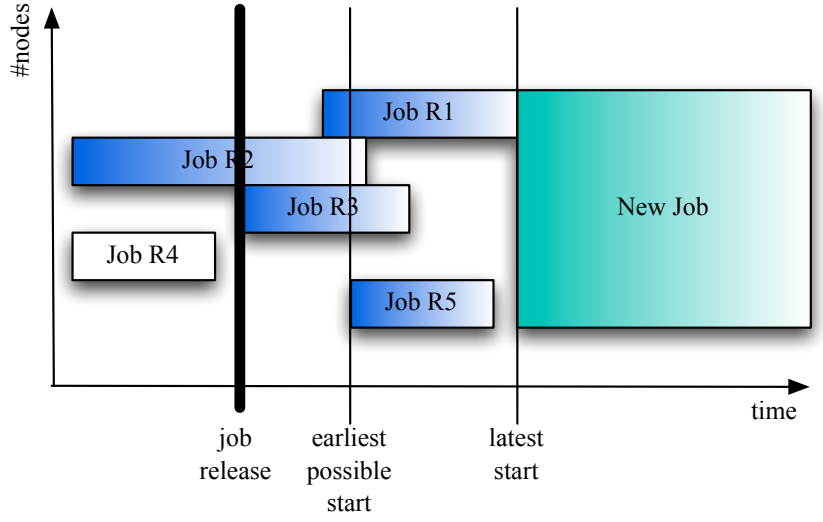
In the overbooking scenario, a job has an assigned start time. However if the corresponding resources are free, it can start anytime after it is eligible. Therefore, all currently running jobs on the assigned resources can influence the start time of a new job. Each of them has its own probability density function that describes its probability to end.

The comprehensive overbooking approach uses the joint PDF of a new job and its direct predecessors to calculate the probability that this job finishes in time. The joint PDF has to include the job's basic function and the combined functions from its predecessors. It contains the information for a set of jobs. Thus, the joint PDF is the basis to calculate the probability that the complete set of jobs ends before the last job's deadline.

The challenge is that several jobs, j_1 to j_n, can end before the start of a new job. The maximum number n of jobs is equal to the amount of required nodes. The minimum number is zero if all resources are free at release time.

The jobs have different estimated runtimes. However, the basic PDF functions store the probabilities in the percent bins. Therefore, when the joint PDF is calculated, the basic PDFs of the jobs are firstly transformed into a PDF format that is based on fixed time step intervals. The PDFs of the fixed time steps have more bins when they are longer and less bins when they are shorter. Afterwards, they can be compared and joint.

Figure 3.3 Exemplary job schedule.



An example is given in Figure 3.3 for a job that requests five resources. We have to calculate the joint PDF of all jobs that are scheduled before and might possibly run before the new job. In the example, these are all jobs but the job R4. The latest start of the job is the latest planned finish of any job planned before (i.e. R1). If possible, a job is allowed to start earlier. The earliest possible start time is either the jobs own release time or the earliest point in time when all previous jobs ended.

The resulting joint PDF at time t is built on the decomposition of the joint CDF.

$$PDF_{\text{joint}}(t) = CDF_{\text{joint}}(t) - CDF_{\text{joint}}(t-1)$$

The time frame the CDF_{joint} represents lies in between a job's release and its latest end. The CDF_{joint} is based on the CDFs of the previously scheduled jobs.

If there is one job before, the joint probability that this job has ended until time t is obviously given by its own CDF. $CDF_{\text{joint}}(t) = CDF(t)$.

If there are two jobs j_1 and j_2 the probability that both jobs have ended until time t are dependent on the probability that j_1 has ended and j_2 has ended. As two different jobs in the schedule are independent, we can multiply both probabilities.

$$CDF_{\text{joint}}(t) = CDF_1(t)CDF_2(t)$$

Following, the resulting $CDF_{\text{joint}}(t)$ for m jobs is:

$$CDF_{\text{joint}}(t) = \prod_{i=1}^{\text{m}} CDF_i(t)$$

The CDFs of the jobs before are already the product of their own CDF and the joint CDFs of their predecessors. Due to this fact, only the last of possibly many jobs on each of the assigned resources has to be considered.

For the complete CDF_{joint} all points in time in between the jobs release and the latest end min(x,l) have to be calculated.

$$CDF_{\text{joint}} = \prod_{i=1}^{\text{\#jobs before}} CDF_i(t)|_{\text{t=0}}^{\text{min(x,l)}}$$

3.2.3 Resulting Combined PDF for PoF Calculation

In case the joint PDF of the previous jobs and the basic PDF of a new job are overlapping, the expected joint execution time distribution consists of the convolution of the job's basic PDF and the calculated joint PDFs of all jobs finishing earlier (see also [Birk 09a]). The resulting combined PDF is the input of the PoF calculation for the comprehensive overbooking.

If the job j has one or more direct predecessors, the convolution of the execution time distribution always has to be computed with the joint distribution of the

previous jobs. The previous jobs already include the distributions of all possibly influential previously planned jobs. For the simulation, the convolutions are based on discrete values.

$$PDF_{\text{combined}} = (PDF_{\text{joint}} \circ PDF_{\text{new job}})$$

The number of steps is given by the used steps per time unit and the length of the job. The distributions are continuous functions and the discrete mapping reduces the accuracy. Nevertheless, the convolutions have to be calculated numerically as no (reasonable) closed formula exists. A discrete convolution is defined as:

$$PDF_{\text{combined}}(n) = \sum_{k=-\infty}^{\infty} PDF_{\text{joint}}(k) PDF_{\text{new job}}(n-k)$$

for each of the n steps in between the job's earliest possible start and latest possible end.

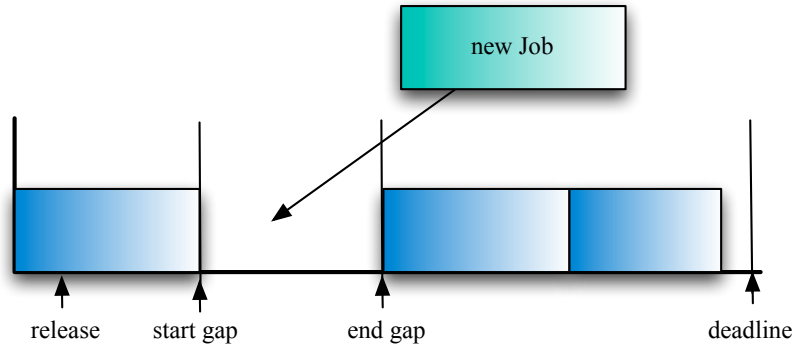
3.3 Overbooking Approaches

This section describes the two approaches for calculating the PoF for overbooking according to the scenario depicted in Figure 3.4. The first presented is the comprehensive approach that covers the most important measurable parameters. Secondly, the developed heuristic is shown. It has a better runtime than the comprehensive approach and is more flexible because it uses more general assumptions.

3.3.1 Comprehensive Approach

The comprehensive overbooking algorithm was designed to support a very accurate PoF calculation. The exemplary job in Figure 3.5 can only be placed overlapping with the jobs on node 1 and 3. It is, however, successful when either

Figure 3.4 The job can only be accepted using overbooking.



the job itself or the jobs on node 1 or 3 run shorter. It is also sufficient, when all jobs together spare the overlapping time.

The comprehensive overbooking procedure maps jobs on resources at SLA negotiation time. This allows using node specific failure predictions. In addition, the placement of jobs to resources at negotiation time allows an inclusion of the failure probability of the specific selected resources into the PoS calculation. Consequently, the probability of successfully completing an overbooked job depends on the probability of resource failures and the probability of finishing in time. To finish in time means the job has an execution time that fits into its time-slot. The result of the calculation is the probability that a job j executes successful in a given time-slot of length l.

PoF(j) and PoS(j) The probability of failure for a job j (PoF(j)) is, with the penalty of the SLA, used to calculate the risk of overbooking. The probability of success for the job j (PoS(j)) is, in cooperation with the fee of the SLA, used to calculate the opportunity of overbooking. The PoF(j) and PoS(j) are directly dependent because $\text{PoF}(j) = 1 - \text{PoS}(j)$. Therefore, the following only describes the calculation of PoS(j).

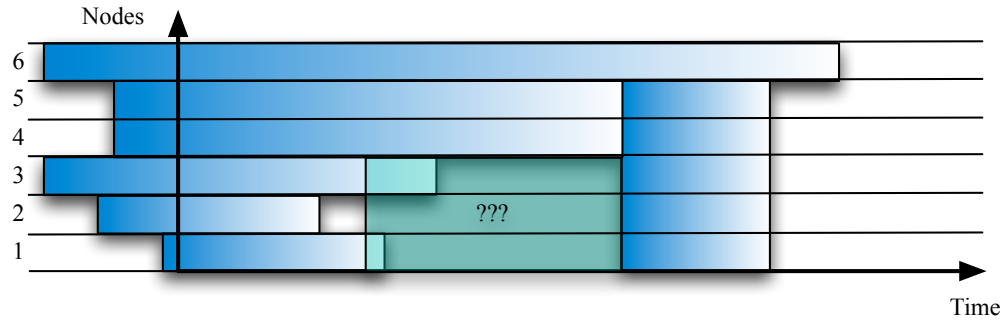
PoS(j) The probability PoS(j) depends on the probability $P_{\text{available}}(\text{R})$ that the requested resources are operational at job start, the probability $P_{\text{executable}}(j)$ that the job is able to end within its given maximum execution time, and the probability $P_{\text{success}}(\min(x, l), R)$ that the resources survive the job's execution. All three factors are independent; therefore, they can be multiplied.

$$\text{PoS}(j) = P_{\text{available}}(\text{R}) \cdot P_{\text{executable}}(j) \cdot P_{\text{success}}(\min(x, l), \text{R}).$$

In the following, the factors are defined in detail.

$P_{\text{available}}$ The probability $P_{\text{available}}$ that a resource is available at job-start depends on its mean time to failure MTTF and its mean time to repair MTTR. The availability of one resource/node is defined as the ratio of its MTTF to the aggregate of its MTTF and MTTR.

Figure 3.5 The new job overlaps with the planned jobs on resources 1 and 3.



$$P_{\text{available}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

Here, MTTF is given by the failure rate λ, MTTF $= \frac{1}{\lambda}$, and MTTR is given by the repair rate μ, MTTR $= \frac{1}{\mu}$, of a resource.

$$P_{\text{available}} = \frac{\frac{1}{\lambda}}{\frac{1}{\lambda} + \frac{1}{\mu}} = \frac{1}{1 + \frac{\lambda}{\mu}}$$

When a job is planned to run on a set of resources R, the probability $P_{\text{available}}(\mathrm{R})$ depends on the probability that all required nodes are available at start time. The distinction of resources allows us to create accurate estimations of the probability of job crashes caused by node outages.

$$P_{\text{available}}(\mathrm{R}) = \prod_{i=1}^{n} \frac{1}{1 + \frac{\lambda_i}{\mu_i}}$$

Here, the index i denotes the failure and repair rates of the i_{th} resource from the set of n resources R. This model assumes that the node failures are independent. This is a simplification compared to previous work [Iosu 07, Schr 06]. It has been shown that node failures are bursty and correlated. However, a successful job execution is not possible, if one of the planned resources fails. Therefore, the amount of other node failures is not included in the calculation.

$P_{\text{executable}}$ The probability of success for a job execution $P_{\text{executable}}(j)$ is defined based on the job's cumulative distribution function. The cumulative distribution function describes the probability that a job finishes until its maximum given runtime $(\min(x, l))$. If the job is planned with the user's estimated execution time x, the probability $P_{\text{executable}}(j)$ is 1. If the runtime is limited to $l < x$, the probability $P_{\text{executable}}(j) < 1$.

If the job has no predecessor in the plan, it is scheduled as soon as possible and $P_{\text{executable}}(j)$ is given by its own execution time distribution. If the job has one or more direct predecessors, the convolution of the execution time distribution always has to be computed with the joint distribution(s) of the previous job(s). The reason for using the joint distribution of the predecessor jobs is that due to overbooking, job j has no defined start time. The job starts at the end of its last ending direct predecessor. Which job of its predecessors ends last is not necessarily known. It can be any job on the planned resources, which has a possible end time after the new job's release e. The predecessor jobs already include the distributions of all possibly influential, previously planned jobs due to their own $P_{\text{executable}}$ calculation.

$P_{\text{executable}}(j)$ is given by the convolution of its own probability density function PDF_j and the $\text{PDF}_{\text{joint}}$ ($\text{PDF}_{\text{combined}} = \text{PDF}_{\text{joint}} \circ \text{PDF}_j$).

The defined $\text{PDF}_{\text{combined}}$ function has to be transformed into its cumulative function $\text{CDF}_{\text{combined}}$. The resulting function is defined as:

$$P_{\text{executable}}(j) = CDF_{\text{combined}}(\min(x, l))$$

The process of building PDF_{joint} as well as the modeling of the convolution process that results in $P_{\text{executable}}(j)$ was described in Section 3.2.

P_{success} $P_{\text{success}}(\min(x, l), \mathrm{R})$ describes the probability that the job's resources R survive the execution time. The maximum possible execution time is either the user-estimated runtime x or, in the overbooking case, the length of the time slot l. According to the underlying failure information, several approaches can model the survival probability.

Weibull Failure Model
Several papers show that crashes in cluster systems are correlated and bursty [Schr 06, Saho 04] and that the failure rates of large clusters follow a Weibull distribution [Iosu 07, Nurm 05]. According to Feitelson, the CDF for a Weibull distribution is defined as

$$F(x) = 1 - e^{-(\frac{x}{\beta})^{\alpha}},$$

where x is the investigated time frame, α is a shape parameter, and β is a scale parameter.

Consequently, for calculating P_{success} according to a Weibull failure model, the definition of $P_{\text{success}}(\min(x, l))$ as $1 - e^{-(\frac{\min(x,l)}{\beta})^{\alpha}}$ would describe the survival rate of a specific node. Here, β describes the spreading of the distribution, this means how fast the tail will decay. α describes the failure rate over time of a specific resource. A value of $\alpha < 1$ indicates that the failure rate decreases over time due to high infant mortality. $\alpha = 1$ means the failure rate is constant, and a value of $\alpha > 1$ indicates that the failure rate increases due to an aging process [Feit 11]. α and β have to be calculated based on monitoring information of the past failure behavior for each of the resources $r \in R$.

Following, the success rate for one resource r can be modeled as:

$$P_{\text{success}}\left(\min(x, l)\right) = 1 - e^{-(\frac{\min(x,l)}{\beta})^{\alpha}}$$

Node failures are typically dependent on the failures of other nodes. Since almost all jobs fail when one of its resources fails, it does not matter if the node failure causes or is caused by other node failures. Causality has no influence on an already failed job. Therefore, the failures are modeled to be independent. Following, the resulting Weibull failure model for a set of resources R is:

$$P_{\text{success}}(\min(x, l), \mathrm{R}) = \prod_{i=1}^{n} P_{\text{success}}(\min(x, l), \mathrm{r_i}) = \prod_{i=1}^{n} 1 - e^{-(\frac{\min(x,l)}{\beta_i})^{\alpha_i}}$$

Here, $r_i \in \mathrm{R}$ denotes the i_{th} resource from the set of n resources R. Each resource r_i has an own failure shape α_i and failure spreading β_i.

Hyper-Exponential Failure Model
Other related work assumes that hyper exponential distributions are the best instrument for failure modeling [Nurm 05, Mutk 87, Lee 93]. According to Feitelson, a hyper-exponential distribution is a combination of several exponential distributions.

This means that each failure can be caused from a set of reasons. Each reason has an own exponential distribution with failure rate λ and a probability p. Following, the $P_{\text{success}}(\min(x, l))$ for one resource and one failure reason would be an exponential distribution.

$$P_{\text{success}}(\min(x, l)) = 1 - e^{-\lambda \min(x,l)}$$

In the case of two failure reasons, the first reason is described by the first exponential distribution with probability p_1 and the failure rate λ_1. The other exponential distribution has the probability p_2 and failure rate λ_2. The probability is $p_2 = 1 - p_1$ and λ_1 should be different from λ_2, else the two failure reasons can be unified to one failure reason. For one resource with two failure reasons, the resulting success probability is:

$$P_{\text{success}}(\min(x, l)) = 1 - (p_1 e^{-\lambda_1 \min(x,l)} + p_2 e^{-\lambda_2 \min(x,l)})$$

If more than two failure reasons Λ exist the probability is the sum of the single exponential distributions. Following, for one resource r and k failure reasons, the definition of P_{success} is:

$$P_{\text{success}}(\min(x, l)) = 1 - \sum_{j=1}^{k} p_j e^{-\lambda_j \min(x,l)}$$

Here, $\lambda_j \in \Lambda$ denotes the failure rate of the j_{th} failure reason. The j_{th} reason has the occurrence probability p_j. It holds $\sum_{j=1}^{k} p_j = 1$.

If a job has several resources $r \in \mathrm{R}$ and k failure reasons, the definition of P_{success} is the product of the single failures of each resource r_i.

$$P_{\text{success}}(\min(x, l), \mathrm{R}) = \prod_{i=1}^{n} P_{\text{success}}(\min(x, l), \mathrm{r_i}) = \prod_{i=1}^{n} 1 - \sum_{j=1}^{k} p_{ij} e^{-\lambda_{ij} \min(x,l)}$$

Here, $r_i \in \mathrm{R}$ denotes the i_{th} resource from the set of n resources R. $\lambda_{ij} \in \Lambda$ denotes the j_{th} failure rate (with the occurrence probability p_{ij}) of the k failure groups of resource r_i.

Poisson Process
The Weibull or Hyper-Exponential distributions describe an aging process of the

resources that takes months, maybe years, while the typical jobs only last between a couple of minutes and some days. In addition, the failure rate has to be adapted over the day and week/weekend because it has been shown that the failure rate also depends on the load of the system [Schr 06, Saho 04]. Therefore, Weibull or Hyper-Exponential distributions were not the best choice for this dissertation.

Instead, focus for the job execution is on a Poisson process with constant failure rate λ. Feitelson gives a good overview about statistical distributions in *Workload Modeling for Computer Systems Performance Evaluation* [Feit 11].

The constant failure probability λ allows a simple model of the probability that the job's resources survive the execution time. For one resource the probability is:

$$P_{\text{success}}(\min(x, l)) = 1 - e^{-\min(x,l)\lambda}$$

If the job is assigned to more than one resource, a statistical estimation of λ for each cluster resource r is needed and the Poisson process describes the probability that all of the job's resources survive the execution time. When the job occupies a set of several resources R the probability is modeled as:

$$P_{\text{success}}(\min(x, l), \mathrm{R}) = \prod_{i=1}^{n} P_{\text{success}}(\min(x, l), r_i) = \prod_{i=1}^{n} 1 - e^{-\min(x,l)\lambda_i}$$

The failure rates $\lambda_1, \ldots, \lambda_n$ describe the probability that the job survives the execution time with n resources $r_1, \ldots, r_n \in \mathrm{R}$.

When the monitoring system is able to create fine-grained failure rates λ, for instance, for each hour of a day, the given λ could be replaced by a sophisticated function.

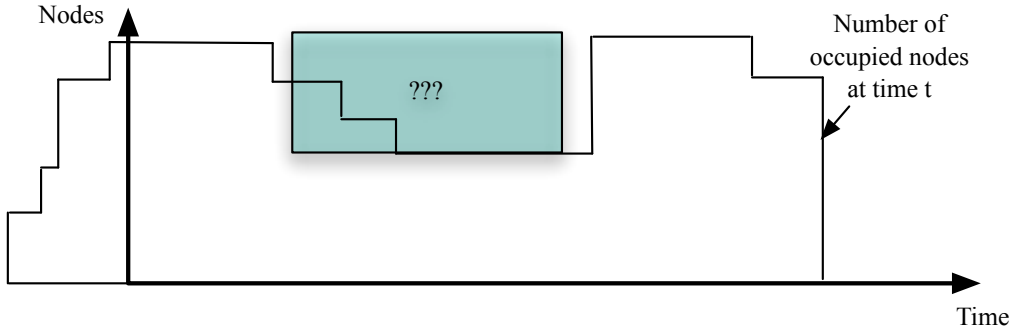
For the simulation, the Poisson process was implemented because the available job traces were limited to a few months. Therefore, no resource aging was assumed and a constant failure rate chosen.

The ideas for the comprehensive overbooking with, first, the focus on single resources and, secondly, the extension to parallel systems, were shown in [Birk 09a, Birk 10].

Drawbacks of the Comprehensive Approach However, the comprehensive approach has some disadvantages for a practical application.

- Cluster systems with many nodes offer a large amount of possible combinations when and where a job can run. Checking all combinations needs building PDF convolutions for every possible gap. This leads to a bad runtime of the planning process. The number of jobs for each simulation run was limited to 1000 in order to retrieve results in an acceptable time.
- The mapping of jobs to specific compute resources at submit-time is inflexible. Many other nodes may become free before the job can start on the planned resources. This earlier start time would often enable the job to be completed in time.

Figure 3.6 Heuristic Overbooking:
The free and occupied nodes are counted for all planed jobs.



- The schedule is unnecessarily fragmented because unused time slots on the resources remain free.

3.3.2 Heuristic Overbooking

Due to the huge amount of convolutions, the comprehensive approach needs a long runtime. Therefore, additionally a heuristic was supplied.

The heuristic overbooking approach no longer decides where the job should run at SLA negotiation time. Instead, it counts the overall resources and uses their number to calculate the probability of the job to be successful. As a consequence, the resources are not assigned to the job at planning time but at its start. The job is accepted only with its planned start and end time. Figure 3.6 highlights the number of free nodes as a line that forms the shape of the already planned jobs.

This approach is more flexible because a job can run on every node. The drawback is that it is not known which job(s) will be the predecessor(s) and that only a general assumption about the resource stability can be made. In addition, the application of the heuristic is only possible if all nodes can execute all jobs with the same performance.

3.3.2.1 Planning and Overbooking: A Heuristic

The heuristic planning strategy is able to overcome the drawbacks of the comprehensive overbooking. It is designed to be more flexible than the comprehensive overbooking and allows a faster acceptance test. The decision which resources should be used is no longer made at submit-time but at the job start. Therefore, specific resources cannot be taken into account anymore. Instead, the calculation uses general assumptions about resource stability. Calculating joined PDFs is impossible because waiting jobs are not mapped to resources.

The heuristic algorithm combines the runtime and the resources of each job into an overall schedule of the machine. When a new job arrives, this plan is used to determine at which point in time a job can be started. The schedule considers the

accumulated runtimes of the already planned jobs, the user-estimated runtime of the new job, and the number of resources.

The heuristic planning approach has the following abilities:

- The system is much more flexible because the planning does not assign nodes; it just counts free resources.
- The procedure is conservative because the algorithm only estimates the probability of the job finishing within the planned start and end time. An SLA violation of one job has no influence on other jobs.
- The schedule is calculated much faster because it does not rely on convolutions.
- The schedule is less fragmented because the scheduler replans the whole schedule each time a job finishes.
- Replanning allows a steady update of the complete schedule, and even overbooked jobs can get their full runtime if previous jobs finish earlier.

In summary, the planning is easier to calculate and more flexible.

3.3.2.2 Calculation of the PoS

This section illustrates the calculation of the PoS for the heuristic. The PoS describes the probability that the job has an execution time that fits into a timeslot between its specified start and end time.

For the following calculations, we keep the attributes $[o, e, x, \omega, d, n, \lambda, \mu]$. The set of resources R is missing because it is not known which resources the job will occupy later. Accordingly, node specific failure rates λ_r and repair rates μ_r cannot be considered anymore either. Instead, a general assumption about the node stability λ, μ is used.

PoS(j) The probability PoS(j) is, like in Section 3.3.1, defined by the availability $P_{\text{available}}(n)$, the probability $P_{\text{executable}}(j)$ that the job is executable, and $P_{\text{success}}(\min(x, l), n)$ the probability that the n job's resources survive the execution.

$$\text{PoS}(j) = P_{\text{available}}(n) \cdot P_{\text{executable}}(j) \cdot P_{\text{success}}(\min(x, l), n)$$

$P_{\textbf{available}}$ The heuristic does not know where a job will run until it is started. Therefore, only a generic estimation of the node stability $P_{\text{available}}(n)$ is possible. From the comprehensive approach, we know that the probability that the resources are available at job-start is the product of probabilities of the single nodes. Where in the comprehensive case, each resources had an own failure probability, in the heuristic there is a general one: λ and μ.

Following, the probability that the resources are operational at the start time is

$$P_{\text{available}}(n) = \prod_{i=1}^{n} \frac{1}{1+\frac{\lambda}{\mu}} = \left(\frac{1}{1+\frac{\lambda}{\mu}}\right)^{n}$$

where $i = 1, \ldots, n$ is the requested number of resources. Due to the fact that all nodes have the same failure probability, it is possible to power up the results to the amount of resources instead computing all equal single results and building the product.

$P_{\text{executable}}$ Another simplification of the heuristic is based on the calculation of $P_{\text{executable}}(j)$. Convolutions are not done anymore. The heuristic only uses the CDF_j of the job itself and the planned job length $\min(x, l)$. The maximal allowed execution time $\min(x, l)$ is either the user-estimated runtime x or, in the overbooking case, the length of the time-slot l.

$$P_{\text{executable}}(j) = CDF_j(\min(x, l))$$

P_{success} $P_{\text{success}}(\min(x, l), n)$ describes the probability that the job's resources survive the execution time. Unlike to the comprehensive approach, the used resources are not known at negotiation time. Therefore, only general assumptions about the stability of a cluster's resources can be made. This leads to a simplification of the formulas used for calculating the success probability. In the next paragraphs, you can see that the indexes of the used products are not used anymore. Each product computes the same value. Therefore, we can just power up one result to the power of the requested resources.

Weibull Failure Model
When only a general stability assumption of the cluster system is possible the variables simplify as follows. Here, n is the number of requested nodes.

$$P_{\text{success}}(\min(x, l), n) = \prod_{i=1}^{n} \left(1 - e^{-(\frac{\min(x,l)}{\beta})^{\alpha}}\right) = \left(1 - e^{-(\frac{\min(x,l)}{\beta})^{\alpha}}\right)^{n}$$

Hyper-Exponential Failure Model
The hyper exponential distributions use generalized groups of exponential distributions for all resources $r_1, \ldots, r_n$.

$$P_{\text{success}}(\min(x, l), n) = \prod_{j=1}^{n} \left(1 - \sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i \min(x,l)}\right) = \left(1 - \sum_{i=1}^{k} p_i e^{-\lambda_i \min(x,l)}\right)^{n}$$

Poisson Process
Each resource has now the same failure rate λ for the job execution. Therefore, a Poisson process has for n requested resources the following probability.

$$P_{\text{success}}(\min(x,l),n) = \prod_{i=1}^{n} \left(1 - e^{-\min(x,l)\lambda}\right) = \left(1 - e^{-\min(x,l)\lambda}\right)^n$$

This process describes the probability that the job's resources survive the execution time. The heuristic overbooking approach was presented in [Birk 11a].

3.4 Runtime Analysis

Here, a runtime analysis of the approaches is given in the Big-O notation. The runtime of the algorithms is dominated by two factors: The selected placing routine and the runtime of the PoS calculation.

The acceptance tests are done in $O(1)$ because they compare 2 numbers, the PoF and the threshold, or the risk and the opportunity.

A convolution has a calculation-runtime that is dependent on the underlying implementation. The most common used convolution algorithm is the fast Fourier transformation (FFT). The FFT is applied by the simulation and has a runtime of $O(t\log(t))$ [Corm 01]. Here, t is the number of steps in the discrete distributions. However, the estimated runtime of a job and therefore the length of a PDF is typically limited. The resulting question is; can we assume a constant runtime $O(1)$ for the calculation (due to the job-runtime limitation) or has the number of steps t an impact on the calculation-runtime $O(t\log(t))$?

3.4.1 Comprehensive Approach

The placing of the comprehensive backfilling and overbooking is comparable to the bottom-left strip packing approach. The proven runtime for an efficient implementation of this approach is $O(n^2)$ [Chaz 06].

The runtime of the PoS calculation depends on the calculation of the three terms $P_{\text{available}}$, $P_{\text{executable}}$, and P_{success}. $P_{\text{available}}$ and P_{success} are calculated by a product of the results of each planned resource and therewith depend on the number of resources k. The runtime of the calculation of $P_{\text{executable}}$ depends on the convolution of the joint PDF of the jobs that are planned directly before. In the worst case, this may be $(n-1)$ jobs. However in this thesis, the jobs are exclusively assigned to the resources and, therefore, the number of parallel running jobs cannot be bigger than the number of resources k. Thus, the runtime of the PoS calculation is combined by the runtime of $P_{\text{available}}$, $P_{\text{executable}}$, and P_{success}. The complexity of building the joint PDF is in $O(k)$.

If we assume the number of resources to be similar compared to the number of jobs that are typically planned at a time and the length of the estimated job-runtime to have influence on convolutions, then, the runtime of the comprehensive

approach would be influenced by the resources $O(k)$, the job placing $O(n^2)$, and the convolutions $O(t \log(t))$. In this case, the theoretical result of the combined runtime of the comprehensive approach would be $O(t \log(t) n^2 k)$.

If we assume the runtime of the jobs t and the number of resources k to be limited, then, the runtime would be constant $O(1)$. In this case, the runtime would be $O(n^2)$. Hence, a measurement of the simulation runtime in practice would be beneficial to compare the approaches.

3.4.2 Heuristic Approach

The heuristic does not place a job on resources directly. Instead, it counts the occupied resources sorted by time. At job-start, the system decides where the job should run. This can be done in constant time because the free resources are known. As a result, omitting resources at planning time reduces the complexity from 2 dimensions (resources $\times$ time) to one dimension (time). In theory, this corresponds to a search and update problem that should be solvable in $O(n \log(n))$.

For the implementation in this work, the scheduler holds a list of free resources. If a new job is put into the schedule, a fitting gap in the list has to be found. This can be done in linear runtime $O(n)$ and has to be done for each of the n jobs. This means the placing strategy also has $O(n^2)$ runtime.

The runtime of the PoS calculation is also dependent on the runtime needed to calculate $P_{\text{available}}$, $P_{\text{executable}}$, and P_{success}. However, the calculation does not depend on the number of resources but can be done in constant time. In addition, there are no convolutions because only the jobs' own PDF is used. Therefore, no FFT algorithm is needed and all PoS calculations have a constant runtime $O(1)$.

If we assume the number of jobs to be dominant compared to the number of resources, the result of the combined runtime analysis of the heuristic algorithm would be also $O(n^2)$. In this case, both approaches would have the same complexity.

However, in practice the number of jobs that are planned by the provider will be limited because the jobs will have deadlines that are near their release. Hardly a customer will accept and pay for jobs that are planned in the far future. Therefore, in practice the estimated job-runtime and the size of the cluster will have an impact on the real runtime of the algorithms. However, these factors only affect the comprehensive approach, thus, the heuristic will very likely be faster. As a consequence, a closer look of the practical runtime issues is beneficial.

3.4.3 Practical Issues

In practice, the jobs are not known in advance and occur online. This means a scheduler does not need to deal with all jobs. At a point in time in which the provider offers his services, not all jobs will have arrived and some were already completed and gone. Thus, only the actual planned jobs p are counting.

In addition, the number of resources in modern clusters can be very large[1]. Therefore, the number of resources can have an impact on the average runtime.

Further, not the complexity class but the real runtime of the algorithms is important. The customer of a resource provider wants a direct feedback. A response time of one second is acceptable. Five or more seconds may not be acceptable. For websites, it is shown that most users are willing to wait for only about two seconds for simple information retrieval tasks [Nah 04].

If the complexity of placing exactly one job is analyzed, the complexity is reduced to the number of planned jobs p. This means that both overbooking algorithms have, according to the runtime analysis before, a complexity class of $O(p)$. In this case, the runtime that is used for the probability calculation is important. The complexity for calculating the probability is $O(t \log(t)k)$ for the comprehensive and $O(1)$ for the heuristic approach. If estimating the runtime complexity under this circumstances, the comprehensive approach is with $O(t \log(t)pk)$ worse than the heuristic with $O(p)$. However in practice, the job-length that influences t will be limited, the resources are limited, and the planned jobs will very likely not exceed a certain limit because users will try other providers if they would have to wait to long for the job execution.

Following, to get a clue whether the assumptions made will be reproducible in practice, the simulation at the end of this thesis should not only show the impact of both overbooking algorithms but in addition measure their runtime.

3.5 Market Mechanisms and Overbooking

A resource provider normally applies overbooking transparently. For example, the customer does not know that his flight company has overbooked his plane until money is offered to take the next one. Accordingly, overbooking of cluster resources could be applied transparently to the SLA negotiation.

However, different market mechanisms for computational markets exist. Therefore, different overbooking strategies might be beneficial. The related work in Section 2.1.2 shows a brief overview of market mechanisms. These are:

- Auctions
- Bidding and Tendering
- Bartering Models
- Commodity Models

Now, a discussion about the applicability of overbooking to the market mechanisms follows.

Auctions [Nisa 98, Pada 03, Wald 92] are the most frequently used market mechanism. Here, the provider of a service might introduce overbooking to offer more

[1]JUGENE of FZ Jülich has nearly 300,000 cores `http://www.fz-juelich.de/portal/DE/Forschung/Informationstechnologie/Supercomputer/JUGENE.html`

resources than available. In the auction based overbooking, the price is not fixed but depends on supply and demand. Therefore with overbooking, a provider might reduce the price that is offered by the customers due to a virtually higher supply. As a consequence, the application of overbooking in auction-based markets has to be applied with caution. However if the user demand is significantly higher or the own resources are already sold, overbooking should provide extra income.

Bidding or *Tendering* [Ston 94] are reverse auctions. The customer defines the price of a service. Here, the provider can calculate the PoS for overbooking for a job under the actual utilization of its services. If the opportunity (PoS and the fee) that is offered for the bid is higher than the risk (PoF and the penalty), the bid can be accepted. In that case, the price is customer defined before. In the long run, the provider can thus reduce prices on the market because the customer might have offered a higher bid when no one wanted to accept his former one.

In *Bartering Models* [Buyy 05], resources are exchanged without the involvement of money. This means, the impact of an SLA violation can only be measured in lost reputation. Reputation and its impact on the next trades is difficult to measure. Consequently, overbooking should only be used for jobs with a minimal PoF. In addition, different traded goods like storage space of CPU cycles need different measures for a PoF estimation. If storage space is exchanged and the space is overbooked, statistics about storage usage are necessary.

Commodity Markets [Amir 98] are markets where the prices are given by a formula. Overbooking might be included in the functions of commodity markets by adding terms that allow the application of overbooking. Another way to include overbooking might be to implement it transparently into the commodity functions. In case of overdemand, a provider might offer more resources than he has available. Like all other market mechanisms, overbooking might reduce the prices because the ratio of supply and demand also influences the formulas. Overbooking creates more virtual resources and hence the overall prices might decrease.

To summarize, overbooking can be included transparently into market mechanisms.

This thesis assumes that the overbooking process is transparently applied to the user. The SLA negotiation is not part of this work. The resource providers can decide which SLA technology they want to take and which market mechanism they want to apply by their own demands.

The negotiation can be done through direct communication between users and customers or over a broker. Based on the offered price for a service, the requested resources, utilization, and resource stability the resource provider can decide if he applies overbooking or not. However independent of the market overlay, overbooking might decrease the market prices.

3.6 Evaluated Strategies

For the purpose of analyzing the benefits of overbooking, this thesis implemented four scheduling strategies. The heuristic and the comprehensive overbooking approach and their corresponding strategies without overbooking were applied.

3.6.1 Conservative Backfilling

Conservative backfilling plans jobs on their resources with the full estimated runtime. The resources that a job will use are determined at negotiation time. It is known which jobs are running where and when. This allows using stability estimations for the specific resources. Consequently, the PoF calculation process is very accurate. The conservative backfilling does not release nodes at job end but at the end of the user-estimated execution time. However when a job can fit in a gap in the plan, it is backfilled and planned at the earliest possible start-point. Only gaps of sufficient size are used. The backfilling is conservative because no already planned job is delayed.

3.6.2 Comprehensive Overbooking

The comprehensive overbooking works on the same basis as the conservative backfilling. In case the comprehensive overbooking approach tries to overbook a job, the PoF is calculated according to the comprehensive overbooking strategy. A promising gap is selected due to a given placing strategy and acceptance test. When a job ends, its resources are freed even when the user-estimated runtime was not fully consumed. It is then checked if any of the planned jobs can start earlier on the freed resources.

3.6.3 Heuristic Planning

Heuristic planning counts the available and already planned resources and checks whether or not a job can be executed with its requested resources and full estimated runtime. Resources are not assigned to jobs at planning time. Instead, the heuristic counts the free resources in a given time plan and, based on this plan, can determine when enough resources will be free. This allows a prediction of a job's start and end time but does not contain information on which resources a job will run. The used resources are selected at job start. When a job ends earlier as user-estimated, the plan is rescheduled. This allows jobs to start earlier, prevents fragmentation of the schedule, and, thus, allows a higher utilization.

3.6.4 Heuristic Overbooking

The heuristic overbooking works similar to the heuristic planning but can assign less runtime to a job. In case the heuristic overbooking approach tries to overbook

a schedule, the PoF is calculated according to the heuristic overbooking strategy. A promising gap is selected due to a given placing strategy and acceptance test.

4 Determination of PDF Classes

This chapter discusses the statistical analysis that provides the basis for estimating the probability of a successful overbooking. The chapter starts with a brief discussion of the possible job attributes that can be used for the creation of statistics. Thereafter, an exemplary analysis of user estimations of traces from the Arminius Cluster in Paderborn is shown. The chapter is completed by a user survey that was designed to reveal how the users of Arminius estimate their job runtimes.

Contents

4.1 Correlating Attributes 57
4.2 Analysis 58
4.2.1 Runtime 60
4.2.2 Resources 63
4.2.3 Applications 65
4.2.4 Users 67
4.2.5 Conclusion 67
4.3 User Survey 69

4.1 Correlating Attributes

Preventing SLA violations is the most important task when applying overbooking. Accurate predictions about the runtime-estimations are key factors to be profitable. At a first glance, an analysis of runtime-estimations and real execution-times shows that the job durations are on average overestimated by a factor of two to three [Stre 03]. Other related work shows that the distribution of the actual maximum runtimes seems to be uniform. Depending on the trace, 15% to nearly 30% of jobs are *underestimated* [Mual 01]. The RMS kills them after the SLA-guaranteed runtime.

To obtain appropriate predictions about the overestimated runtime, job observations on clusters and an analysis of the user-committed SLAs are necessary. If enough monitoring data is available, the following question arises: *How can statistical information about actual job runtimes be used to efficiently overbook machines?*

Related work shows two approaches to estimate runtime. The first approach focuses on user estimations, and the other one tries to automatically generate the estimations, based on previously measured application runtimes.

In this dissertation, the users estimate their job runtimes and the overbooking algorithms try to evaluate the estimation quality based on statistical information. To find convincing statistics, the jobs were classified by the following attributes:

1. *Runtime-Estimation Analysis* The basic analysis could be about the relation of estimated to real runtimes. The question is, how accurate is the user estimation ability, and further, is the estimation quality dependent on the job length, i.e. are longer estimated jobs estimated better [Down 99, Feit 95]?
2. *Requested Resources Analysis* Another anchor point could be the analysis of correlations between the resource consumption of jobs and the estimation accuracy.
3. *User Runtime Analysis* Better results could be achieved if the runtime-estimation analysis is done on a user basis. Users often submit the same applications with similar input and similar runtime. However, for a user specific analysis, many previous jobs of the users have to be collected. This approach is not possible for new users.
4. *Application Runtime Analysis* In addition, an application-oriented statistical analysis of monitoring data could be applied. This analysis can identify correlations of overestimations and specific applications. Performed studies [Gibb 97, Down 97, Smit 98] show that automatically determined runtime-estimations based on historical information can be better than the user's estimations. Enough data has to be available to allow such an analysis.
5. *Combined Runtime Analysis* In addition to the runtime-estimation, resources, user, and application analysis, another way could combine the approaches in order to identify specific correlations.

For all methods, the selected jobs have to be analyzed, and their behavior has to be described by the corresponding probability density functions.

In case the statistical interpretation shows that a user made accurate estimations, the scheduler should not use his jobs for overbooking. If the statistic shows that another user underestimates the runtime, the scheduler might even assign more runtime to avoid job kills. If the statistic shows that some users tend to overestimation, their jobs are promising for overbooking.

4.2 Analysis

This section presents the results of the interpretation of job traces of the Arminius-cluster in Paderborn. The analysis is the basis for the creation of different PDF classes. The Arminius cluster had 200 nodes, each with dual core Xeon CPUs and 4 GB RAM. OpenCCS managed the cluster system. Every user had to provide a runtime-estimation when submitting a job. OpenCCS does not log single entries for a job in the log-files, but different entries for different events. Each standard event, SUBMIT, CHTIME, ALLOC, EXEC, or FREE has its own entry.

- A SUBMIT entry is logged when a user inserts a job into CCS.

- A CHTIME entry is logged when a user changes the estimated duration of a job.
- An ALLOC entry is logged when the resources are allocated.
- An EXEC entry is logged when a job finishes. This entry contains the real runtime a job needed.
- A FREE entry is logged when the resources are released.

For parsing the traces, the distributed log entries needed to be combined because different information is needed for building the statistical analysis.

- Submit-time (job occurrence) (from the SUBMIT log entry)
- Estimated execution-time (from ALLOC or, if existing, CHTIME)
- The user (from SUBMIT)
- Requested resources (from SUBMIT)
- Start-time (from ALLOC)
- The executable (from EXEC)
- Real execution-time (from EXEC)

The users can adapt the reserved runtime-estimation during their job's run when they recognize that the job would otherwise be killed. Each adaption is logged by a CHTIME log entry. When one or more CHTIME commands adapt the runtime-estimation, the last runtime-estimation is taken as input for the statistics. If no CHTIME event occurred the runtime-estimation is collected from the ALLOC entry. Taking the latest runtime-estimation entry statistically means that the users created better estimations as they really did. The fact that they adapted the runtime during a job's execution means that they miss-estimated the runtime before.

Two kinds of reservations in OpenCCS cannot be used for analyzing the user estimation quality.

- OpenCCS allows empty advance reservations. This means a reservation does not necessarily have a job (an EXEC entry). The entry is missing when the user interactively works on the nodes and starts jobs himself via the command line. In such a case, there is no real execution-time logged.
- The other subset of reservations that cannot be used are reservations with more than one EXEC command. A user can reserve nodes and start jobs on the reservation with a specific CCS command. In this case, many real runtimes occur for a reservation for only one estimated runtime.

In both cases, the system has to provide the full estimated runtime for the reservation. Overbooking is not even possible when the advance reservation remains unused because the user might access the reservation at its end. In the following, the advance reservations were ignored for the Arminius analysis.

Clustering Estimations For the analysis of the user runtime-estimations, the Arminius traces from 2007 were used. The investigated job classes were:

- *Runtime classes:* The question is if grouping jobs into different classes of estimated runtime helps to get more precise PDFs. For example, longer running jobs might be better estimated.
- *Resource classes:* Is grouping jobs into classes according to the requested number of cluster nodes beneficial? The underlying assumption here is that jobs with more resources might be better estimated.
- *Application classes:* There are several kinds of applications: some have the same runtime, for others the runtime depends directly on the input, and there are still others where it is uncertain how long a job will take. Grouping jobs into classes according to the used applications might be an advantage for the PoF estimation accuracy.
- *User classes:* There are users that try to estimate the runtime for every job, and there are users that always estimate a too long runtime to be sure that the job has enough time to finish. Therefore, an analysis based on different users themselves might also provide good results.

Table 4.1: *Runtime of Arminius' jobs in 2007 in seconds*

Jobs	33,318
Total Runtime	571,256,953
Total Estimated Runtime	1,013,932,300
Average Runtime	17,145
Average Estimated Runtime	30,431

This chapter finishes with the summary of a survey that asked the most frequent users about their estimations and whether they would profit from statistical interpretations of their estimation quality.

4.2.1 Runtime

This section presents the time analysis on the basis of different types of runtime-estimation classes. The first task was the definition of the time-frames to get an adequate classification. On the one hand, a fine granular classification with many classes might lead to a better estimation process. On the other hand, a class needs a measurable amount of jobs for the statistical interpretation, otherwise the resulting PDFs have a weak expressiveness. A short overview led to the following *time-frames* for the estimated runtime: $x < 10$ minutes, 10 minutes to 1 hour, one hour to 2 hours, two hours to 3 hours, three hours to 5 hours, five hours to 12 hours, and more than 12 hours. Table 4.2 summarizes the jobs belonging to the classes.

Figure 4.1 shows the probability density functions that were created based on the jobs of the Arminius system and filtered according to the eight runtime classes. Figure 4.2 shows the corresponding cumulative density functions.

An ordering of the CDFs according to the estimation quality is difficult because many of the CDFs are overlapping and consist of a number of very early ending

Table 4.2: *Runtime-estimation classes and the belonging jobs*

Timeframe	Number of jobs
under 10 minutes	5,685
10 minutes to 1 hour	5,806
one hour to 2 hours	1,070
two hours to 3 hours	827
three hours to 4 hours	11,945
four hours to 5 hours	814
five hours to 12 hours	3,770
more than 12 hours	3,401

jobs and also a number of very late ending jobs with less jobs in between. The resulting curve of such a class of jobs is increasing very fast at the beginning and also increases very fast at the end but has a very low pitch in the middle.

However, a class with a huge number of jobs with a real-runtime near the runtime-estimation, can be counted as good. This means that the curve increases slowly at the beginning and more rapidly at the end because well estimated jobs are finishing near the end of the estimated runtime. A fast increasing curve (like the three curves for the classes of jobs under 10 minutes, jobs of 10 minutes to 1 hour, and jobs of one hour to 2 hours estimated runtime) symbolizes classes where are

Figure 4.1 The resulting PDFs from the runtime-estimation analysis.

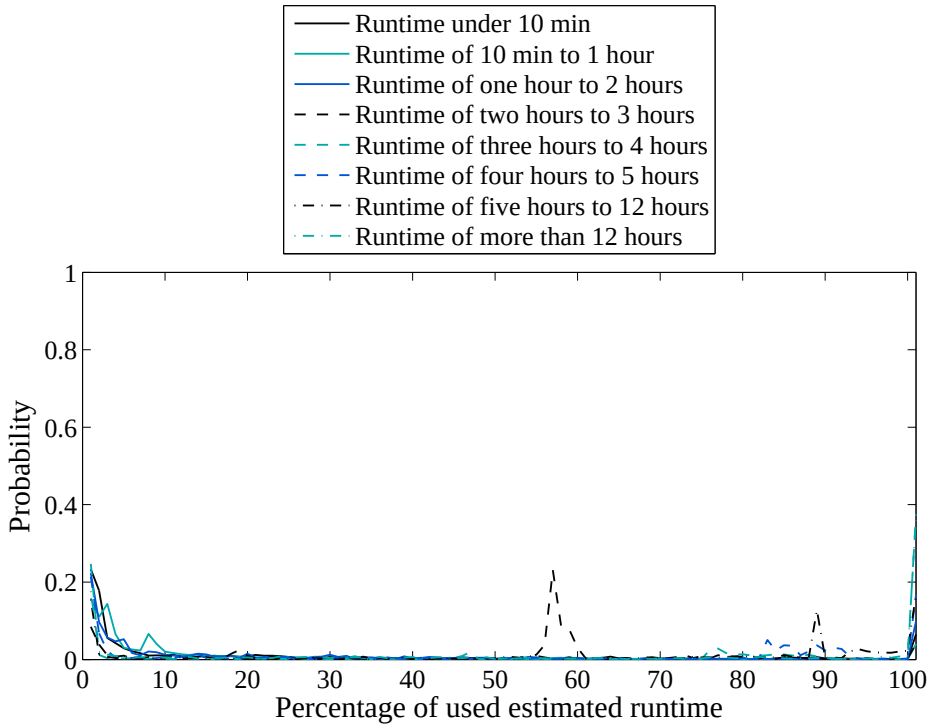
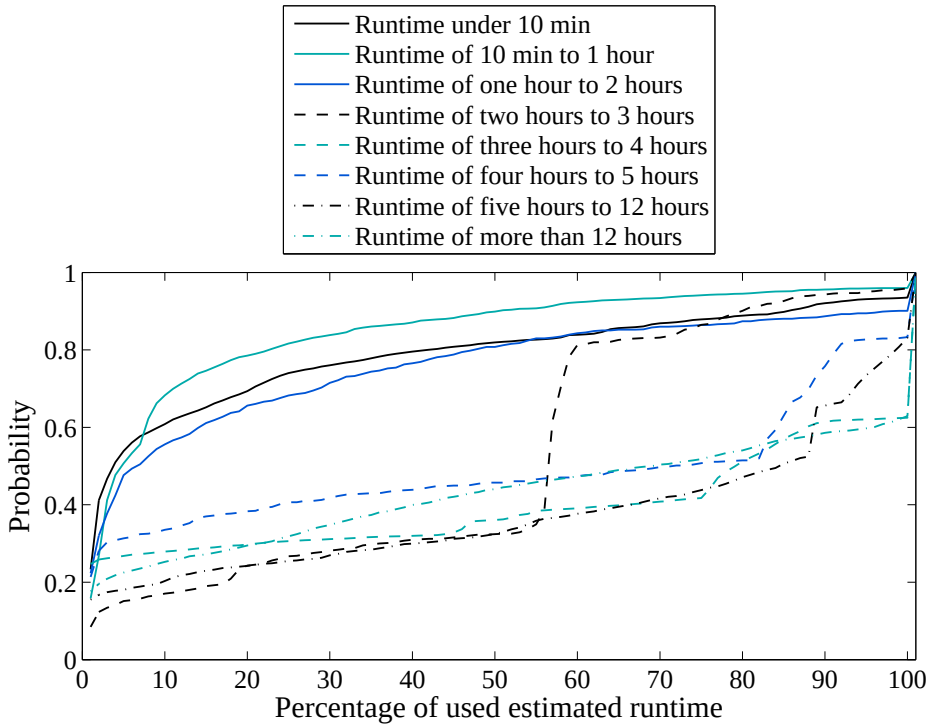


Figure 4.2 The resulting PDFs from the runtime-estimation analysis.



a huge amount of jobs is ending very early. Over 50% of these jobs use less than 10% of their estimated runtime.

The analysis of the Arminius data shows that the assumption that longer estimated jobs are estimated better does hold. Of the jobs with a runtime-estimation of more than three hours, more than the half need more than 80% of their estimated runtime.

The class of jobs from two to 3 hours shows an interesting result. Here, a measurable amount of jobs ended between 50 to 60 percent of the runtime-estimation.

Organized according to the estimation quality, the best estimated jobs are: over 12 hours, three to four hours, five to 12 hours, four to 5 hours, one to 2 hours, the class for 10 minute jobs, two to 3 hours, and 10 minutes to one hour.

4.2.2 Resources

This section presents the resource analysis on the basis of different resource classes. The classes are build on the jobs number of requested nodes. The classes are summarized in Table 4.3.

Table 4.3: *Resource classes and the belonging jobs*

Requested number of nodes	Number of jobs
1 Node	7,582
2 Nodes	11,600
3 to 4 Nodes	5,675
5 to 8 Nodes	4,418
9 to 16 Nodes	2,864
17 to 32 Nodes	982
33 to 64 Nodes	131
more than 64 Nodes	66

Figure 4.3 shows the PDFs that were created based on the Arminius trace and sorted according to eight resource classes. The goal was to find meaningful groups of jobs with a similar number of jobs in it. This was difficult because there is a dominating number of jobs with a few resources. Therefore, the classes where

Figure 4.3 The resulting PDFs from the statically analysis about requested nodes.

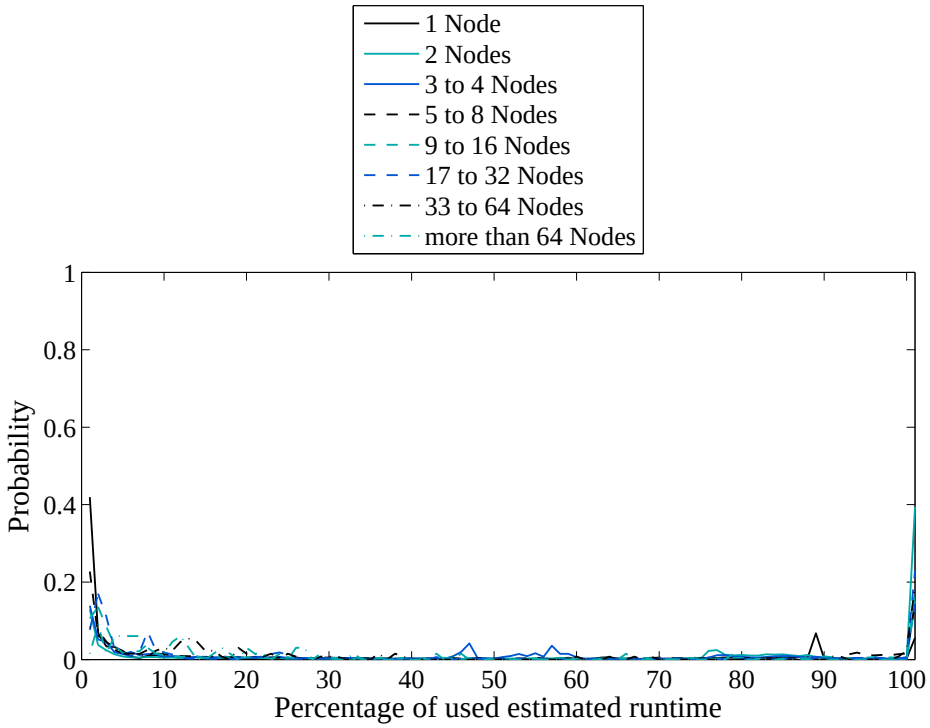
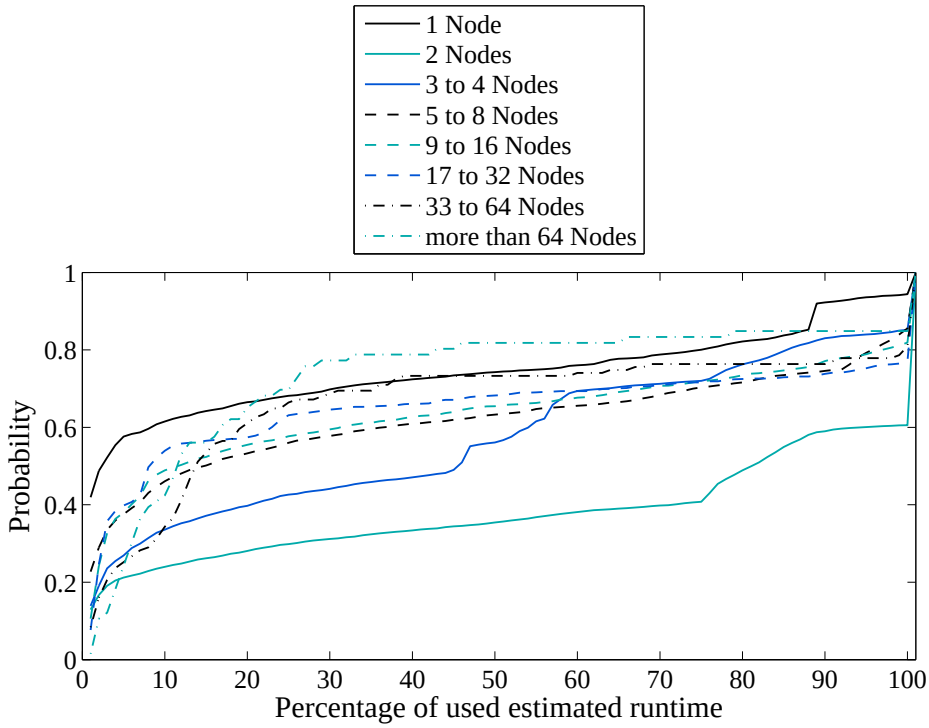


Figure 4.4 The resulting CDFs from the statically analysis about requested nodes.



selected according to requested resources and the resources grouped in borders by the powers of 2.

The CDFs of Figure 4.4 show that the class of jobs with two requested nodes has the best estimation quality. The class with three to four nodes contains 50% of jobs that use more than 50% of the estimated runtime. The class for jobs with one node has, on the one hand, the biggest amount of jobs that use zero percent of the estimated runtime but also a measurable amount of jobs that are well estimated (30% of the jobs use over 50% of the estimated runtime). Such jobs are inappropriate for overbooking because of their extreme behavior. The reason for the huge amount of one-node jobs ending with zero runtime is that many test jobs were executed. They were used to detect if OpenCCS was operational or if the availability of the overlaying Grid software was monitored by automated test jobs. The jobs were submitted for one node, the typical 10-minutes default duration, and only contained a *date* or *hostname* command. For overbooking, such test jobs might be detected and removed because they are submitted without negotiation or the monitoring daemon is negotiating.

The curves of the other classes are similar. The analysis shows that the jobs with a small amount of nodes are in tendency better estimated than the jobs with many nodes.

In practice, similar job-classes could be combined. However, in this dissertation the classes were kept. For the simulations, also logs of other clusters systems

were applied and these clusters could have jobs that show more divergent behavior dependent on the requested resources.

4.2.3 Applications

This section describes the application analysis on the basis of classes according to different applications. Arminius was a compute cluster frequently used by scientists. Consequently, it was regularly occupied by applications that were under development and improved and completed over time. Improved simulations allow us to get results with an increasing level of detail. This process, however, may require more runtime according to the increasingly complex calculations. On the contrary, the runtime might decrease due to more efficient algorithms or data structures.

As a consequence, grouping the jobs into classes of executables might lead to many possibly imprecise classes. As a result, classes for bigger groups of jobs were necessary. A very interesting idea is the use of classes for different *workers*. OpenCCS allows the creation of workers for special kinds of jobs. These workers define the execution environments for applications. Thus, the workers ease the usability of the cluster system. The workers were used to group jobs into different classes and these classes were used for the statistical interpretation.

Figure 4.5 The resulting PDFs from the statistical analysis of requested workers.

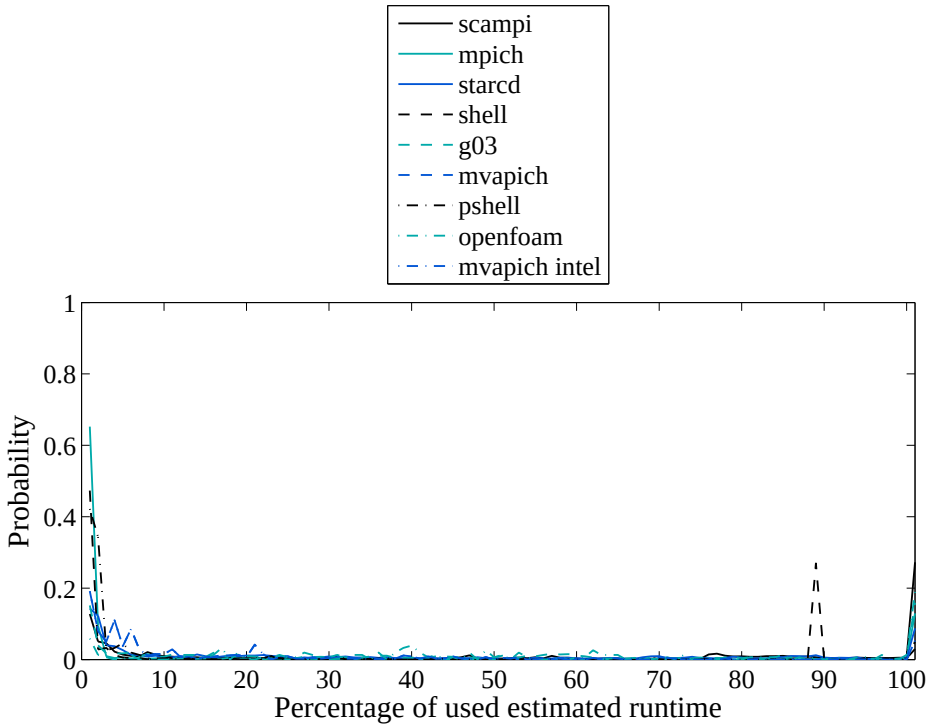
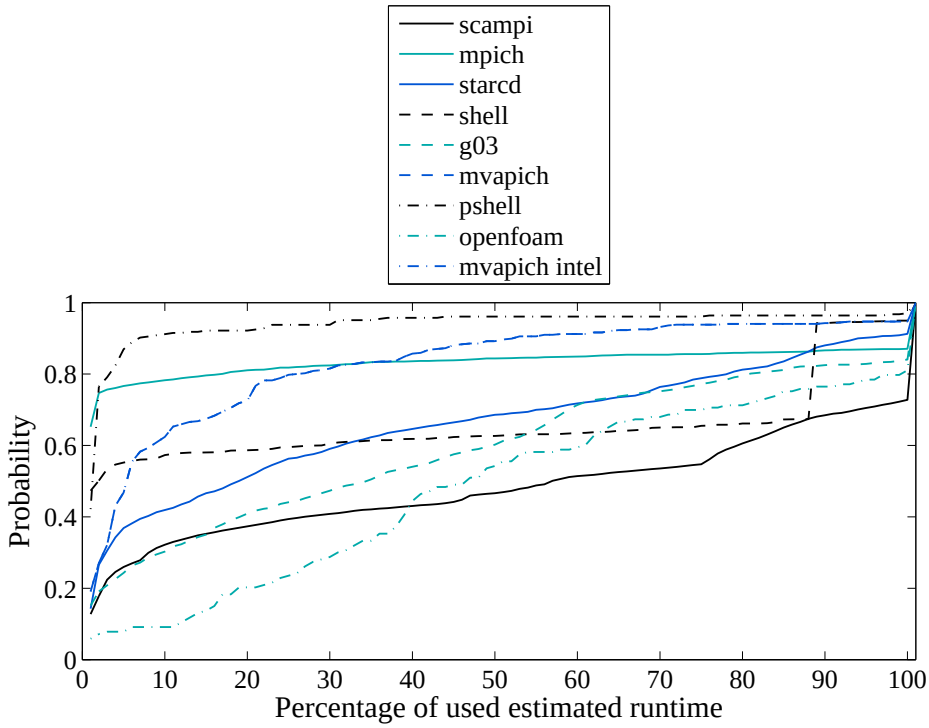


Figure 4.6 The resulting CDFs from the statically analysis about requested workers.



There are workers for applications like g03, abaqus, openfoam, or starcd, and there are workers for execution environments like scampi, mpich, mvapich, shell, or pshell. Table 4.4 summarizes the workers and their submitted jobs. Figure

Table 4.4: *Workers and the belonging jobs*

Worker	Number of jobs belonging to the worker
scampi	22,635
mpich	3,447
starcd	3,017
shell	1,767
g03	1,325
mvapich	455
pshell	306
openfoam	153
mvapich	455

4.5 shows the PDFs that were created based on the workers of Arminius. Figure 4.6 shows the corresponding CDFs. Scampi is the best estimated execution environment. Openfoam only has a small amount of really badly estimated jobs, while mpich has over 60% of jobs that use less than 1% of their runtime-estimation. Overall, the worst estimated workers were mpich and pshell.

4.2.4 Users

This section provides the results of the user analysis on the basis of different users and their estimation abilities. Arminius was utilized by 112 different scientists in 2007. Some of them submitted a huge amount of jobs and some only very few. Statistics are more conclusive when they are based on many jobs. Therefore, the top 10 users were selected for analysis. They submitted 22923 jobs that represent 69% of all jobs in 2007. Table 4.5 summarizes the analysis of the top 10 users.

Table 4.5: *Top 10 users and their jobs*

Worker	Number of jobs submitted from the user
user 1	11,480
user 2	2,821
user 3	1,830
user 4	1,524
user 5	1,247
user 6	1,205
user 7	1,130
user 8	1,115
user 9	750
user 10	740

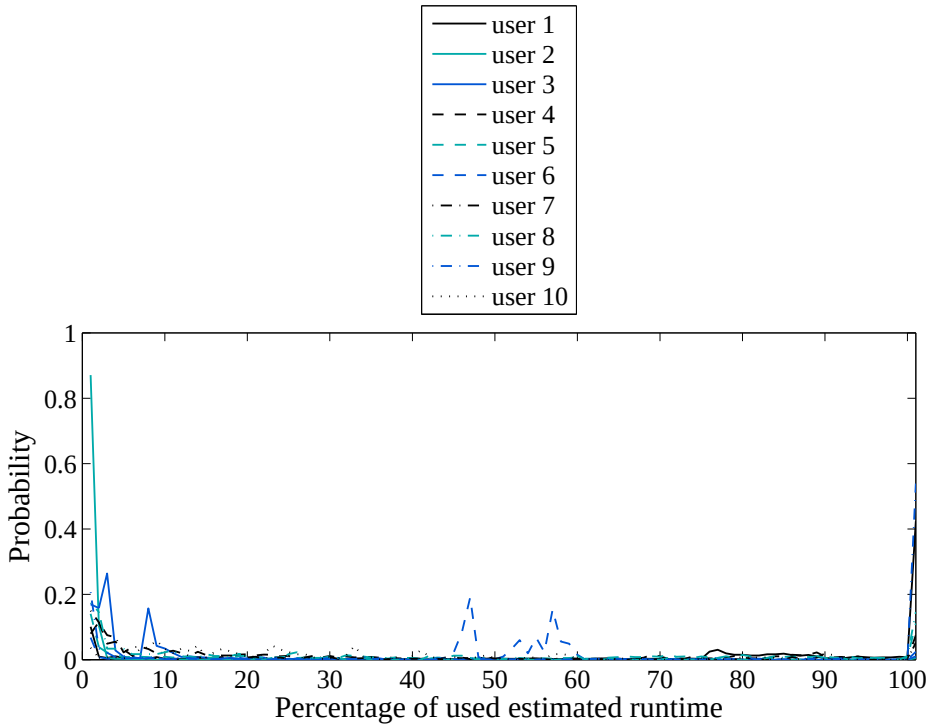
Figure 4.7 shows the PDFs and Figure 4.8 the corresponding CDFs. User 1 submitted over 11,000 jobs, almost one-third of all jobs. He estimated very well compared to other users. The following user survey revealed that his program supported checkpointing. As a consequence, he did not care about a job-kill in case of overestimation; the job could be restarted from its last checkpoint. Users like him have to be treated carefully when overbooking is applied or such users and their jobs might be excluded from overbooking. The number of jobs he submitted is so high that the shape of the curves can also be seen in the other classes his jobs fall into.

In addition, 80% of the jobs of user 6 use between 45% and 60% percent of their runtime-estimation. Such users are more reliable for overbooking. User 5 has a nearly uniform looking PDF and according an almost linear increasing CDF. The users 4, 7, 8, and 10 have similar distributions. The users 2 and 3 have the worst estimations. User 2 greatly overestimated the runtime, over 95% of his jobs ended directly at their start-time.

4.2.5 Conclusion

The very divergent statistics for the different categories indicate that the chosen categorization will be useful. They reveal the greatly varying behaviors of jobs for the different classes. Knowing the behavior of highly specialized classes of jobs allows us to create far more precise probability density functions and, consequently, more precise estimations of the probability that a job will finish in

Figure 4.7 The resulting PDFs from the statistical analysis of the users.



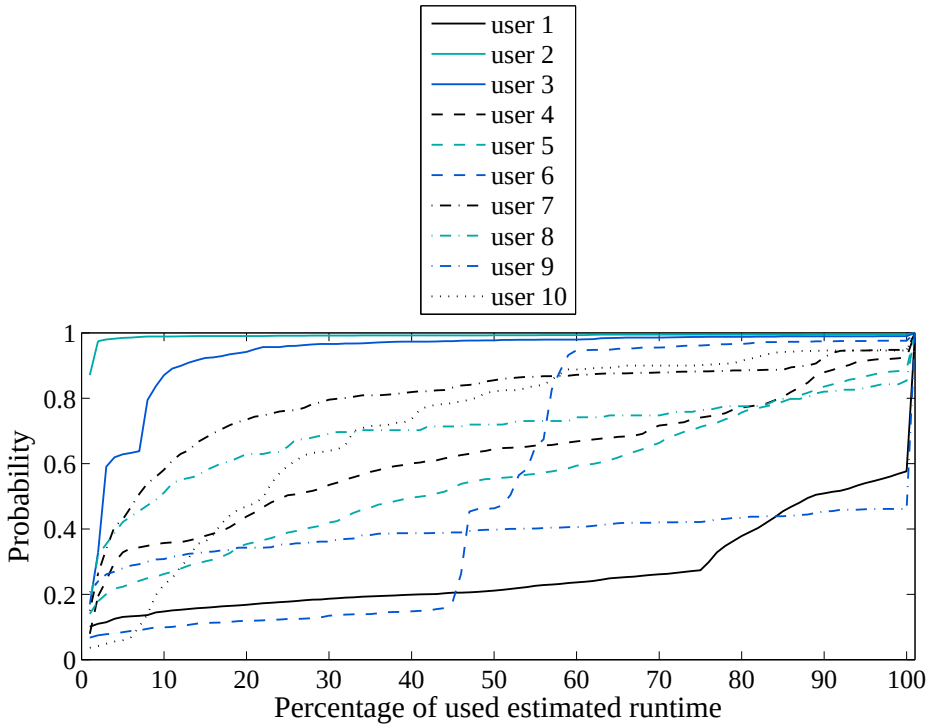
a given gap. Therefore, the chosen classes seem to be promising in supporting overbooking with accurate PoS estimations.

However, a number of different classes contain jobs with a similar behavior. The distinction of the jobs in these classes is not very useful. The classes could be combined or a new partitioning could create more expressive results. Nevertheless, in this dissertation the classes were kept. For the simulations, also logs of other clusters systems were applied and these clusters could have jobs that show more divergent behavior.

Concluding, the user CDFs show the highest spreading of the curves. As a consequence, this analysis might be best to apprehend and map divergent estimation qualities. Somehow, this seems to be obvious because an analysis cannot be more specific as focussed on one user. Such analyses are the best fitting statistical inputs for the overbooking algorithms. Therefore, overbooking might be most promising with the PoS calculation based on this classification. However for users with little jobs or new users, using such a specific analysis is not possible. For them, a more general assumption has to be used.

The simulation of the overbooking strategies will reveal which classification might be most useful.

Figure 4.8 The resulting CDFs from the statistical analysis of the users.



4.3 User Survey

As an extension to the analysis of the user runtime-estimation abilities, a survey asked the users of 2007 about how they estimate their job's runtime.

The users were asked if and how they estimate runtimes, what kind of applications they use (HTC, massively parallel, long running, batch), if they develop the used applications themselves, if they use one or many applications, and if their applications are dependent on each other. Further, the survey asked them about the reasons for their runtime-estimation and if they use checkpointing. At last, the study asked if the users would profit from a statistical interpretation of their runtime-estimations. The top 40 users were queried and 16 surveys came back. Table 4.6 shows the results.

From the top ten users in 2007, five filled out the survey. Of the five surveys, two were from the best estimating users. Both use applications with an embedded checkpointing mechanism, where a restart of a job is possible. This indicates that applications will have better estimated jobs. Thus, overbooking should be used with caution for those users.

The result indicates that checkpointing allows jobs to more likely use their full runtime. As a consequence, applications with checkpointing should only be overbooked very carefully.

Table 4.6: *Summary of the user survey.*

Answers	16
use third party software	8
use self developments	10
HTC/batch jobs	7
massive parallel jobs	10
long running jobs	6
Runtime of test jobs	
short	3
variable	3
long	0
very long	0
Runtime of jobs	
always the same	1
often the same	4
no idea	1
rarely the same	7
never the same	1
If more than one application is used, how frequently is it changed	
rarely	4
dependent on project	5
parallel execution	0
rarely dependent on results	1
frequently	0
Is checkpointing possible	
no	4
no and I miss it	0
yes, but I do not use it	1
yes, rarely I use it	2
yes, I use it frequently	9
How do you estimate the runtime of your jobs	
I always book the same time	3
trial and error	0
overestimation	5
I use checkpointing and re-start from checkpoint if it was underestimated	3
I know the runtime from the last runs	2
other estimations	3
Profit from statistical analysis of the job runtimes	
yes	5
no	11

Other interesting observations are that more than half of the users developed their application themselves (10 out of 16) More than half of the users' jobs were massively parallel (10 out of 16). When checkpointing is available, users tended to use it (9 out of 12).

The analysis of the user estimation abilities presented here is based on the scientific users and traces of an academic compute provider. OpenCCS operates at best effort, this means that a user has no guarantees that the job will be executed. OpenCCS plans the jobs and offers the user a defined start date, but in case the cluster system crashes, the user's job will be replanned once the cluster system is available again.

The users were allowed to use the cluster system based on their scientific projects. They did not have to pay for an execution. This might have encouraged them to overestimate the resources because they only wanted to make sure that their jobs were not killed due to a deadline. An indication for this assumption is that most users thought that they would not profit from an analysis of their runtime-estimations. The users were happy with the state of the art process. However, if the users had to pay money, they might try to buy tighter runtime bounds. As a consequence in a commercial scenario, the jobs might be estimated better.

Depending on the application, the survey indicates that for some users a better estimation process might be possible. Four out of 16 answers said that the runtime of jobs can be estimated well and one answer said that his application always needs the same runtime.

However, 9 out of 16 answered that the runtime is not assessable. Even in a commercial scenario, this group of users will further tend to overestimate.

Conclusion What do we learn from the survey? First of all, 16 answers are not sufficient for a survey to get a broad knowledge about user estimation abilities. However, some conclusions can be extracted.

When the used applications support checkpointing, the probability that jobs will take their full runtime increases; such jobs should be overbooked more carefully.

In scientific environments, the users do not care much about their estimation quality and, consequently, are not interested in a statistical analysis of their runtime-estimations. This behavior might change in a commercial environment where the users have to pay for their estimated runtime. Accordingly, they might estimate better. As a consequence, the results of the simulations of overbooking clusters might not directly be applicable to the commercial case.

However, the log traces from non-commercial clusters can still be conclusive and give an idea. Even in the commercial case, a broad number of customers (in the survey 56%, or 9 out of 16 cases) would overestimate their jobs because they have no or little idea about the requested runtime. As a consequence, they would still have to overestimate their job-runtimes to be sure to get the results.

5 Simulation Environment

The simulation environment was designed to evaluate the benefits of the overbooking approaches. This chapter discusses the used parameters first. It is described which parameters can be taken directly from the input sources, which are adjusted, and which are created. Secondly, the implementation of the simulation environment is shown, including the used data structures and the event system. At last, the section presents the implemented placing routines first fit and best fit.

Contents

5.1 Simulation Parameters 74

5.1.1 Simulation Input 74

5.1.2 Parallel Workload Archive 76

5.1.3 Arminius 77

5.1.4 Acceptance Tests 77

5.2 Implementation 78

5.2.1 Used Data Structures 78

5.2.2 Events 78

5.2.3 Placing Routines 81

The proposed overbooking algorithms need to be evaluated to know their value for cluster providers. A theoretical proof of the quality and abilities of mathematical models serves this purpose. However when looking into real job data, no suitable underlying statistical function was found that could have been the basis for a mathematical proof. Consequently, a simulation should evaluate the abilities of the overbooking algorithms on the basis of real job traces. The simulations used two sources of real world job traces.

The first one is the Parallel Workload Archive (PWA) [Feit 10]. Here, many job traces of various, mostly scientific compute providers are listed. These traces are frequently used to create evaluations of new scheduling ideas and algorithms. The traces should allow comparing the simulations of this dissertation to simulations in related work. The traces of the workload archive are an important source of job traces in the field of scheduling simulations. However, the job information of the traces is condensed. Some information, like used applications or users, are reduced to numbers. As a consequence, a suitable grouping of these kinds of information is not possible.

Therefore, job-traces of the PC^2's own cluster Arminius were applied additionally. This allows asking the cluster operators about job behaviors or just querying

Table 5.1: *Job Creation Model*

	Variable	Description
x	requested job length	from job traces
ω	real job length	from job traces
o	occurrence time	adapted from job traces
e	earliest allowed start	equal to the occurrence time
d	deadline	$o + 2x$
n	requested resources	from job traces
λ	failure rate	from failure archive
μ	repair rate	from failure archive

the users about the ways they estimate the job runtime. The user-survey was described in Section 4.3.

5.1 Simulation Parameters

Basis for the simulation are the job information that were retrieved from the job-traces. Most attributes and parameters were directly extracted and used. However, some attributes had to be adjusted because the simulations need a high load of jobs. In addition, deadlines had to be created; otherwise overbooking could be done without any risk.

5.1.1 Simulation Input

The following paragraphs describe the used information for the simulation input. The description starts with the directly selected information, describes where the simulation adjusted job information, and finishes with the information that had to be guessed because they were not contained in the job traces. Table 5.1 shows the simulation's job data.

Selected Job Information

For each job, the directly selected information were the user-estimated runtime x, the real runtime ω, and the number of resources n. Not listed in the table but extracted for the simulation were the user and the used executable.

The failure and repair rates of the resources were not contained in the job traces. The failure archive from the Los Alamos National Laboratory [Los 11] was used as an alternative source. To be comparable, the same failure traces were used for every simulation run (LANL traces 18 and 19). Both traces had 1024 cluster nodes. Therefore, it was possible to simulate up to 2048 different resources. For clusters with more resources, the simulation started for the 2049^{th} node at the first LANL node again. According to the LANL traces, the failure rate was defined as

$\lambda = 1.2904 \cdot 10^{-4}$ and the repair rate as $\mu = 0.4333$. These values are used as the statistical assessments of the resource stability.

Adjusted Job Information

In practice, job submissions are bursty. As a consequence, the occurrence times o from the traces should be used to preserve the natural behavior. Unfortunately, it is shown by the traces in the Parallel Workload Archive [Feit 10] that clusters are not fully utilized during the complete runtime and the utilization varies, depending on the trace, between 10.7% and 83.5%. To create an overload, the occurrence rate of the jobs had to be increased. To reach this goal, the occurrence times of the jobs were adjusted by a fixed factor that was determined for each simulation run. The adjusted occurrence times created an occurrence speed of jobs that would create 200% cluster utilization if all jobs would use their full estimated runtime. The resulting overload allows evaluating the overbooking strategy.

Generated Job Information

The simulation requires a *deadline* for each job. Such a deadline was not available in the job traces because a cluster normally operates on a best effort basis. Therefore, the deadline had to be guessed. It was assumed that users would not pay money for a job if they would not get a defined QoS in terms of a guaranteed job end. A user wants his job to be executed as fast a possible. Consequently if the provider will not be able to provide an early deadline, the user will choose another provider even if it costs more money. For the simulation, the deadline was defined to be twice the estimated runtime ($d = o + 2x$).

The simulation environment allows to define a release or earliest-start time for a job. The release time is normally as soon as possible. However, in practice it might be in the future because a job can have dependencies to other jobs or just needs time for the stage-in of huge amount of input data. This release time is not part of the traces. Therefore, it had to be guessed. For a simplification of the simulation, it was assumed that the release would be as soon as possible. Therefore, it was set to the occurrence time ($e = o$). For future work, it is however possible to define release times in the future. The simulation is capable of such a task.

Information about SLA *charges/fees* and *penalty* are also not contained in the job traces. However in SLAs, this information is paramount. The fee for a successful job execution describes the profit for the provider, and the penalty must be paid if violating the deadline.

Due to the distinct market mechanisms described in Section 3.5, fee and penalty of an SLA could be defined based on the requested resources, the ratio of supply and demand, or the user can describe it himself. A realistic guess about the height and the ratio of fees to penalties is important because the results are dependent on this assumption. The challenge here is, that even the actual commercial cloud providers like the Amazon EC2, demand fees for the use of their resources but do not offer any penalty for a violated SLA[1].

[1] Amazon EC2 Cloud Pricing: `http://aws.amazon.com/en/ec2/#pricing`

Table 5.2: *The used traces for the PWA simulations*

Trace	Resources	Jobs	Complete Util.	Jobs/statistics	Input Util.	Occurrence Factor
CTC	430	79,302	66.2%	31,509	29.67%	0.15
HPC2n	240	527,371	72%	181,121	20.18%	0.1
LANL	1024	201,387	75.2%	50,956	51.4%	0.26
SDSC-BLUE	1152	243,314	76,2%	138,553	23.5%	0.1175
SDSC-DataStar	1664	96,089	62,8%	30,708	21.54%	0.1
SDSC-SP2	126	73,496	83.5%	17,060	62.14%	0.3107

Therefore, in the simulations with PoF thresholds, the simulation environment defines the fee and penalty, based on the requested resources' quantity, as one virtual coin (VC) for each booked CPU hour (fee = penalty = nx). For the risk based simulation, it was decided to run 4 simulations with four different ratios of fees to penalties. This range of fees to penalties should cover the range from a low penalty, smaller than the fee, to penalties higher than the fee. The range begins with a ratio of $\frac{1}{2}$ (penalty = $\frac{1}{2}$fee), continues over a ratio of 1 (penalty = fee), and a ratio of 2 (penalty = $2 \times$ fee), and ends with a ratio of 4 (penalty = $4 \times$ fee).

5.1.2 Parallel Workload Archive

The archive contains several workload logs. However, not every trace contains estimated runtimes. These, though, are necessary to evaluate the overbooking approach. As a consequence, all traces without user estimations were omitted. From the remaining traces, all jobs without user runtime-estimations or missing required information were removed before the simulation. The necessary information were occurrence time, resources, and estimated and real runtime. Consequently in some traces, so many entries had to be removed that they at the end had a too low utilization and were not useful for the simulation anymore. As a result, the traces CTC, HPC2n, LANL-CM5, SDSC-BLUE, SDSC-DataStar, and SDSC-SP2 were selected for the simulation. The simulation used the last 20,000 jobs of the traces in 20 test batteries of 1000 jobs. The PDFs were based on the jobs, prior to the 20,000 simulation input jobs.

Job Trace Information Table 5.2 shows the key information of the traces that were used as simulation input. The first row of the table describes the CTC trace. The cluster system had 430 nodes and 79,302 jobs that created a utilization of 66.2%. From the 79,302 jobs every job was removed where information like occurrence time, resources, and estimated and real runtime was missing. After removing the not usable jobs, 51,509 jobs remained. 31,509 Jobs were taken to learn the PDFs. The remaining 20,000 jobs still had a utilization of 29.67%, thus the occurrence rate of the jobs was adapted by a factor of 0.15 to create the desired utilization of 200%. This means, the original occurrence time was multiplied with 0.15. In the traces, the first job has a occurrence time of 0 seconds and the occurrence time for all following jobs is the real distance in which jobs were submitted. Multiplying this time with a fixed factor keeps the behavior that jobs occur in bursts but the jobs occur nearer to each other and thus the utilization increases. The other traces were treated the same way.

5.1.3 Arminius

In addition to the simulations with the Workload Archive, a job trace of the Arminius cluster system was used. The simulation with the Arminius trace used the jobs from 2007 to create the statistics and the first 60,000 jobs from 2008 to implement the simulation. The jobs were submitted in 60 test-bulks of 1000 jobs. Due to the fact that OpenCCS is a planning based resource management system, every job contained a runtime estimation. Every job entry that was missing information was deleted. In addition, every advance reservation and every submission with more than one *executable* entry had to be removed.

The Arminius cluster had 200 nodes and 33,318 jobs in 2007 that created a utilization of 86.9%. After removing the unusable advance reservations or submissions with more than one job, 33,245 jobs remained and were taken to learn the PDFs. From the traces in 2008, 60,000 jobs were taken as input for the simulation. The average utilization of the 2008 cluster was 91.22%. The remaining jobs lead to a utilization of 44%; thus, the occurrence rate of the jobs was adapted by a factor of 0.22.

5.1.4 Acceptance Tests

The simulation uses the two acceptance tests from Section 3.1.3. The applicability of the overbooking strategies should be shown using the example of commercial markets with money exchange and scientific environments where no money is paid.

Acceptance Tests Based on $\mathbf{PoF_{max}}$ The acceptance based on probabilities is applied with increasing PoF_{max} thresholds. The thresholds were increased, from $PoF_{\text{max}} = 0.05$, in 5% steps, to $PoF_{\text{max}} = 1$. The goal of the PoF based acceptance test is to show that overbooking can be applied in an environment where no money is exchanged for instance, in academia. Additionally, the aim is to find appropriate PoF_{max} values.

Acceptance Based on Risk and Opportunity The acceptance based on risk and opportunity applies the equation: *Do not accept jobs where the risk (PoF of the job multiplied by the penalty) is higher than the opportunity (PoS of the job multiplied by the charge).* The simple equation with (Risk $<$ Opportunity) from Section 3.1.3 could have little influence on the results because nearly the same value of won money would be lost. Therefore, a *SecurityFactor* was added to the equation. The SecurityFactor should ensure that the overbooking was able to win more money than to loose. The applied formula is shown in Algorithm 5.1.

For the simulations, the chosen *SecurityFactor* was 2 for the heuristic overbooking strategy and 4 for the comprehensive strategy. The values were chosen as an educated guess based on former simulations. The *SecurityFactor* is a means to take more or less risk into the overbooking. In practice, it has to be adapted to the provider's strategy.

Algorithm 5.1 Applied risk acceptance test.

if $PoS \times Charge > PoF \times Penalty \times SecurityFactor$ **then**
  accept the SLA.
else
  reject the SLA.
end if

5.2 Implementation

This section describes the implementation of the simulation. The simulation was implemented in Matlab[2] because it is a high level programming language with a broad support of optimized libraries for mathematical calculations. For instance, the quite frequently used convolutions are supported as functions on the basis of fast Fourier transformations (FFT).

This section starts with an overview of the used data structures. In the following, first the event system and then the placing routines for jobs are described.

5.2.1 Used Data Structures

The simulation manages several general data structures. There are structs for *incoming jobs*, *planned jobs*, *dropped jobs*, *successful jobs*, and *failed jobs*. Each entry for a job contains information about the *job-ID*, *occurrence time*, *release time*, *deadline*, *estimated runtime*, *basic PDF*, the *real runtime*, the *planned start time*, the *calculated PDF*, the *calculated PoF*, a *boolean value if the job is overbooked*, the *planned finish time*, the *requested number of resources*, and the *number of assigned resources*.

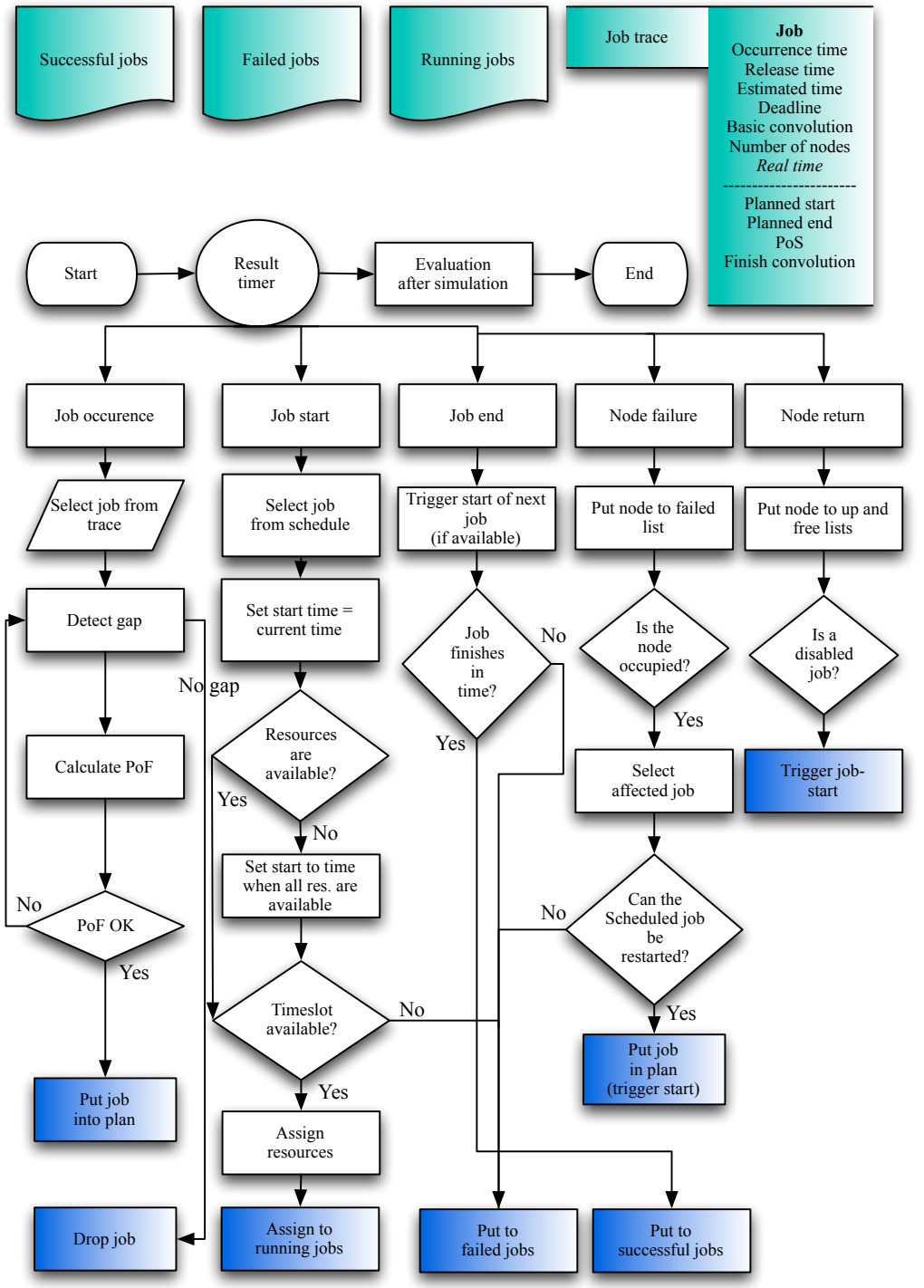
In addition, the simulation manages the current plan. It consists of data structures for the running jobs and the disabled jobs. Disabled jobs were already running and became disabled because node(s) crashed. These job(s) can be restarted if there is time available until the deadline. Two lists for the availability of resources are used to keep track of the node crashes. In the comprehensive overbooking approach, each resource has its own plan. It is needed because the allocation of every single node has to be mapped. The plan keeps track of every job that does or will run on a node. The entry for each job contains the job-ID, start time and the end time. This allows us to always know which resource is free at each time.

5.2.2 Events

The simulation is steered by an event system. The events are triggered at defined points in time. Each event has a corresponding procedure that consists of one or more tasks. The simulation executes the tasks and evaluates the results. Figure 5.1 gives an overview of the events.

[2]Matlab, `http://www.mathworks.de/`

Figure 5.1 The simulation event processing.



Events can be:

- Occurrence of a job
- Start of a job
- End of a job
- Node failure

- Node repair

The following describes the functionality of the procedures to handle the events.

Job Occurrence

At job occurrence, the scheduler tries to place the job in the current plan. The job can start at its release time and has to finish before its deadline. In between release and deadline, the job can be moved on the resources.

Depending on the scheduling strategy, the job gets is whole runtime (comprehensive backfilling, heuristic planning) or it can be overbooked (comprehensive overbooking, heuristic overbooking). For all strategies, the job is killed at the start of the next planned job.

To reach this goal, the method detects feasible gaps in the plan. In case of backfilling or heuristic planning, the gaps have to be the size of the estimated runtime. For overbooking, the PoF for placing the job in a gap that is too short has to be calculated. When the PoF is acceptable, the job is selected. Otherwise, the next gap is selected until no other gap is found. In this case, the job is counted towards the amount of dropped jobs.

Job Start

When a job start event occurs, the corresponding job is selected. Then, it is checked whether or not the resources are free. It is possible that there is a lack of free resources at a planned job-start because some resources might have crashed. If enough resources are free, the start time is set to the current time, and the job is assigned to the resources. If not enough resources are free, the job is placed as a disabled job. Depending on the strategy, jobs on the disabled list can be delayed or lost. In strategies without overbooking, the whole runtime is assigned to the job. A job is lost when the time until the deadline is shorter than the user estimation. In the overbooking scenario, the job can start later when the resource returns because it can run shorter and still finish before the gap ends.

Job End

If the job is running and the end of the job is reached, the result is determined.

- The job is successful when the end of the actual runtime is reached.
- It is not successful when the next job should start and the job has not reached the end of the computation. In that case, it has to be killed.

However when the job got the full maximal estimated runtime, the SLA is fulfilled because the user misestimated the execution time.

- If the job finished successfully, it is counted towards the amount of successful jobs.
- If the job is on the disabled list, it is counted as failed.

If a following job is scheduled, the start event for the job is triggered.

Node Failure

When a node failure event occurs, it is checked if the node is occupied. If the node is not occupied, it is just put to the list of crashed nodes. If the node is occupied, the assigned job is moved to the disabled job-list and the job's resources are freed. The heuristic additionally tries to replan the job directly.

Node Repair

If a node repair event occurs, the node becomes part of the amount of available nodes. The comprehensive overbooking algorithm detects if there is a disabled job assigned to this node. When there is such a job, its start event is triggered. The heuristic replans with the additional resource.

5.2.3 Placing Routines

Several placing strategies could be applied to the overbooking algorithms. First fit (FF), more precisely bottom left first is one of them. The advantage of FF is that the algorithm is fast because it ends when a gap is found. Only if no suitable place for a job is found, all possible gaps have to be checked.

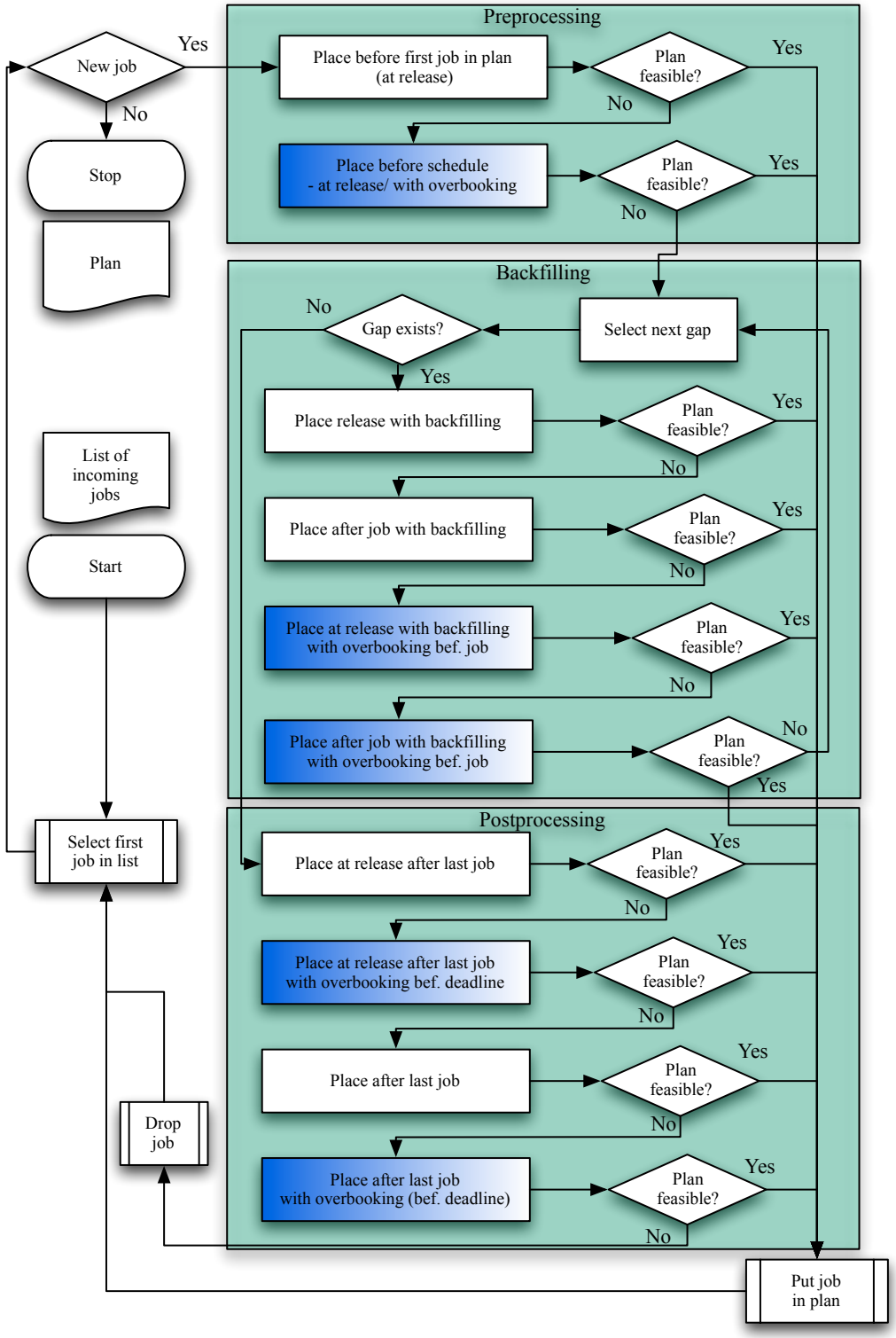
On the other hand, best fit (BF) provides better results. However, every possible time slot hast to be checked to select the best one. For the simulation, FF and BF were implemented and are described in the following.

First Fit

Figure 5.2 exemplarily shows the applied first fit (FF) placing. Independent of the overbooking strategies, three steps occur. The differences are that the comprehensive approach selects resources directly and the heuristic planning counts nodes overall. In the beginning, a job is selected and the first gap in the plan is detected.

The first step is the *pre-scheduling* step, which handles the empty plan as well as jobs that start before all others. The second case, *backfilling*, handles the backfilling part, where the algorithm tries to place the job in between others in the plan. The last part, *post-scheduling*, is applied if the job is the last one in the plan. The transparent blue parts in Figure 5.2 are applied in the overbooking cases only. They are not used if the algorithms run without overbooking. When the algorithms found a suitable gap, the job is put into the plan and is accepted. When a gap is not sufficient, the next gap is searched, selected, and checked. Only if no suitable gap can be detected, the job is dropped.

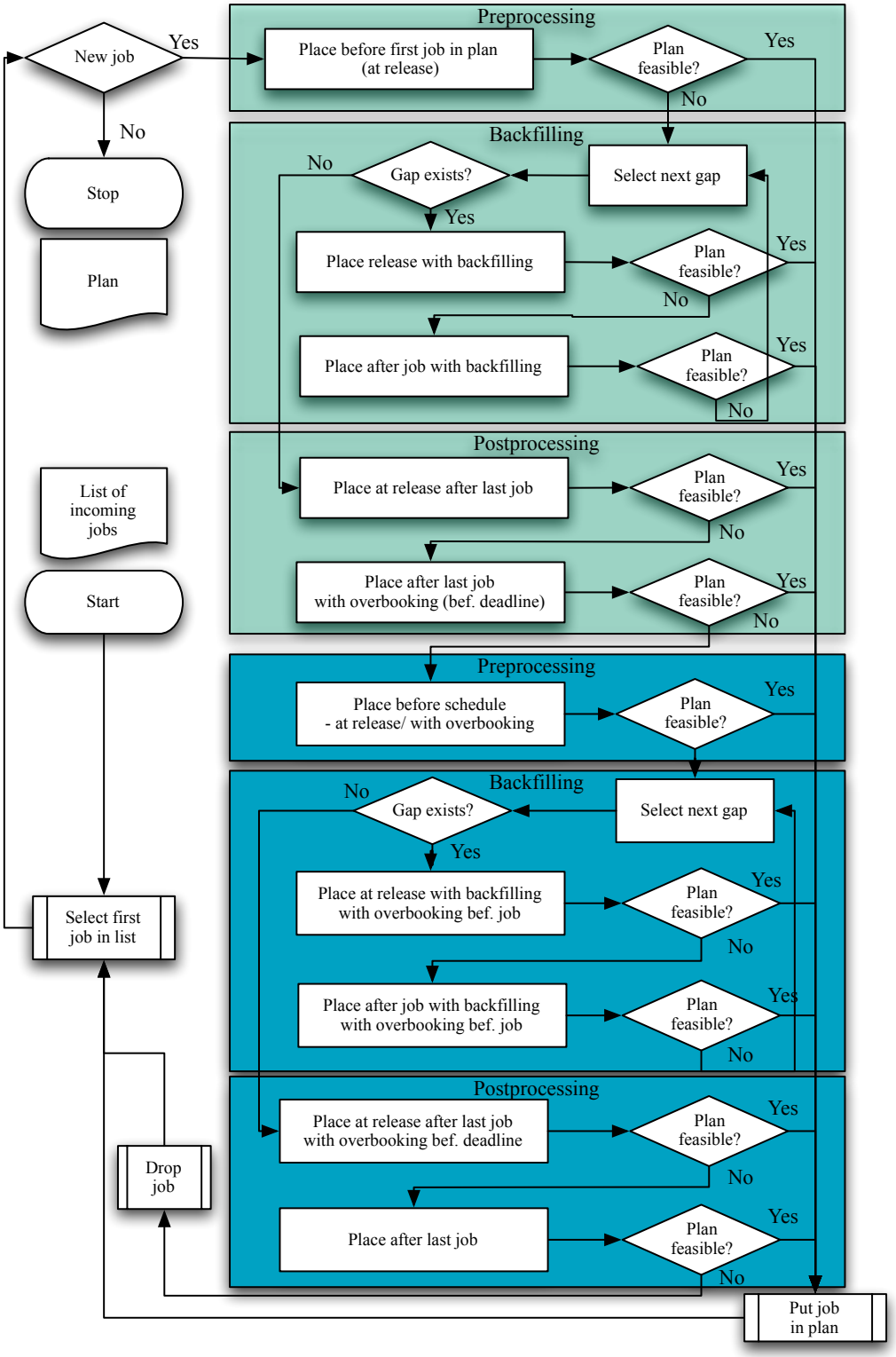
Figure 5.2 The process for the first fit job placing.



Best Fit

In theory, best fit (BF) provides the better results [John 74b]. For the simulations, a near BF solution was easily applied.

Figure 5.3 The process for the best fit job placing.



The algorithms first try to place a job without overbooking, and only in case this is not possible, the FF approach with overbooking is chosen. The BF algorithm

is shown in Figure 5.3. Basically, it first uses the standard algorithm without overbooking and then the overbooking parts of the FF approach.

6 Evaluation with PWA Traces

This and the next chapter present the abilities of the overbooking strategies. This chapter shows the results from simulations with job traces from the Parallel Workload Archive (PWA).

Contents

6.1 CTC 86
6.1.1 PoF Acceptance Test 88
6.1.2 Risk Acceptance Test 96
6.2 HPC2n 103
6.2.1 PoF Acceptance Test 104
6.2.2 Risk Acceptance Test 106
6.3 LANL 109
6.3.1 PoF Acceptance Test 110
6.3.2 Risk Acceptance Test 112
6.4 SDSC-BLUE 115
6.4.1 PoF Acceptance Test 116
6.4.2 Risk Acceptance Test 118
6.5 SDSC-DataStar 121
6.5.1 PoF Acceptance Test 122
6.5.2 Risk Acceptance Test 124
6.6 SDSC-SP2 127
6.6.1 PoF Acceptance Test 128
6.6.2 Risk Acceptance Test 130
6.7 Summary 133
6.7.1 PoF Results 133
6.7.2 Risk Results 135
6.7.3 Runtime 138

The different statistical sources for the evaluation of the benefits of overbooking were defined in the previous chapters. The job parameters like estimated duration, deadline, or the requested resources were described in Section 5.1 and the PDFs that were the simulation input origin in Section 3.2 and Chapter 4. The underlying assumption is that a well-defined input function should have a positive impact on the overbooking results.

Overbooking Based on Statistical Runtime Analysis The statistical runtime-estimation analysis was based on the user job-runtime-estimations in the PWA. The overbooking strategies were evaluated with the following traces: CTC, HPC2n, LANL, SDSC-BLUE, SDSC-DataStar, and SDSC-SP2. An important task for the analysis was grouping jobs in appropriate classes based on their estimated runtime. The following timeframes of the estimated runtime were used: under 10 minutes, 10 minutes to 1 hour, one hour to 2 hours, two hours to 3 hours, three hours to 5 hours, five hours to 12 hours, and more than 12 hours. The runtime-classes were defined and discussed in Section 4.2.1.

Overbooking Based on Statistical Resource Analysis The second analysis was based on the requested resources. The same traces from the runtime-estimation analysis were applied for the resource analysis. The statistical analysis was based on grouping jobs according to the amount of requested resources. The following resource groups resulted: 1 node, 2 nodes, 3 to 4 nodes, 5 to 8 nodes, 9 to 16 nodes, 17 to 32 nodes, 33 to 64 nodes, and more than 64 nodes. The resource-classes were defined and discussed in Section 4.2.2.

Evaluated Planning Strategies To reveal the benefit of the overbooking approach, the four scheduling strategies introduced in Section 3.6 were implemented.

- *Comprehensive Backfilling*
- *Comprehensive Backfilling with overbooking*
- *Heuristic planning*
- *Heuristic planning with overbooking*

For the evaluation, two different acceptance tests were applied

- one on the *PoF threshold* and
- one on the *risk assessment*

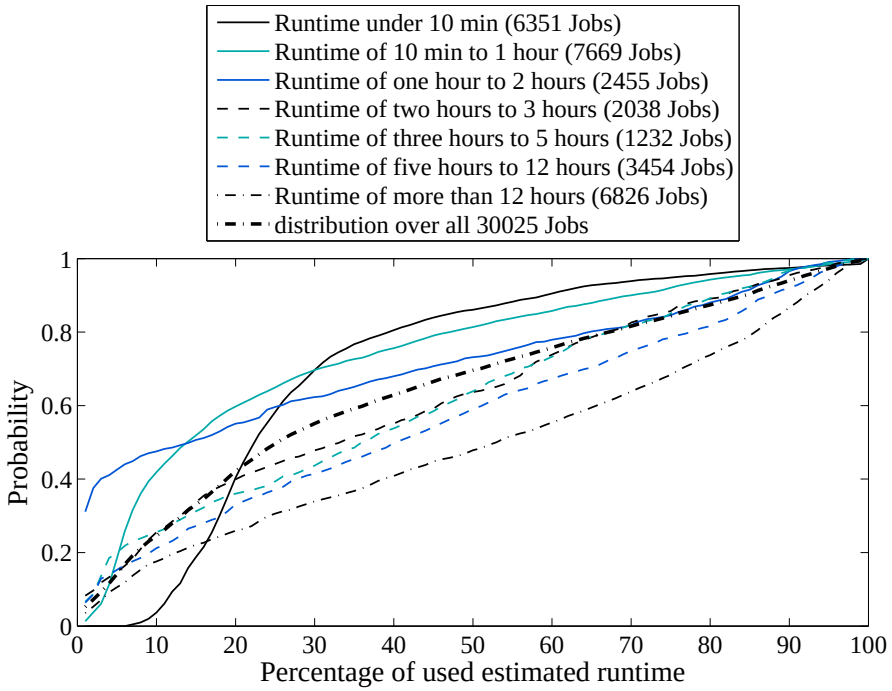
The results of the simulation are either the number of successful or failed jobs or the amount of sold compute power. The sold compute power is the number of resources multiplied by the booked runtime in hours. The compute power is counted in virtual coins (VCs). Each virtual coin corresponds to an estimated resource usage hour.

6.1 CTC

The first simulation was based on the Cornell Theory Center (CTC) trace from the PWA. The cluster system had 430 nodes and 79,302 jobs that created a utilization of 66.2%. From the 79,302 jobs, every job that was missing information like occurrence time, resources, and estimated and real runtime was removed. After removing the unusable jobs, 50,000 jobs remained. From this jobs 30,025 were selected to learn the PDFs. The remaining 20,000 jobs still had a utilization of

29.67% according to the runtime-estimations. Thus, the occurrence rate of the jobs was multiplied by a factor of 0.15. Therefore, the jobs occur about 6.6 times faster as in reality. This factor increases the estimated utilization to 200% (if all jobs would need the full estimated runtime). This overload allowed the evaluation of the overbooking strategies. In the following simulations, the other traces were treated the same way.

Figure 6.1 The runtime-estimation analysis of the CTC trace.



Input Statistics Figure 6.1 shows the CDFs calculated from the statistical runtime-estimation analysis of the CTC trace. These CDFs, or more precisely the corresponding PDFs, were the input for the first part of the simulation. It was based on user runtime-estimations. The line plots describe the different runtime classes. In the legend, the number behind each entry shows how many jobs were assigned to this class. The line for *Runtime of one hour to 2 hours (2,455 Jobs)* starts at 0.3. This means that in this group three out of ten jobs ended directly at the beginning of the execution. In contrast, the jobs with a *Runtime under 10 min (6,351 Jobs)* used at least 8% of their estimated runtime. The last line in the legend shows the distribution over all jobs *(30,025 Jobs)*.

Figure 6.2 shows the functions that were created by the amount of booked resources. These CDFs were the input for the second part of the simulation.

When comparing the CDFs, one can see that the spreading of the plots for the runtime-estimation based CDFs seems to be wider than that of the resource CDFs. As a consequence, the use of functions from the runtime-estimation analysis might produce better results for the simulation.

Figure 6.2 The resource analysis of the CTC trace.

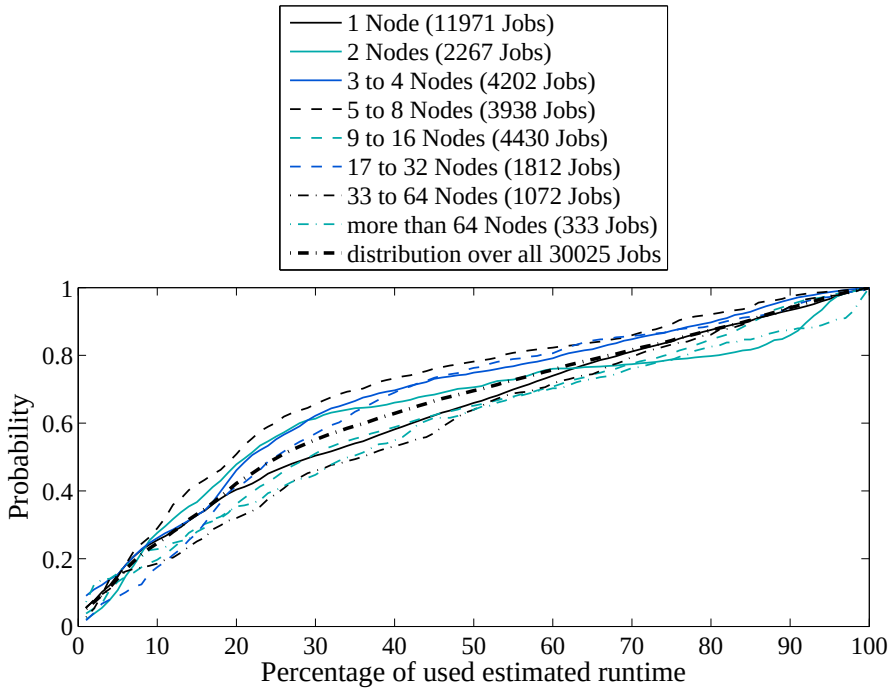


Table 6.1: *The PoF_{max} gain with the CTC trace.*

PoF simulation	runt. est.	plus	$PoF_{\max}$	resources	plus	$PoF_{\max}$
Comp. backfilling	14,260			14,260		
Heuristic planning	23,860			23,860		
Comp. overbooking	23,300	63%	0.1	23,110	62%	0.6
Heur. overbooking	28,280	19%	0.15	27,330	15%	0.6

6.1.1 PoF Acceptance Test

This section presents the results of the simulation with the acceptance tests based on the $PoF_{\max}$ threshold. For this and the following PWA traces, the simulation started with the two, before mentioned, statistical analyses based on the booked runtime and the chosen resources. Table 6.1 summarizes the maximum gain for both simulations. The first column lists the evaluated strategy. The second column shows the maximum result of the simulation with the runtime-estimation based simulations. This maximum result is the average value of the 20 simulation runs for a given $\text{PoF}_{\max}$ threshold. This threshold is written in the 4_{th} column. The third column shows the surplus of the overbooking strategies in percent. The columns 5 to 7 show the corresponding results of the simulations based on the resource CDFs. Now follows a discussion of the results in more detail.

Results of the Runtime-Estimation based PDF simulations The input statistics for the simulations were those illustrated in Figure 6.1. The achievable jobs of the simulation with 20 test batteries of 1000 jobs are pictured in Figures 6.3 and 6.4.

Figure 6.3 (a) contains eight lines. Four lines for the successfully executed jobs and four lines for the failed ones. The x axis of the figures describes the PoF_{max} threshold. It was increased in 5% steps from 0.05 to 1. The transparent error areas around the values are 95% confidence intervals. The y axis shows the number of jobs.

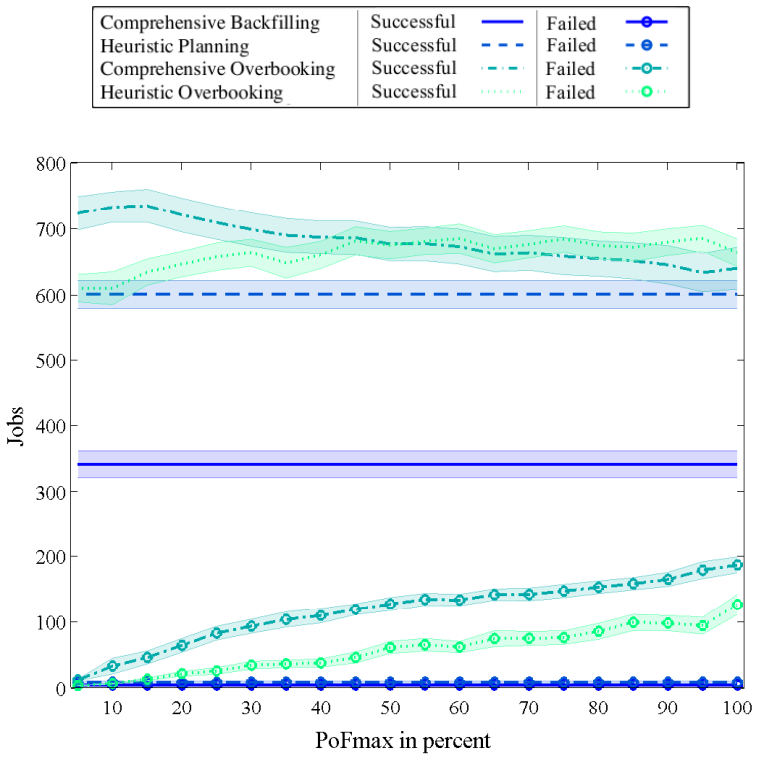
For the successful jobs, two lines, one for the *heuristic planning* and the other for the *comprehensive backfilling*, are horizontal and never change. This is due to the fact that these two strategies do not overbook jobs and also do not consider the results of a PoF calculation. A job is accepted when it completely fits into a gap. One can directly see that the heuristic allowed a more profitable result than the comprehensive backfilling approach. The reason is that the heuristic replans all jobs after each job's end and, consequently, can accept new jobs when others ended earlier. The comprehensive backfilling assigns the resources to their jobs for the entire booked runtime. The comprehensive backfilling accepted 330 and the heuristic planning 600 of 1000 possible jobs.

Two other lines for the successful jobs are for the two overbooking strategies. The successful jobs changed depending on the PoF_{max}. In the beginning, the number of successfully executed jobs increased with a higher accepted PoF_{max} because new jobs could be accepted and the maximum failure probability was low. The accepted jobs were successful. Eventually at a certain threshold, the number of successful jobs decreased. This could be caused by two reasons. On the one hand, additional jobs were accepted, but due to the higher PoF_{max} it was more likely that they fail and, on the other hand bigger jobs with more resources and therefore a higher PoF were accepted. These bigger jobs prevented the acceptance of other smaller ones. Consequently, the overall number of successful jobs fell. For the comprehensive overbooking, the most jobs were accepted with a PoF_{max} of 0.15 (See Figure 6.3(a)). For the heuristic overbooking, the number of jobs increased, and from PoF_{max} of 0.45 on, it was relatively stable. It varied within the 95% confidence interval. The maximum number of successful jobs for the heuristic was 680 out of 1000 and for the comprehensive overbooking, it was 740.

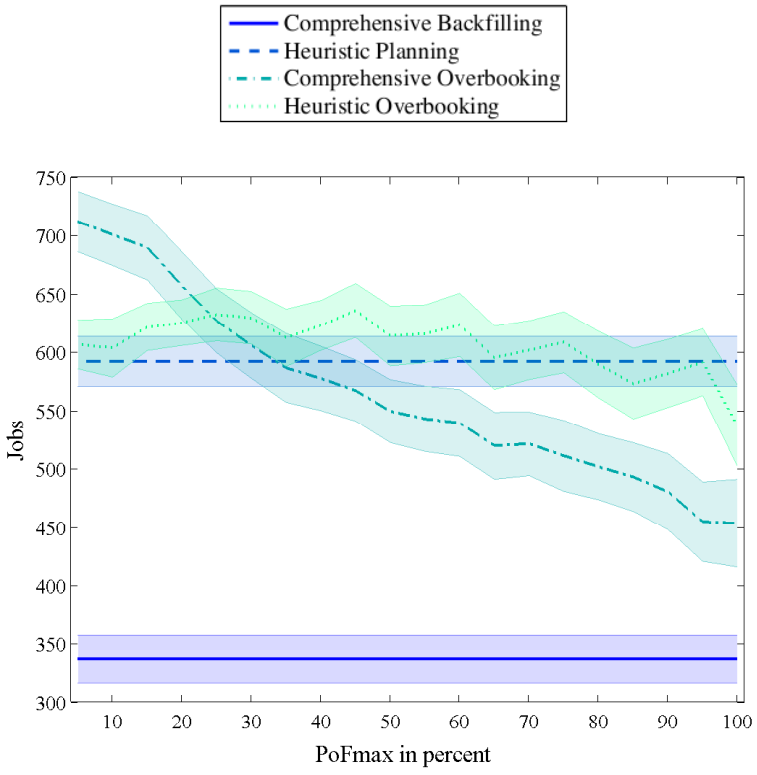
For the failed jobs, the lines for heuristic planning and comprehensive backfilling never change. A job that is not overbooked fails because one of its resources crashed. In this simulation, nearly no job failed this way. For both overbooking approaches, the number of failing jobs steadily increased. This was caused by the failing jobs that were accepted with higher PoF. The amount of failed jobs of the comprehensive approach was always higher than the amount of failed jobs of the heuristic.

Figure 6.3(b) shows the difference of the successful minus the failed jobs. As mentioned above, the non-overbooking strategies did not change. For the overbooking strategies, the number of jobs increased in the beginning with increasing PoF_{max} due to a higher number of accepted and successful jobs. It then fell

Figure 6.3 CTC: Jobs with PoF acceptance test and runtime-estimation based PDFs.

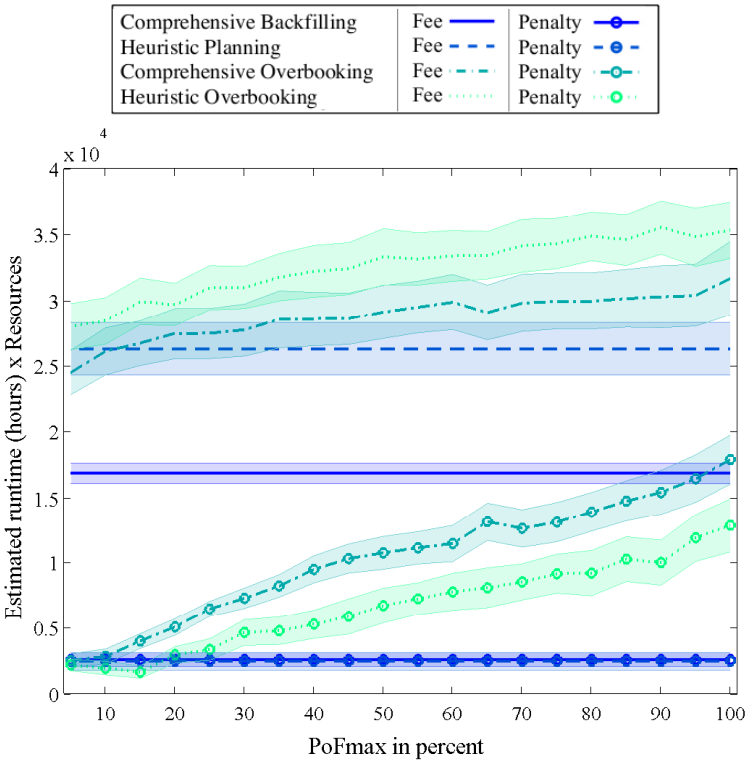


(a) The successful and failed jobs of the CTC trace.

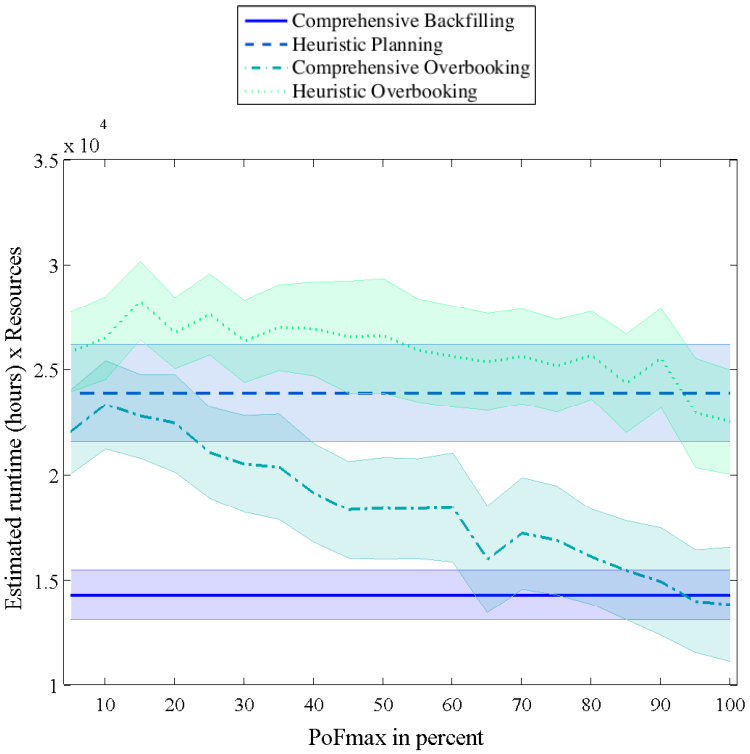


(b) The difference of successful and failed jobs.

Figure 6.4 CTC: Profit and penalties with PoF acceptance test and runtime-estimation based PDFs.



(a) The fees and penalties of the CTC trace.



(b) The gain.

because more of the newly accepted jobs were failing or fewer but bigger jobs were accepted. For the comprehensive approach, the maximum number of these jobs was at $PoF_{\text{max}} = 0.05$ and then steadily fell. For the heuristic overbooking, the number of jobs in tendency increased until an $PoF_{\text{max}} = 0.45$ and afterwards tended to fall until a $PoF_{\text{max}} = 0.95$. From this value on, the number of jobs fell straight. Interestingly, the plot does not behave monotonic (firstly increasing and then falling). Instead, it fluctuates albeit in the borders of the confidence intervals. This means that sometimes bigger jobs were chosen that allowed a higher utilization. A higher amount of simulation runs should create fewer fluctuating curves. However, due to the limited amount of jobs in the traces this was not possible for the PWA based simulations. The Arminius based simulations in the next chapter used more jobs.

More important than the number of jobs is the generated profit of the strategies. The y axis in Figure 6.4(a) shows the fees in virtual coins for successful jobs and the penalties for the failed. A virtual coin (VC) is one booked node hour. Figure 6.4(b) shows the gain (fees - penalties).

The successful jobs and fees of the overbooking approaches show an interesting difference. While the successful jobs only increased in the beginning, the fees always did. This means that due to a higher PoF_{max} not more but bigger jobs were accepted. The bigger jobs had a higher failure probability because they used more resources for a longer time. They were accepted with a higher PoF_{max} and prevented the acceptance of some shorter jobs that were accepted before. While in Figure 6.3(a) almost no jobs seemed to have failed due to resource outages, one can see that there were jobs that failed by the penalties that had to be paid for them.

Figure 6.4(b) points out the combined gain (fees - penalties). For the non-overbooking strategies, the gain did not change. For the overbooking strategies, the gain increased in the beginning due to the higher amount of successfully accepted jobs, and after the peak value, the gain fell because more of the additionally accepted jobs were failing.

This indicates that the underlying statistical functions were accurate and the PoF of the jobs was well estimated. With a low threshold, most additionally accepted runtime was successfully sold and when the accepted risk became too high, more and more additionally accepted runtime failed. If the underlying statistic would not have been accurate, the shape of the plot would be different. If the PoF would be underestimated, the additional accepted jobs would fail at the beginning and the gain decrease because the strategy would be too offensive. If the PoF would be overestimated, no additional runtime would be accepted at the beginning or no runtime would fail at the end because the strategy was too conservative.

In this setting, the maximum gain for the heuristic was 28,280 VCs with a PoF_{max} of 0.15. For the comprehensive overbooking, the maximum gain was 23,300 VCs at $PoF_{\text{max}} = 0.1$. From a PoF_{max} of 0.95, the gain dropped below the results of the heuristic planning approach. The comprehensive approach showed the same behavior. This was caused by the fact that the maximum threshold of $PoF_{\text{max}} = 1$ did not mean that every job would fail. Even in this setting, many jobs with a

lower PoF, or full estimated runtime, were accepted and successful. In addition, if the schedule was full and no gap with enough resources was available, no more jobs could be accepted even with $PoF_{\text{max}} = 1$.

Comparing the non-overbooking strategies, the heuristic planning earned 70% more money compared to the backfilling because the heuristic was more flexible. If comparing both overbooking strategies, the heuristic overbooking earned 22% more profit than the comprehensive overbooking. The difference, between 22% and 70% additional profit shows that the comprehensive overbooking earned more additional profit than the heuristic overbooking. The result can be seen in Table 6.1. The comprehensive overbooking increased the profit by 63% and the heuristic by 19%. With other words, overbooking is more profitable for worse scheduling strategies.

In addition, one can see an important difference between the jobs and the resulting fees. With the same incoming jobs, the heuristic overbooking sold more resources and made more profit than the comprehensive overbooking. This was caused by the fact that the heuristic replanned all jobs after a job's end, while the comprehensive approach only moved the jobs on the assigned resources. This means that the heuristic had a less fragmented schedule, could accept bigger jobs, and thus it successfully sold more compute time. The comprehensive approach had a more fragmented schedule that could be filled with a higher number of smaller jobs. In this setting, the flexibility of the heuristic was much higher than the comprehensive approach. The non-overbooking heuristic planning even had a better result than the comprehensive overbooking.

Results Resources Here, are the results of the simulation described that was based on an analysis of the number of booked resources and the corresponding runtime-estimation quality. The input statistics for calculating the PoF for overbooking are illustrated in Figure 6.2. Figures 6.5 and 6.6 plot the results.

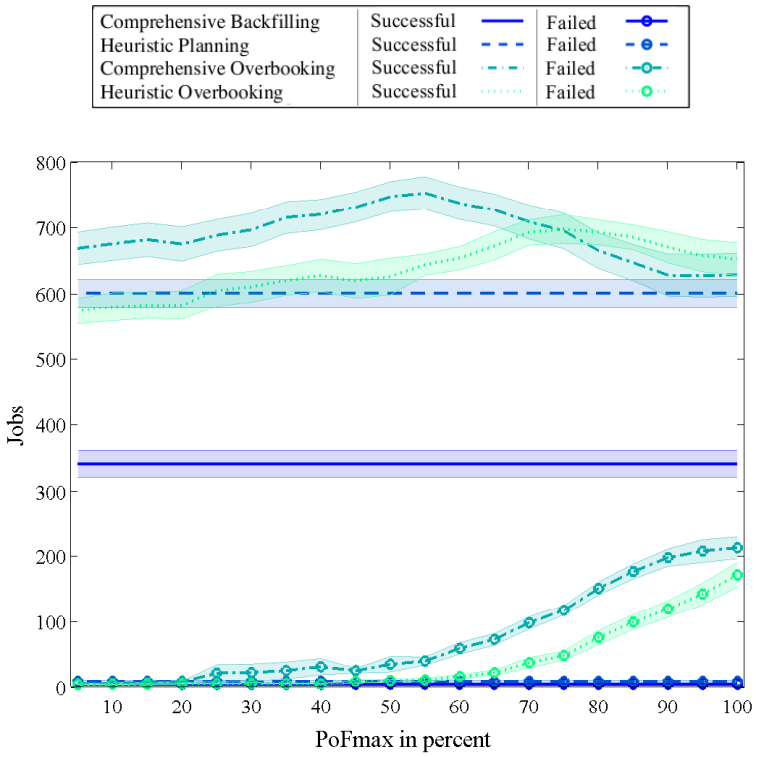
Figure 6.5(a) shows that with the PDFs based on the booked resources the number of accepted jobs increased over a longer period compared to the simulation with the input PDFs from the runtime-estimation frames.

The comprehensive overbooking had the highest number of successful jobs, about 750, with a PoF_{max} of 0.55. For the heuristic overbooking, with a PoF_{max} of 0.75, about 700 jobs were successful. The number of failed jobs, for both overbooking strategies, remained low up to a PoF_{max} of 0.55.

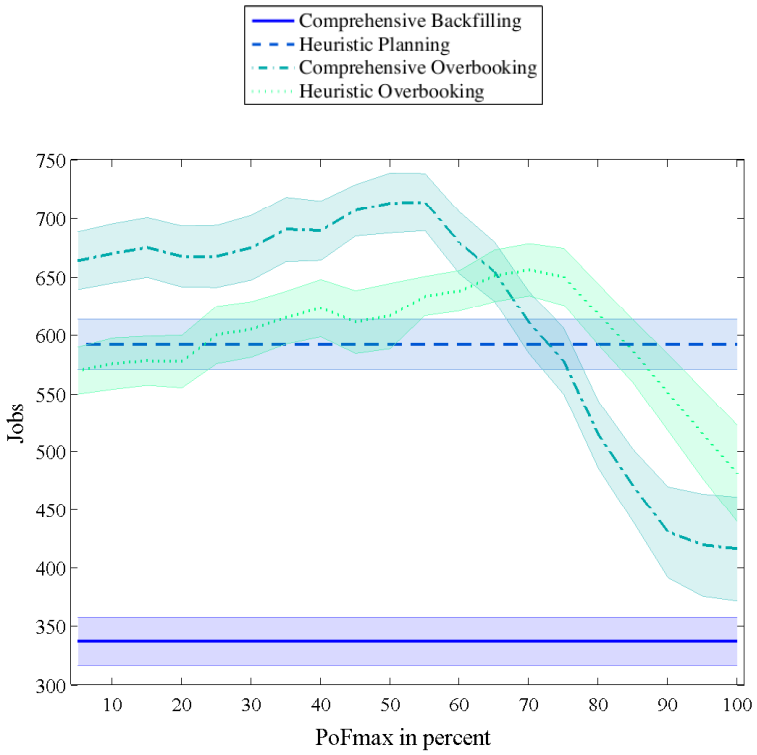
Figure 6.5(b) shows the difference of successful minus failed jobs. Here, one can see that for the comprehensive strategy the maximum difference of jobs was 700 at a PoF_{max} of 0.55. The heuristic approach had 650 jobs with a PoF_{max} of 0.7. Following to the peak values, the plots decrease directly. The number and difference of jobs for the strategies that did not overbook were the same as for the simulation with runtime-estimation based PDFs. The reason is that the input data for the simulation was exactly the same. The different PDFs have only an influence on the results of the overbooking approaches.

Figure 6.6(a) shows the fees for successful jobs and the penalties for the failed ones. Figure 6.6(b) shows the gain.

Figure 6.5 CTC: Jobs with PoF acceptance test and resource based PDFs.

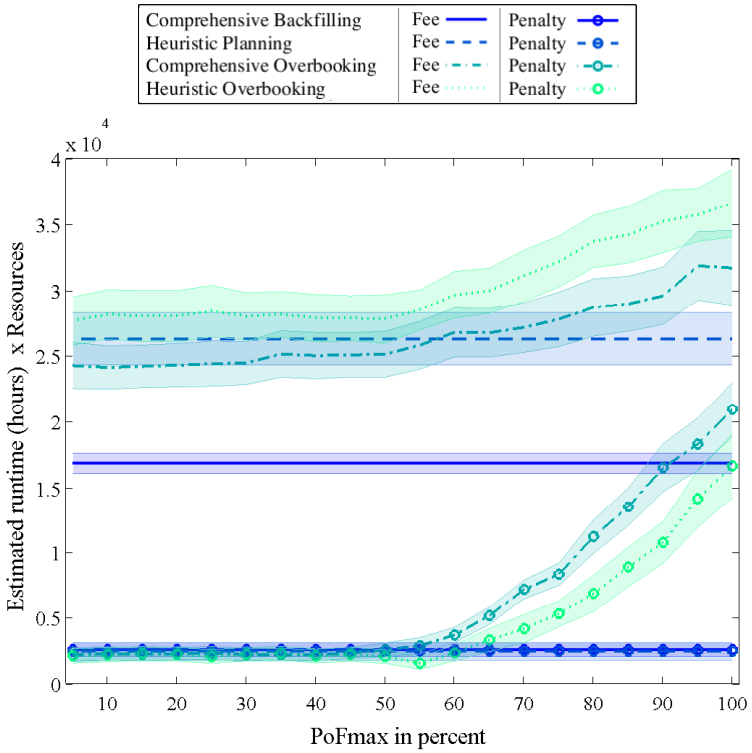


(a) The successful and failed jobs of the CTC trace.

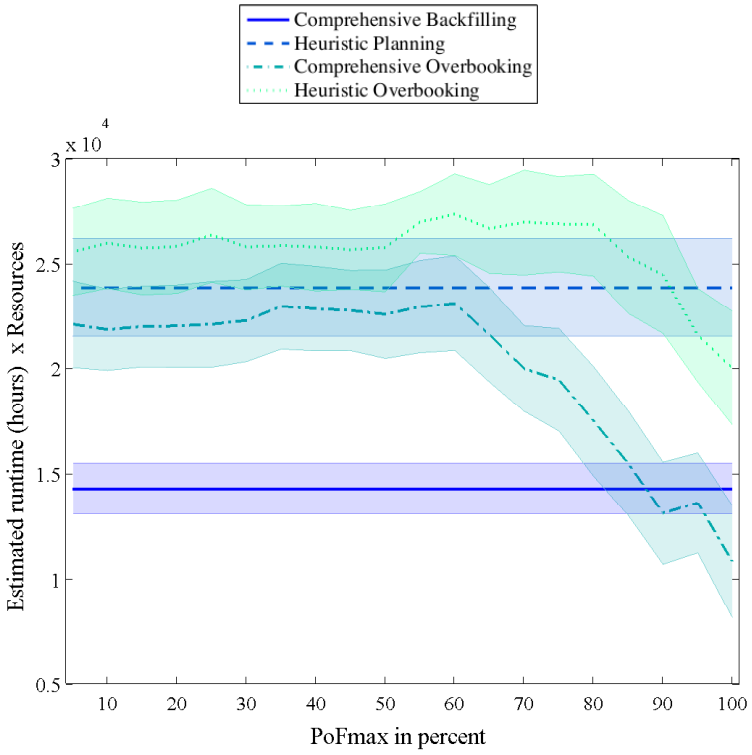


(b) The difference of successful and failed jobs.

Figure 6.6 CTC: Profit and penalties with PoF acceptance test and resource based PDFs.



(a) The fees and penalties of the CTC trace.



(b) The gain.

Figure 6.6(a) shows that the fees were steadily increasing with a higher PoF_{max}. The comprehensive overbooking decreased beginning from a PoF_{max} of 0.95. The penalties remained on the level of the non-overbooking approaches or were even lower until a PoF_{max} of 0.55. This means that even for a high PoF_{max} threshold accepting new jobs is beneficial. Thus, the PoF estimation process is conservative and tends to overestimate.

The reason that less jobs for the overbooking approaches failed than for the corresponding non-overbooking is that the non-overbooking approaches fail if one of their resources fails. The overbooking strategies are allowed to plan the jobs with less runtime and can therefore restart jobs after a node failure even if the full runtime is not available anymore. If a before crashed job takes less runtime than assigned, it still can be successful.

Figure 6.6(b) shows the gain of the simulation with the resource based PDFs. The shape of the curves in this figure indicates that the underlying statistical functions are quite accurate. With a low threshold until 0.6, runtime was additionally successful sold and when the accepted risk became too high (PoF_{max} above 0.6) more and more additionally sold jobs failed. Here again, the heuristic without overbooking was better than the comprehensive overbooking approach. The heuristic overbooking had a maximum gain of 27,330 VCs with a PoF_{max} of 0.6. The comprehensive overbooking had a maximum gain of 23,110 VCs with a PoF_{max} between 0.35 and 0.6. From a PoF_{max} of 0.9, both overbooking approaches were worse than the approaches without overbooking.

The profits of the runtime-estimation function based simulation were higher than the results of the resource functions. We already assumed this when we had a look at the input CDFs. The runtime-estimation CDFs were distributed wider than the resource based CDFs. This means that the effect of the chosen groups is better if the groups are more divergent. Homogenous group classes are less supportive for overbooking. The runtime-estimation PDF simulation had a maximum gain of 28,280 VCs compared to 27,330 VCs with the resource based PDFs.

6.1.2 Risk Acceptance Test

This section contains the results of the simulation with the CTC trace and a risk based acceptance test according to Section 3.1.3. This simulation applied a commercial environment where the penalties for missing job deadlines are often higher than the fees. Thus in the commercial environment, jobs should be accepted if the opportunity (fee multiplied by PoS) is significantly higher than the risk. The opportunity should be as twice as high as the risk to yield realistic profits. For the comprehensive overbooking, the opportunity has to be four times higher than the risk. Table 6.2 summarizes the results of the risk estimation simulation. The table shows the average results of the 20 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent.

Table 6.2: *The risk simulation results with CTC trace.*

Risk simulation	Penalty Ratio			
Runtime based statistics	0.5	1	2	4
Comprehensive backfilling	15,535	14,260	11,710	6,611
Heuristic planning	25,071	23,862	21,444	16,608
Comprehensive overbooking	24,571	22,410	19,179	14,278
Additional comp. profit	58%	57%	63%	160%
Heuristic overbooking	30,148	27,528	23,756	19,279
Additional heuristic profit	20%	15%	10%	16%
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	15,535	14,260	11,710	6,611
Heuristic planning	25,071	23,862	21,444	16,608
Comprehensive overbooking	24,038	22,019	19,357	15,464
Additional comp. profit	55%	54%	65%	134%
Heuristic overbooking	26,776	25,886	23,623	19,785
Additional heuristic profit	7%	9%	10%	19%

Results of the Runtime-Estimation based PDF simulations The results of the simulation with the risk acceptance test and the CTC runtime-estimation statistics from Figure 6.1 are described in the following.

Figure 6.7(a) shows the results of the jobs with the risk acceptance test. In this simulation, the risk was built based on the calculated PoF and PoS and the fee and penalty of the SLA. In practice, it is likely that the penalty for violating an SLA is higher than the fee. Therefore, the simulation evaluated the behavior of the overbooking algorithms based on different ratios of penalty to fee. The simulation started with a ratio of 0.5 and doubled the ratio to 1, 2, and 4. The results are shown in bar graphs and the error bars illustrate 95% confidence intervals.

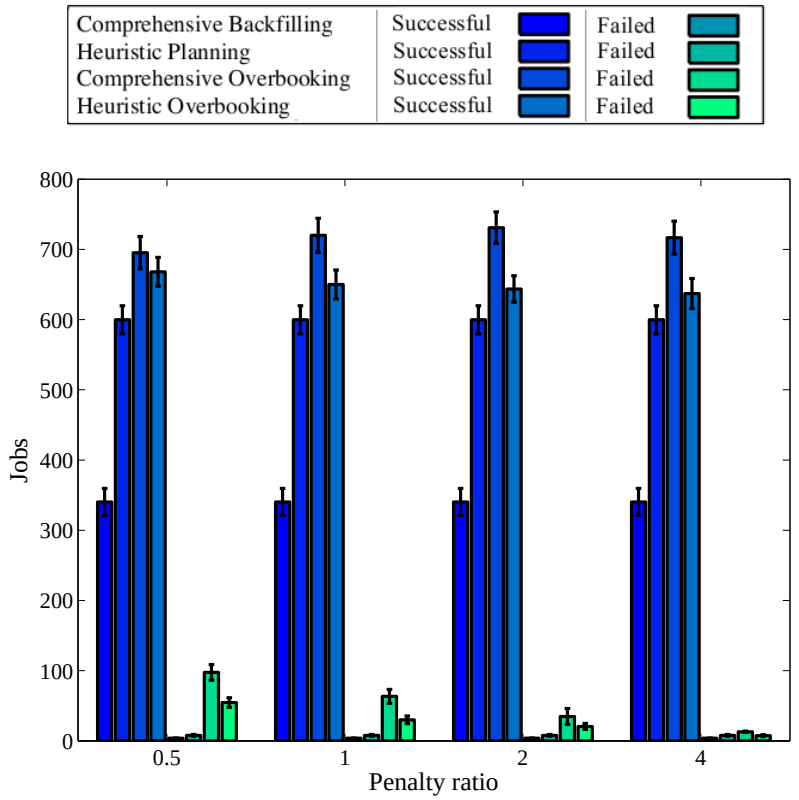
Similar to the runs before, the non-overbooking strategies were not affected by the different penalty ratios.

For the overbooking algorithms, Figure 6.7(a) shows that with an increasing penalty factor the risk increased. Consequently, the number of successful and failed jobs decreased because the algorithms tended to be more cautious. Due to the more cautious behavior, Figure 6.7(b) shows that the difference of successful to failed jobs increased.

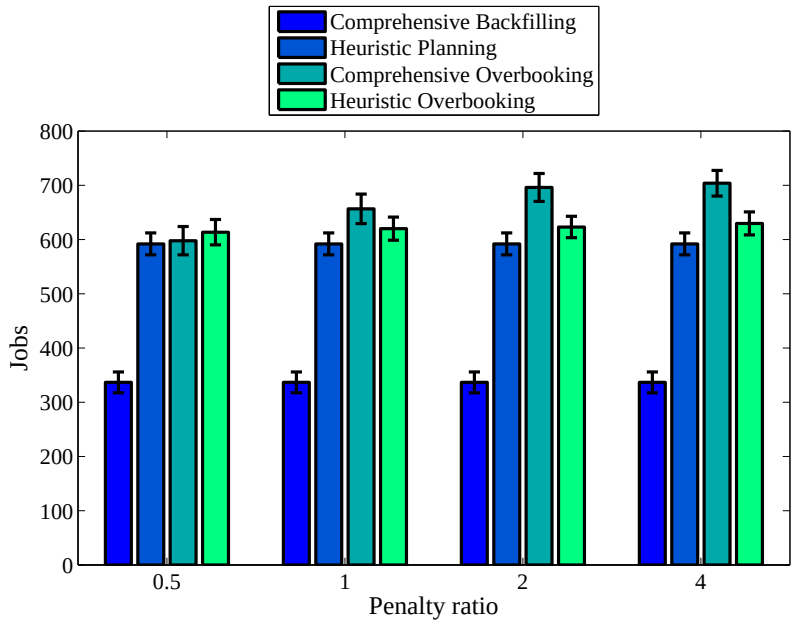
More interesting than the number of jobs was the resulting gain of the simulation. Figure 6.8(a) show the fees for successful jobs and the penalties for the failed. Figure 6.8(b) illustrates the gain (fees - penalties).

Figure 6.8(a) underlines that the fees for the non-overbooking strategies remained the same because nothing changes. For the overbooked jobs, however, it is shown that the fees decreased. This happened because fewer jobs were accepted and could be successfully finished. On the other hand, the penalties for the overbooking approaches only increased slightly even if each penalty factor doubled the penalty. The results of the overbooking approaches decreased less because fewer jobs failed due to the more failure-preventing job acceptance.

Figure 6.7 CTC: Jobs with risk acceptance test and runtime-estimation based PDFs.

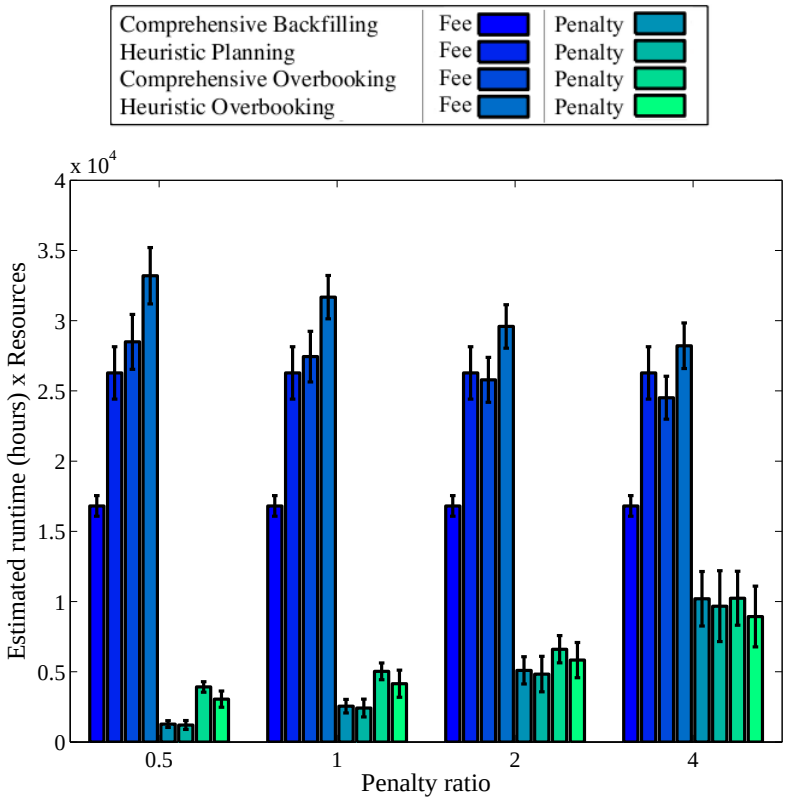


(a) The successful and failed jobs of the CTC trace.

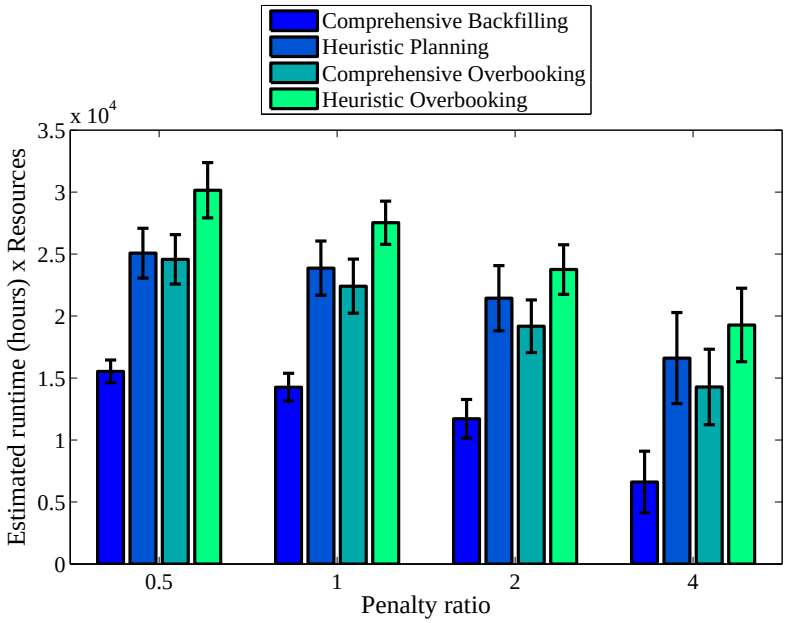


(b) The difference of successful and failed jobs.

Figure 6.8 CTC: Profit and penalties with risk acceptance test and runtime-estimation based PDFs.



(a) The CTC traces fees for the successful jobs and the penalties of the failed jobs.



(b) The CTC traces gain.

For the non-overbooking strategies, jobs only failed because resources broke down. The number of failed jobs did not change but for each penalty factor the total number of penalties doubled.

In addition, Figure 6.8(b) shows that even with an increasing penalty factor, the overbooking strategies had a higher income than the non-overbooking. The total gain for the heuristic overbooking decreased from 30,148 VCs to 19,279 VCs. More interestingly, while the total gain for the comprehensive overbooking decreased from 24,571 VCs to 14,278 VCs, the additional gain compared to the backfilling increased from 58% to 160%.

This indicates that the overbooking approaches are also applicable to commercial markets where fees and penalties are negotiated.

Results of the Resource based PDF simulations The last simulation for the CTC trace was based on the PDFs for the booked resources and the risk acceptance test. The input statistics for calculating the PoF for overbooking were those from Figure 6.2. Figures 6.9 and 6.10 contain the experimental results and Table 6.2 provides a summary.

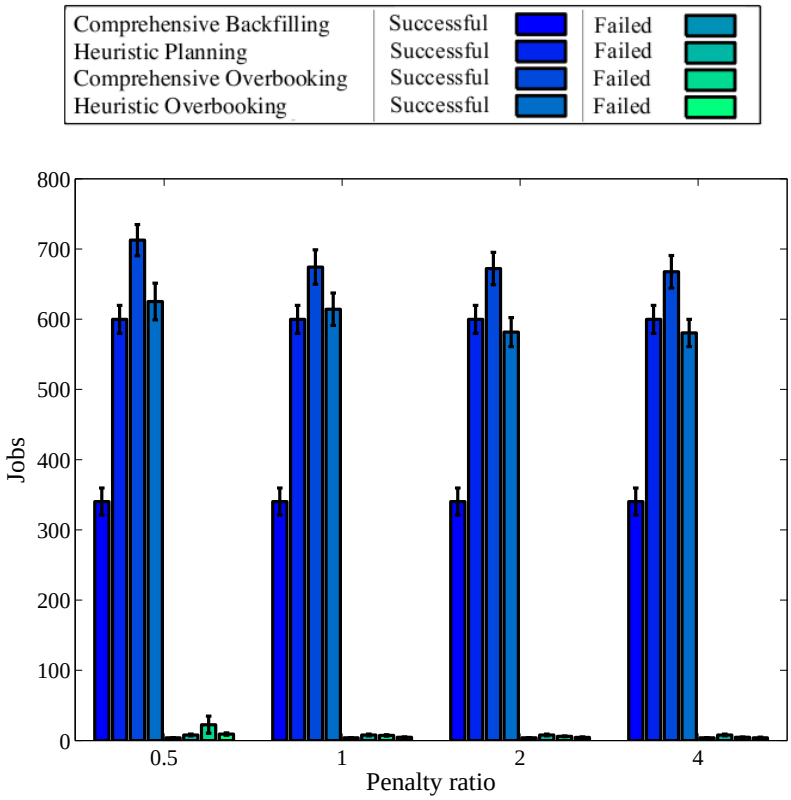
Figure 6.9(a) shows the successful and failed jobs. The results were similar to the runtime-estimation PDF function based simulation. The number of successful jobs decreased with a higher penalty ratio because only jobs with a lower PoF were accepted and accordingly the number of failed jobs was reduced. Consequently, it can be seen in Figure 6.9(b) that the ratio of successful jobs dropped slightly with a higher penalty ratio.

Figure 6.10(a) presents the fees and penalties of the simulation. Figure 6.10(b) shows the gain. While the penalties for the non-overbooking strategies doubled each time, the penalties for overbooking increased less. Subsequently in this trace even with a higher penalty, overbooking approaches were more competitive than strategies that did not overbook.

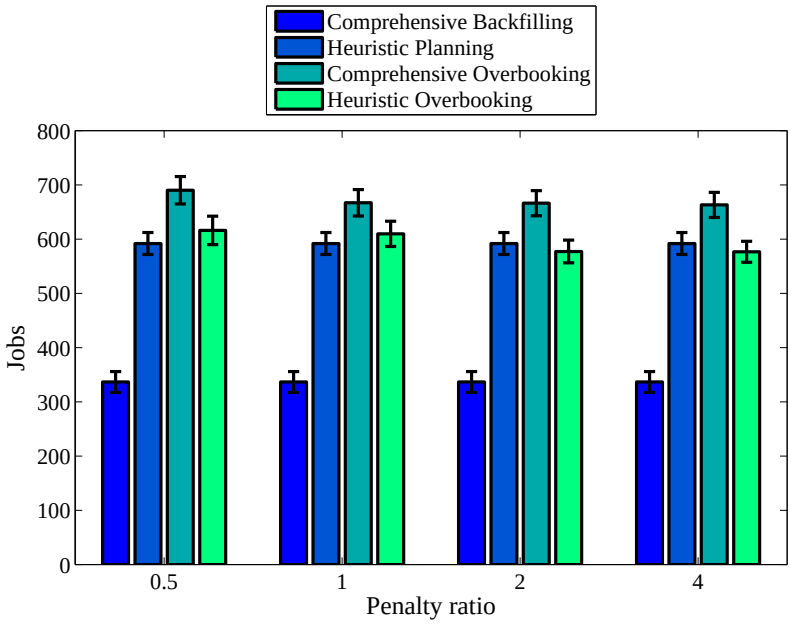
The effect remained, comparing the simulation results of the runtime-estimation based PDFs with the simulation results of the resource based PDFs. The total gain for the heuristic overbooking decreased from 26,776 VCs to 19,785 VCs. However, the additional gain compared to the heuristic planning increased from 7% to 19%. For the comprehensive overbooking, the gain decreased from 24,038 to 15,464 VCs. Here, the additional gain increased from 55% to 134% compared to the backfilling. The overbooking strategies got better results with more divergent runtime-estimation based PDFs.

The presentation of the results of the simulation with the CTC trace ends here. The remainder of this chapter shows the results of the remaining 5 simulations based on traces of the PWA. All simulations were conducted in the same way. However, the reader should have an idea by now how the simulations are evaluated. Therefore, the following results are described in a condensed manner.

Figure 6.9 CTC: Jobs with risk acceptance test and resource based PDFs.

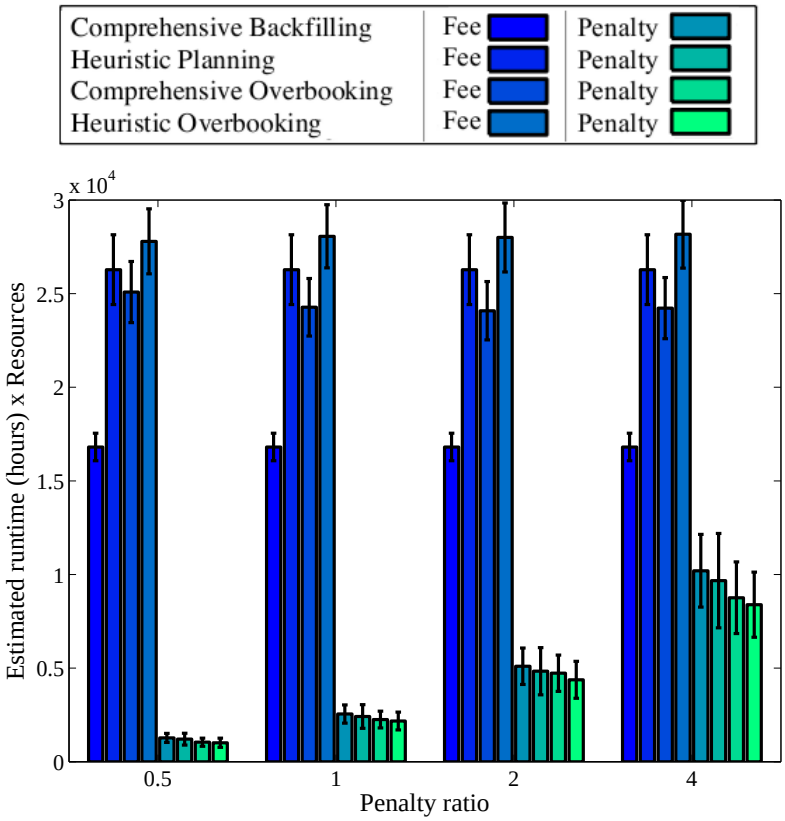


(a) The successful and failed jobs of the CTC trace.

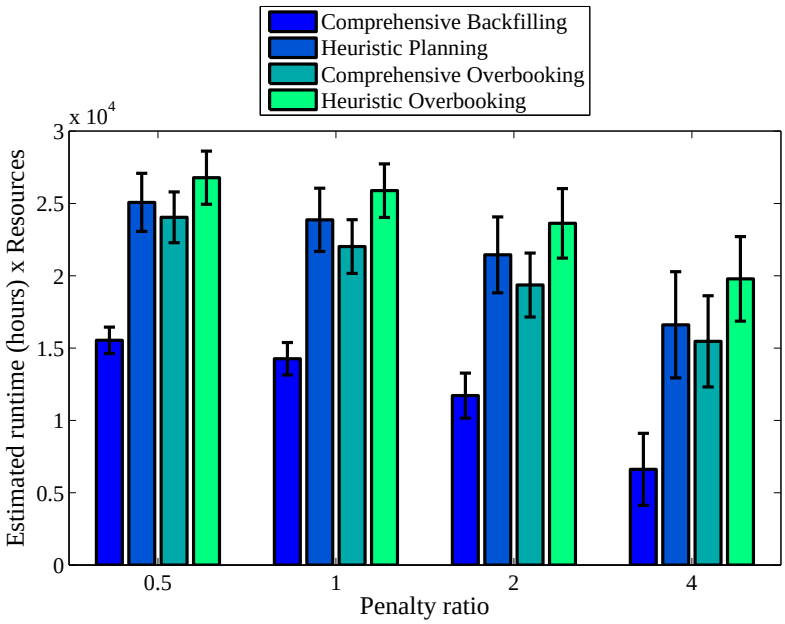


(b) The difference of successful and failed jobs.

Figure 6.10 CTC: Profit and penalties with risk acceptance test and resource based PDFs.



(a) The fees and penalties of the CTC trace.

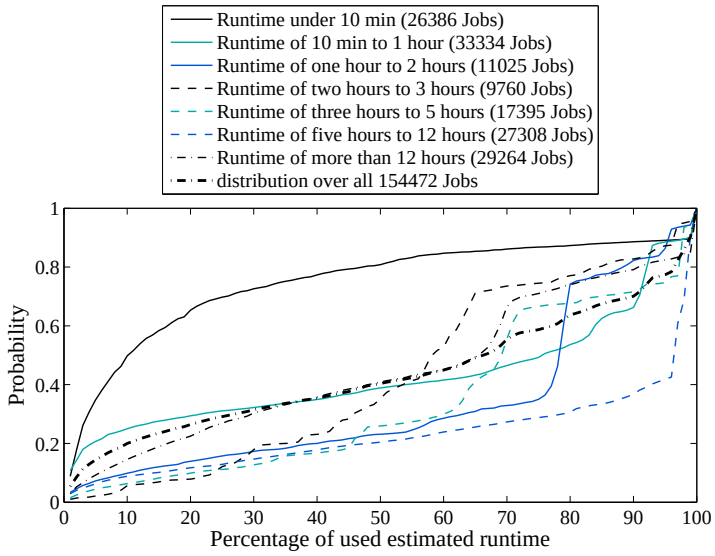


(b) The combined gain of the CTC trace.

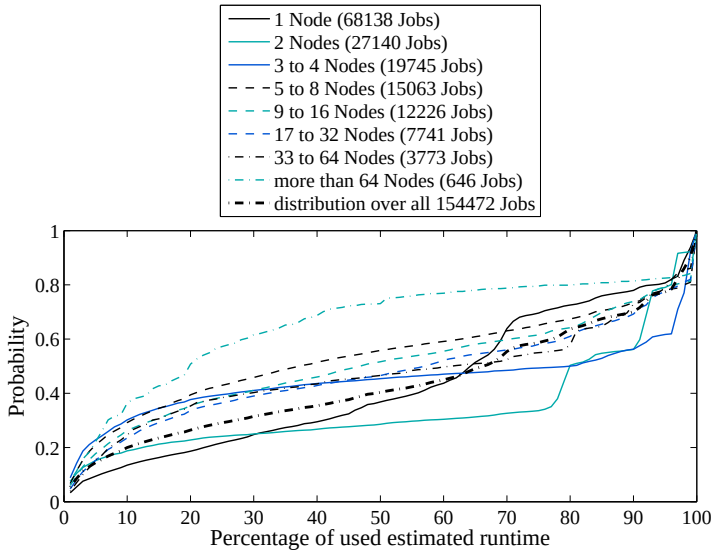
6.2 HPC2n

This section presents the results of a cluster trace of the High-Performance Computing Center North (HPC2n) in Sweden. The simulation had 20 test runs of 1000 jobs. The HPC2n cluster system had 240 resources. From the 202,876 jobs of the trace, the last 20,000 jobs were submitted to the system. The previous jobs were used to learn the statistical input; these were 154,472 entries.

Figure 6.11 HPC2n: The CDF functions.



(a) The runtime-estimation analysis of the HPC2n trace.



(b) The resource analysis of the HPC2n trace.

Approximately 27,000 jobs were not usable because the runtime-estimations or other important information was missing.

Figure 6.11(a) shows the CDFs calculated from the statistical runtime-estimation analysis and Figure 6.11(b) shows the CDFs according to the booked resources.

All in all, the runtime-estimation analysis reveals more distributed probability density functions. Figure 6.11(a) demonstrates that the jobs with a runtime-estimation of under ten minutes have the worst accuracy, while the jobs with an estimated-runtime of five to ten hours are estimated best.

The CDFs for the resource analysis are illustrated in Figure 6.11(b) and seem less distributed than the runtime-estimation CDFs. Jobs with more than 64 resources have the worst estimations and jobs with 2 nodes the best. Of course, the CDF distribution among all jobs is the same for both analyses because it had the same input. In this and the following traces, the simulation results are shown in a more condensed view. The focus is on the combined gain.

6.2.1 PoF Acceptance Test

The evaluation starts with the PoF based acceptance test. Table 6.3 summarizes the maximum values for the runtime-estimation and resource statistics and shows the surplus of the overbooking strategies in percent.

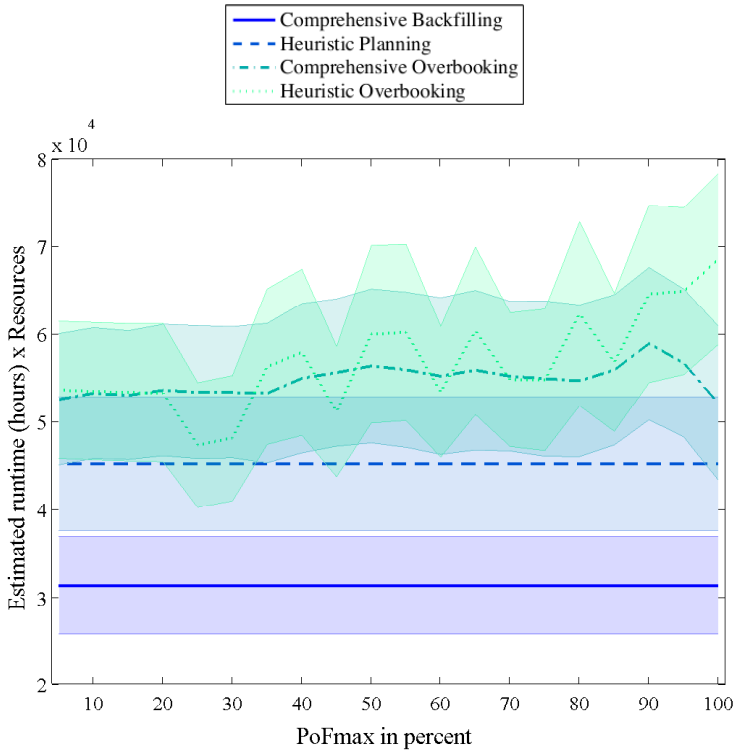
Table 6.3: *The PoF max gain with HPC2n trace.*

PoF simulation	runt. est.	plus	PoF_{max}	resources	plus	PoF_{max}
Comp. backfilling	31,250			31,250		
Heuristic planning	45,150			45,150		
Comp. overbooking	58,860	88%	0.9	60,750	94%	0.9
Heur. overbooking	68,570	52%	1	68,570	52%	1

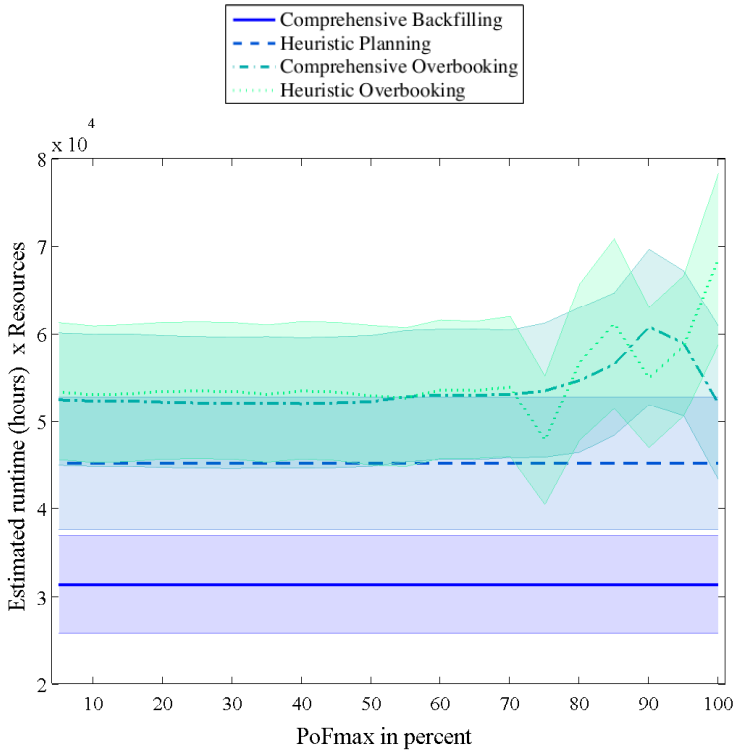
Figures 6.12(a) and 6.12(b) show that the results of the PoF calculation with the underlying PDF functions are tending to PoF overestimation. With a low threshold until PoF_{max} of 0.3 for the runtime-estimation based PDFs and 0.6 for the resource based PDFs no additional runtime was sold. Starting above these thresholds, the curves show that it was possible to successfully overbook additional jobs. For the heuristic, with a PoF_{max} of 1 it was possible to successfully get the highest gain.

Another indication that the underlying statistics were not very accurate was shown by the fluctuating results. For some simulation runs, with the given threshold some very big and dominating jobs were accepted and were successful or failed and for the next higher threshold, they were not accepted. This behavior underlines the uncertainty of the real average value being printed in the line plots, which is illustrated by the 95% confidence intervals. The simulation only had 20 runs due to the limited amount of jobs in the traces. However, the more careful comprehensive backfilling avoids the fluctuating results by not accepting big jobs. The approach cannot move jobs on the resources; therefore, the plan is more scattered and fewer big gaps for big jobs exist.

Figure 6.12 HPC2n: The gain for the PoF acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

Table 6.4: *The Risk simulation results with HPC2n trace.*

Risk simulation	Penalty Ratio			
Runtime based statistics	0.5	1	2	4
Comprehensive backfilling	34,024	31,246	25,691	14,581
Heuristic planning	47,756	45,149	39,936	29,511
Comprehensive overbooking	60,318	55,789	52,605	48,664
Additional comp. profit	77%	79%	105%	234%
Heuristic overbooking	64,716	59,486	51,189	47,416
Additional heuristic profit	36%	32%	28%	61%
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	34,024	31,246	25,691	14,581
Heuristic planning	47,756	45,149	39,936	29,511
Comprehensive overbooking	52,910	52,223	50,988	48,329
Additional comp. profit	56%	67%	99%	231%
Heuristic overbooking	54,085	53,340	51,622	47,129
Additional heuristic profit	13%	18%	29%	60%

For the heuristic, the best results were achieved with a very high PoF_{max} of 1. It allowed an income of 68,570 VCs. For the more stable comprehensive approach, the best results were 58,860 and 60,750 VCs with a PoF_{max} of 0.9. The comprehensive approach could increase the gain of backfilling by about 88% to 94% and the heuristic overbooking by 52%.

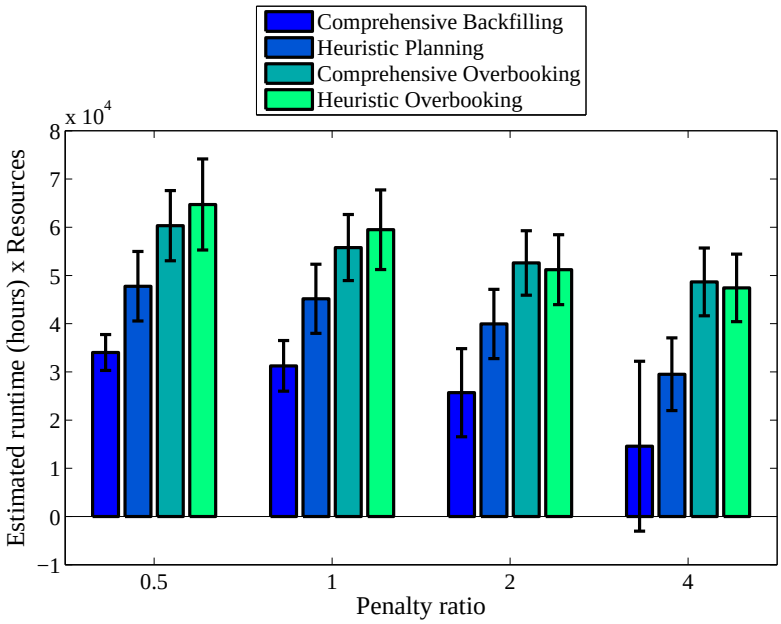
For the comprehensive approach, the resource density functions provided a slightly higher maximum result and fluctuated less as when used by heuristic approach. Therefore, the resource based PDFs seemed to be more appropriate than the runtime-estimation based PDFs. Instead, the runtime-estimation probability density functions were dangerous, especially for the heuristic overbooking. To be usable, they will have to be adapted with more recent monitoring information.

It is important to point out that the average results of the simulation runs were overlapping. This is shown by the fluctuating results of the heuristic overbooking and the huge and overlapping 95% confidence intervals of the strategies. This means that with the given PDFs, the jobs of the traces were so divergent that the strategies could beat one another. Thus, this simulation showed that it is important to adapt the statistical input to the current monitoring information to be able to cope with changing behaviors.

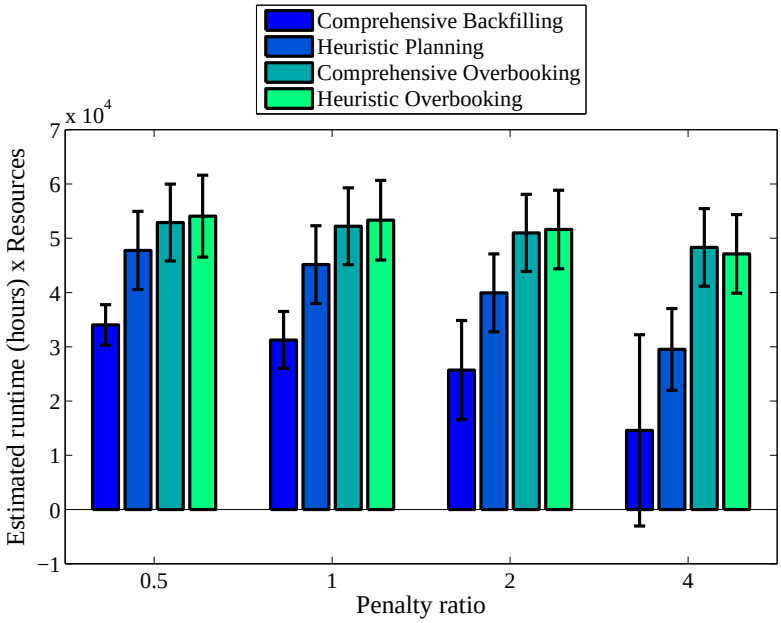
6.2.2 Risk Acceptance Test

Here, the risk acceptance test based simulation is evaluated, and the results of the 20 test runs according to the penalty ratios are shown in Table 6.4. The table shows the average results and the surplus of the overbooking strategies in percent.

Figure 6.13 HPC2n: The gain for the risk acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

Figure 6.13 shows that the runtime-estimation PDF based overbooking had the best results for the low penalty ratio of 0.5. For all scheduling strategies, the results shrank when the penalty ratio increased. However compared to the non-overbooking strategies, the results of the overbooking strategies shrank less. The overbooking strategies were relatively stable, especially with the runtime-estimation probability density functions. One can see this in Table 6.4 with a look on the percentage of profit. The heuristic overbooking increased from 36% to 61% with the runtime-estimation functions and from 13% to 60% for the resource based PDFs.

The comprehensive overbooking benefited a lot more from the penalty ratios. The profit increased from 77% to 234% with the runtime-estimation based CDFs and from 56% to 231% for the resource analysis based overbooking.

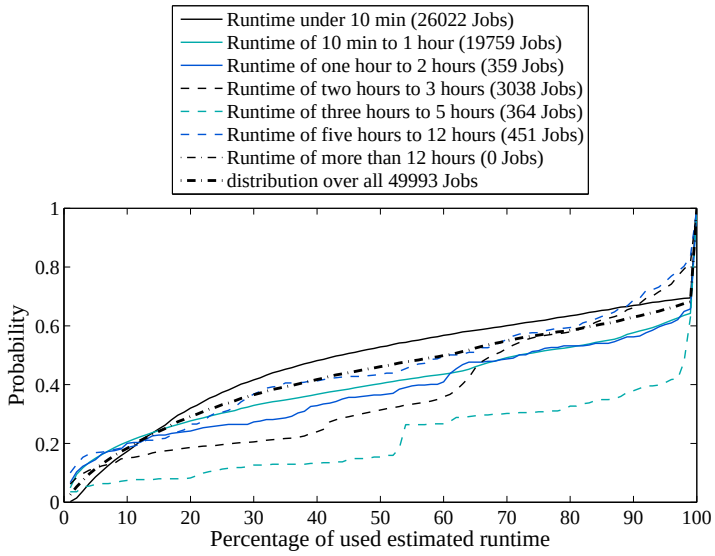
Accepting fewer jobs due to higher risk increased the profit of the overbooking strategies compared to the non-overbooking counterparts. This indicates that the strategies were too offensive. Either the input functions have to be adapted or the security factor should be increased to get more reliable and better results.

At this point, the presentation of the results of the HPC2n trace is finished. The four remaining simulations with traces of the PWA are presented in the same format.

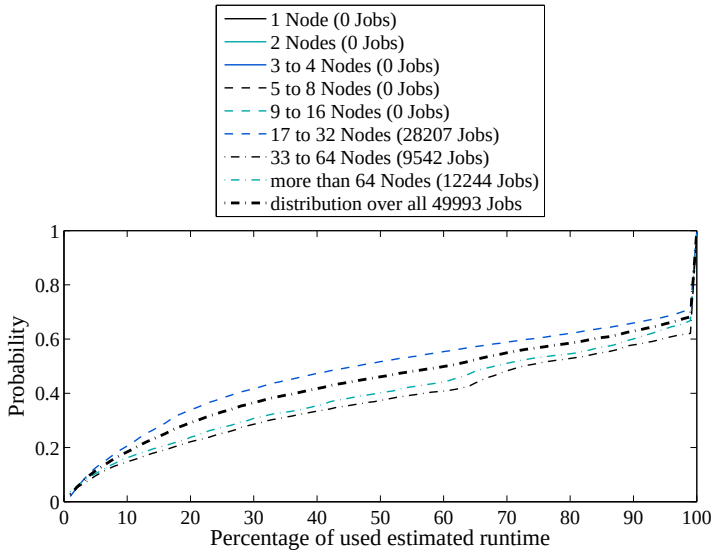
6.3 LANL

This section shows the results of the simulation with 20 test runs of 1000 jobs for the Los Alamos National Lab (LANL) trace of the PWA. The LANL cluster system had 1024 resources. From the 122,060 jobs of the trace, the last 20,000 were submitted to the system. The 49,993 previous jobs were used to learn the statistical input. The remaining about 50,000 jobs were not usable.

Figure 6.14 LANL: The CDF functions.



(a) The runtime-estimation analysis of the LANL trace.



(b) The resource analysis of the LANL trace.

Table 6.5: *The PoF max gain with LANL trace.*

PoF simulation	runt. est.	plus	PoF_{max}	resources	plus	PoF_{max}
Comp. backfilling	39,070			39,070		
Heuristic planning	39,180			39,180		
Comp. overbooking	52,460	34%	0.05	52,970	36%	0.05
Heur. overbooking	56,690	45%	0.1	59,400	52%	0.05

Figure 6.14(a) shows the cumulative density functions calculated from the statistical runtime-estimation analysis of the workload trace. Figure 6.14(b) illustrates the cumulative density functions according to the booked resources. The best runtime-estimation frame is the frame for jobs between three and 5 hours. The jobs with a runtime-estimation from two to 3 hours are also estimated well. The LANL trace contained no jobs with a duration of more than 12 hours. While the runtime-estimation analysis of the LANL trace shows divergent CDFs, the resource analysis shows that the cluster only had jobs that used more than 32 resources, and the resulting CDFs of this analysis are very similar. Therefore, the runtime-estimation analysis seems to be more beneficial for overbooking.

6.3.1 PoF Acceptance Test

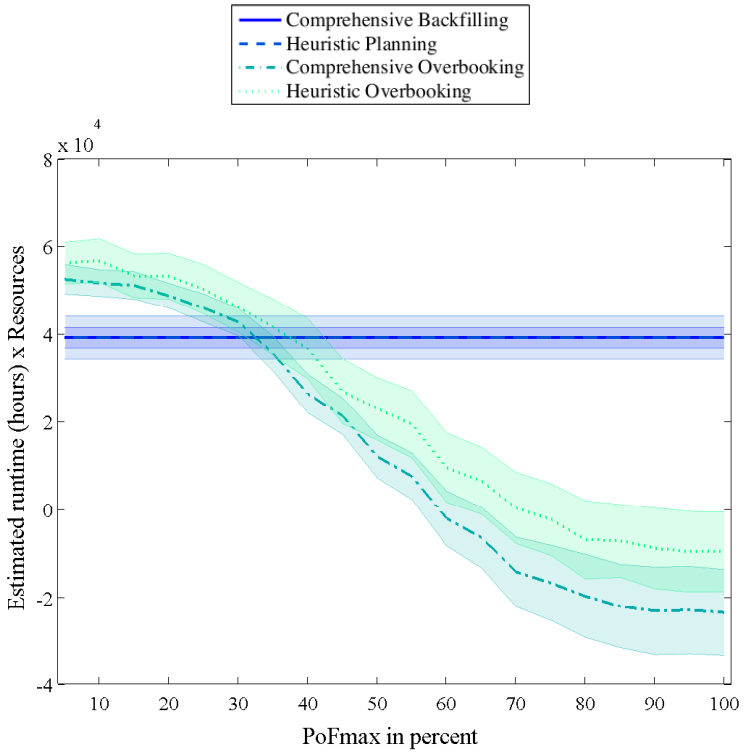
The results of the PoF based acceptance tests are illustrated in Figures 6.15(a) and 6.15(b). Table 6.5 summarizes the maximum values for the resource and runtime-estimation statistics and shows the surplus of the overbooking strategies in percent.

Figure 6.15 shows interesting behaviors. Backfilling and heuristic planning had the same results with about 39,000 VCs. If comparing the two overbooking strategies, both lines have a very similar shape. Both overbooking approaches had the maximum gain at a PoF_{max} of 0.05 and fell at a PoF_{max} between 0.3 and 0.45 through the lines of the non-overbooking strategies.

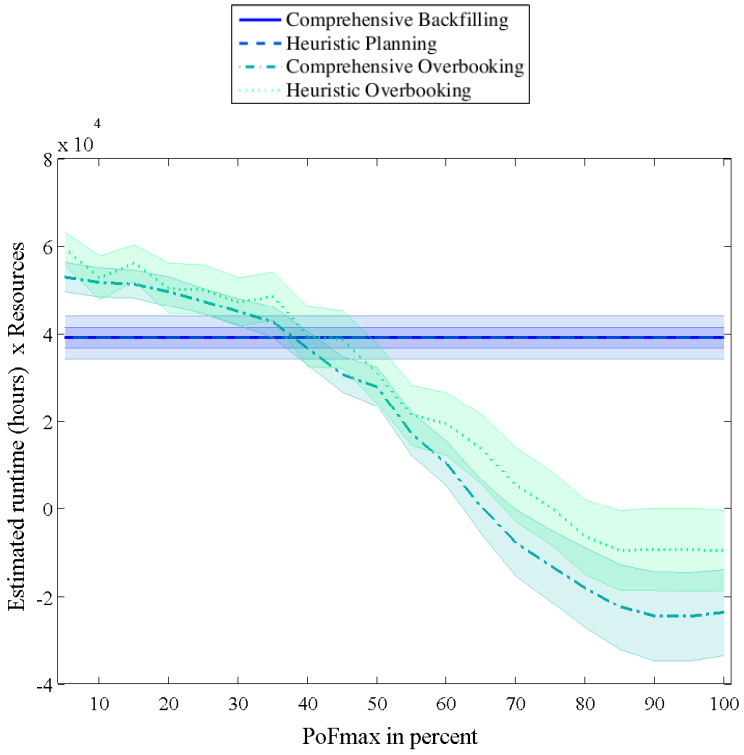
The shape of the curves indicates that the PoF calculations with the underlying functions were tending to failure underestimation. With a low threshold until PoF_{max} of 0.05 additional runtime was sold but the curves show that with a higher threshold the gain nearly directly decreased.

In both cases, the heuristic had a higher profit of 56,690 VCs ($PoF_{\text{max}} = 0.1$) with the runtime-estimation analysis and 59,400 VCs ($PoF_{\text{max}} = 0.05$) with the resource analysis. The comprehensive approach had a result of 52,460 VCs (time functions, $PoF_{\text{max}} = 0.05$) and 52,970 VCs (resource functions, $PoF_{\text{max}} = 0.05$). The results of the simulation with the two analyses were very similar. The comprehensive approach earned 34% and 36% additional profit. The heuristic received an additional gain between 45% and 52%. While the non-overbooking strategies were very similar, the heuristic interestingly profited more from the overbooking. This implies that the heuristic is more flexible. While the heuristic planning was not able to profit from the flexibility, the heuristic overbooking was.

Figure 6.15 LANL: The gain for the PoF acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

Table 6.6: *The Risk simulation results with LANL trace.*

Risk simulation	Penalty Ratio			
Runtime based statistics	0.5	1	2	4
Comprehensive backfilling	43,299	39,070	30,614	13,701
Heuristic planning	50,627	39,180	16,273	-29,532
Comprehensive overbooking	47,325	48,672	42,922	27,354
Additional comp. profit	9%	25%	40%	100%
Heuristic overbooking	45,863	40,557	35,244	7,633
Additional heuristic profit	-9%	3%	116%	-125%
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	43,299	39,070	30,614	13,701
Heuristic planning	50,627	39,180	16,273	-29,532
Comprehensive overbooking	52,603	49,628	43,032	28,298
Additional comp. profit	21%	27%	41%	107%
Heuristic overbooking	49,633	47,837	29,965	02,342
Additional heuristic profit	-2%	22%	84%	-107%

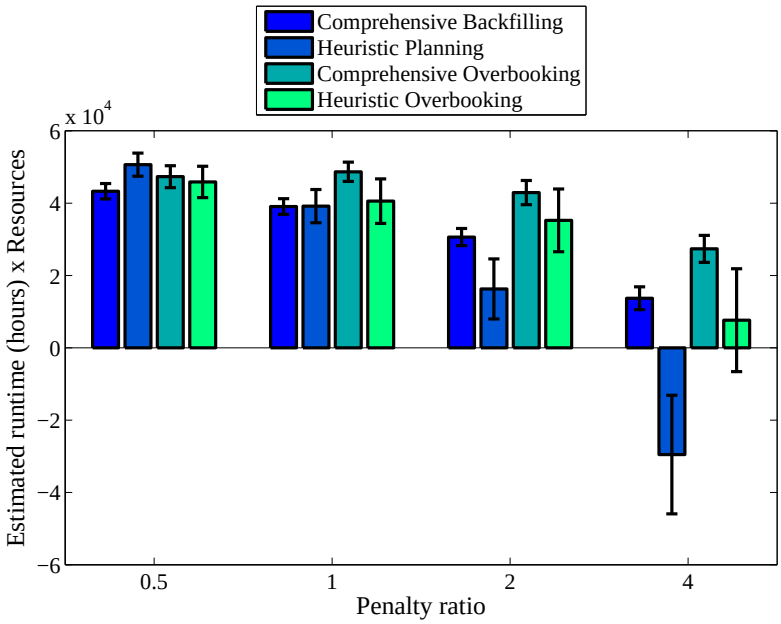
The assumption that due to the spreading of the distribution the runtime-estimation analysis would be better than the resource analysis did not hold. The results of the simulation based on the different input statistics were similar. The comprehensive approach was a little better when using the resource density functions and the heuristic was better with the runtime-estimation based PDFs.

The reason for the similar results is that the estimations change their accuracy over time. This is the case when a project is finished or new users begin to use the cluster. Then, different jobs with new applications occur. This simulation shows that it is important to adapt the statistical input to the current monitoring information to be able to cope with changing estimation accuracy.

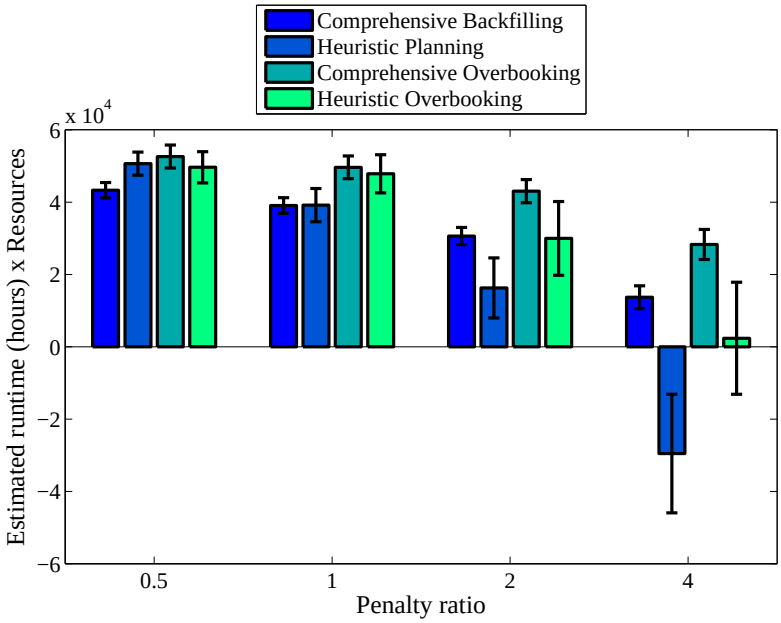
6.3.2 Risk Acceptance Test

The risk based acceptance test is depicted in Figures 6.16(a) and 6.16(b). Table 6.6 summarizes the results of the risk estimation simulation. The table shows the average results of the 20 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent.

Figure 6.16 LANL: The gain for the risk acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

The non-overbooking approaches had the same results as in the PoF acceptance test with a penalty ratio of 1. However, the non-overbooking heuristic had a better result with a penalty ratio of 0.5 (with a penalty ratio of 0.5, the profit is 50,627 VCs) and declines with a penalty ratio of 2 (16,273 VCs). With a penalty ratio of 4, the heuristic planning had a negative income (-29,532 VCs). This means that the heuristic lost a lot more jobs than the backfilling.

Here, the overbooking approaches profited from the policy that overbooked jobs could be started with less runtime than estimated and, thus, could be restarted after a node crash. Therefore, the loss due to higher penalties was smaller than for the non-overbooking strategies. In this simulation, the comprehensive overbooking strategy had a higher profit than the heuristic. However with a higher risk due to a higher penalty ratio, the overbooking strategies accepted jobs with a lower calculated PoF only, and the additional profit increased.

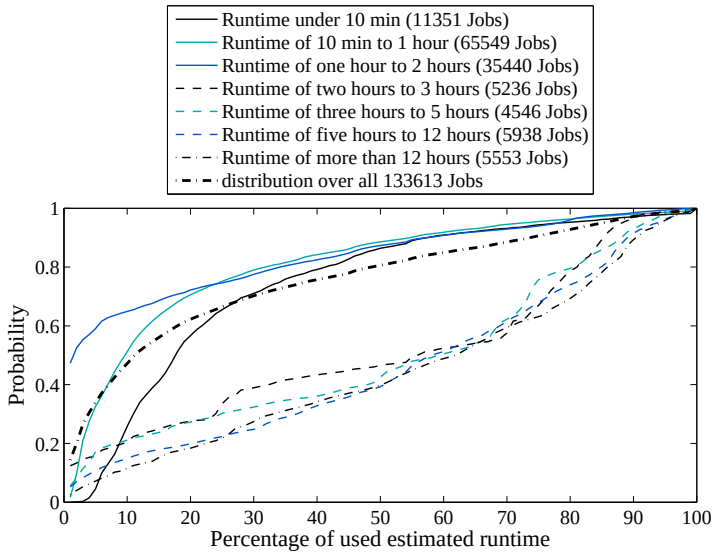
The negative values of the additional profit, with a penalty ratio of 4, for the heuristic simply shows that the result of the heuristic planning was negative. However, it remains positive for the overbooking approach. The heuristic lost too many SLAs due to node outages, which made the approach lose more money than it gained. The heuristic fills the resources to the maximum, even if nodes crashed. This could cause a great loss of money for the heuristic.

The presentation of the LANL based simulation ends here, and the chapter continues with three traces from the San-Diego Supercomputer Center SDSC.

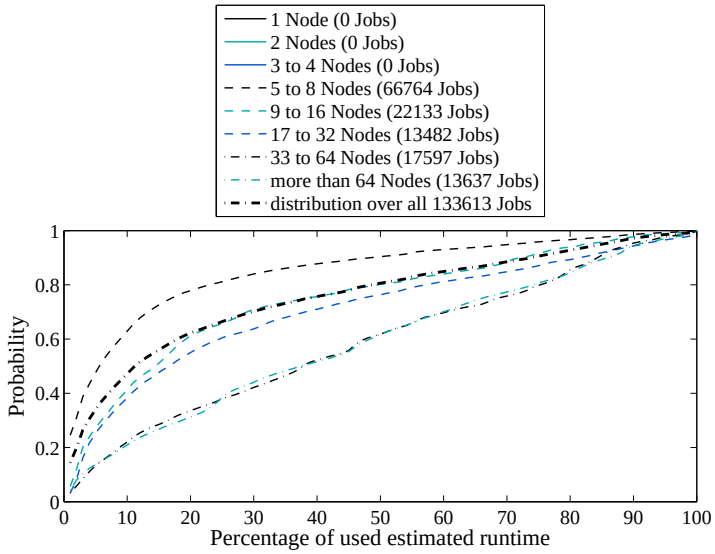
6.4 SDSC-BLUE

This section shows the simulation results for the San-Diego Supercomputer Center Blue Horizon (SDSC-BLUE) trace with 20 test runs of 1000 jobs. The SDSC-BLUE cluster system had 1,152 resources. From the 243,314 jobs of the trace, the last 20,000 jobs were submitted to the simulation. The previous jobs were used to learn the statistical input; these were 133,613 entries. The remaining about 90,000 jobs were not usable.

Figure 6.17 SDSC-BLUE: The CDF functions.



(a) The runtime-estimation analysis of the SDSC-BLUE trace.



(b) The resource analysis of the SDSC-BLUE trace.

Table 6.7: *The PoF max gain with SDSC-BLUE trace.*

PoF simulation	runt. est.	plus	$PoF_{\max}$	resources	plus	$PoF_{\max}$
Comp. backfilling	52,420			52,420		
Heuristic planning	70,390			70,390		
Comp. overbooking	81,810	56%	0.6	82,940	58%	0.05
Heur. overbooking	96,620	37%	0.75	92,260	31%	0.85

Figure 6.17(a) presents the cumulative density functions calculated with the statistical analysis of the SDSC-BLUE trace. One can see two different groups of CDFs. The shorter jobs (up to 2 hours) are estimated worse than the jobs from two to 12 hours.

Figure 6.17(b) plots the cumulative density functions according to the booked resources. Here, one can see that only jobs with 8 or more resources were submitted. The jobs with more than 32 resources were better estimated than the jobs with 8 to 32 resources.

If the simulation's jobs would have the same behavior, the results of the overbooking should profit from the more distributed CDFs of the runtime-estimation analysis. However, the CDFs of the resource analysis also displayed different behaviors of the resource classes even if three of the classes remained unused. Therefore, overbooking might also profit from the results of the resource analysis.

6.4.1 PoF Acceptance Test

Table 6.7 shows the maximum values of the simulation with the PoF based acceptance test. The table shows the average results of the 20 test runs for the runtime-estimation and resource statistics and the surplus of the overbooking strategies in percent.

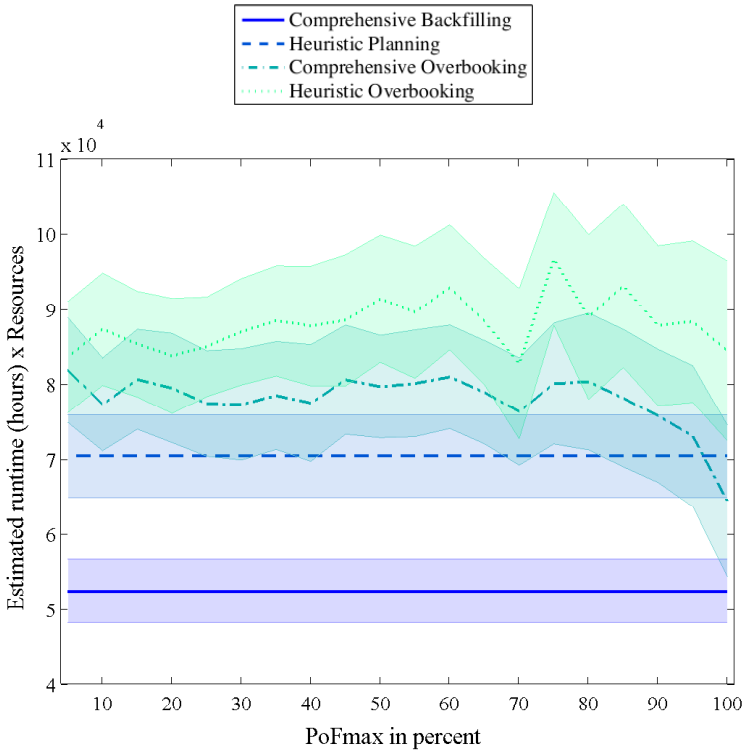
From the non-overbooking strategies, the heuristic earned a lot more with about 70,000 VCs average gain than the backfilling strategy with about 52,000 VCs.

The results of the overbooking strategies were similar. The heuristic had, with 96,620 VCs for the runtime-estimation based PDFs and 92,260 VCs for the resource functions, a better result than the comprehensive approach with 81,810 and 82,940 VCs, respectively. The heuristic seemed to profit more from the runtime-estimation based PDFs, while the comprehensive overbooking worked better with the resource based PDFs.

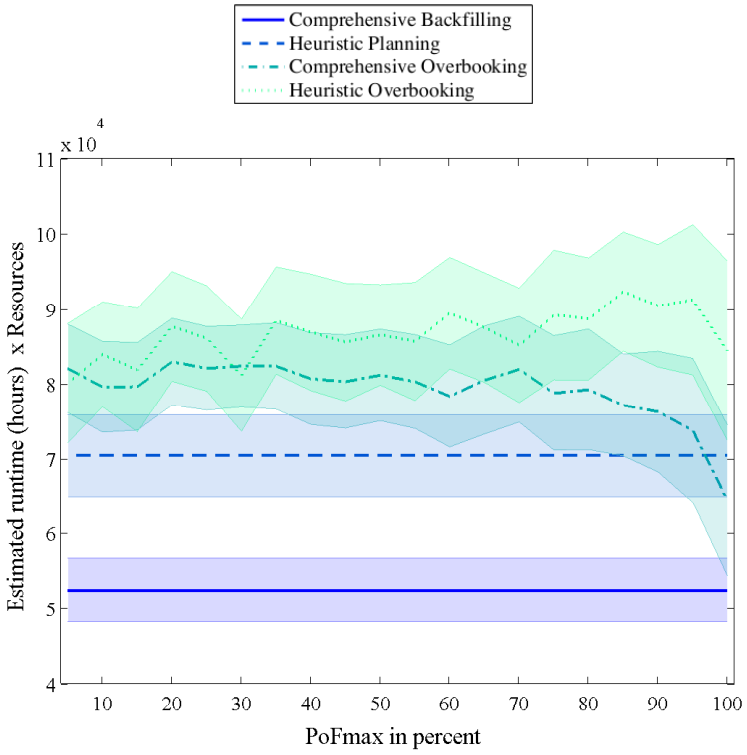
Like the HPC2n trace, the heuristic had a fluctuating result. Especially the difference between a $PoF_{\max}$ of 0.6 and 0.9 for the runtime-estimation probability density function based simulation was enormous. However, the 95% confidence intervals indicate that this might happen. One can see that many jobs had over 64 resources. This means that a job with a long runtime-estimation was worth so many VCs that it could influence the whole result of the test run.

Beneath the fluctuating curves, their shape indicates that the underlying functions were suitable. With a low threshold, more and more additionally accepted runtime

Figure 6.18 SDSC-BLUE: The gain for the PoF acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

Table 6.8: *The Risk simulation results with SDSC-BLUE trace.*

Risk simulation	Penalty Ratio			
Runtime based statistics	0.5	1	2	4
Comprehensive backfilling	57,962	52,416	41,325	19,142
Heuristic planning	77,695	70,387	55,772	26,541
Comprehensive overbooking	85,460	79,439	70,629	58,177
Additional comp. profit	47%	51%	71%	204%
Heuristic overbooking	95,475	89,787	73,210	58,411
Additional heuristic profit	23%	28%	31%	120%
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	57,962	52,416	41,325	19,142
Heuristic planning	77,695	70,387	55,772	26,541
Comprehensive overbooking	86,498	82,937	70,541	59,896
Additional comp. profit	49%	58%	71%	213%
Heuristic overbooking	91,319	85,556	79,276	44,648
Additional heuristic profit	18%	22%	42%	68%

was successfully sold and if the accepted PoF became too high more and more additionally accepted jobs failed.

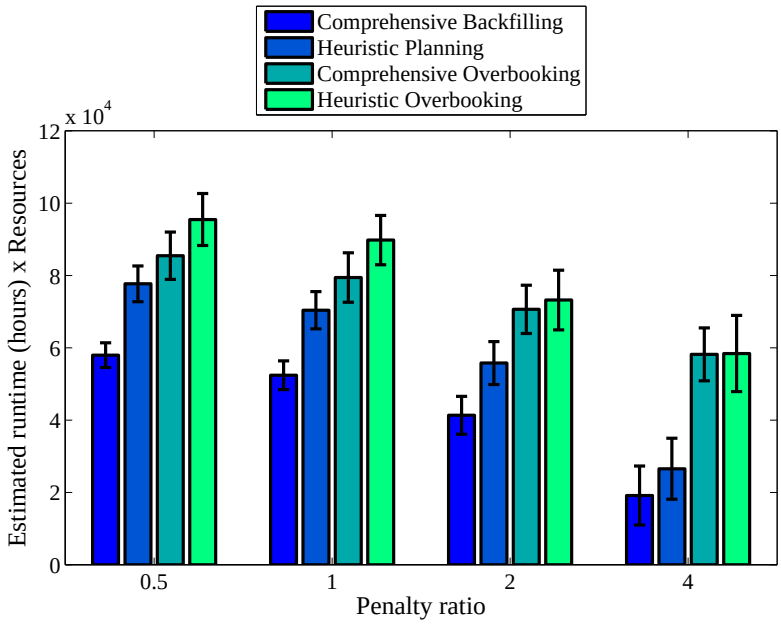
The heuristic produced the best result with a high PoF_{max} of about 0.75 while the comprehensive approach had nearly the same gain for a broad PoF_{max} range from 0.05 to 0.8. The comprehensive overbooking had a smaller maximum gain. However, according to its underlying strategy, the additional overbooking-profit was with 56% and 58% for the comprehensive overbooking higher than the additional profit of the heuristic. It only had 37% and 31% of additional profit.

The two underlying statistical analyses had similar results. The heuristic profited a little bit more from the runtime-estimation based PDFs, while the comprehensive approach profited more from the resource analysis.

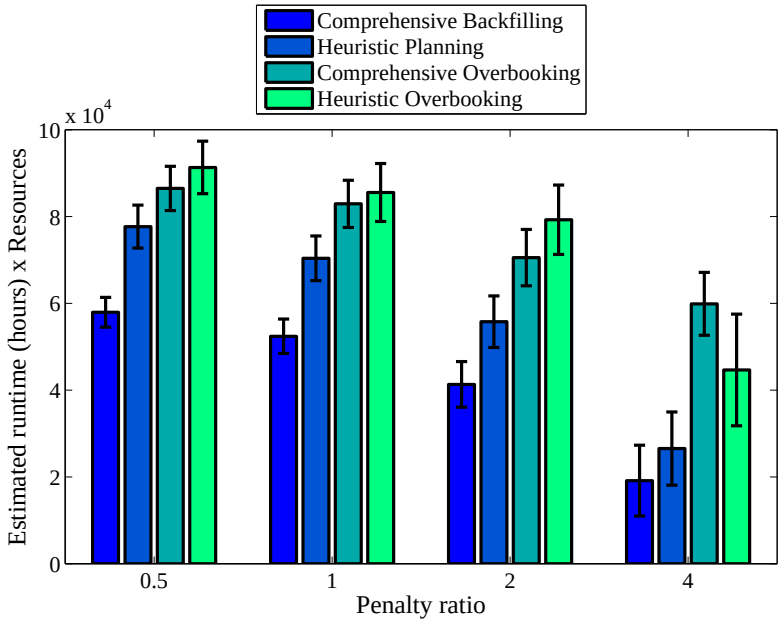
6.4.2 Risk Acceptance Test

The simulation with the risk based acceptance test is summarized in Table 6.8. The table shows the average results of the 20 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent.

Figure 6.19 SDSC-BLUE: The gain for the risk acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

The outcome of the risk based simulation with the SDSC-BLUE shows that both results of the non-overbooking strategies decreased linearly with an increasing penalty ratio.

From the overbooking algorithms, the heuristic suffered more on the higher penalty ratios. The heuristic's results were at the beginning better than the comprehensive ones and at the end on or below the level of the comprehensive overbooking.

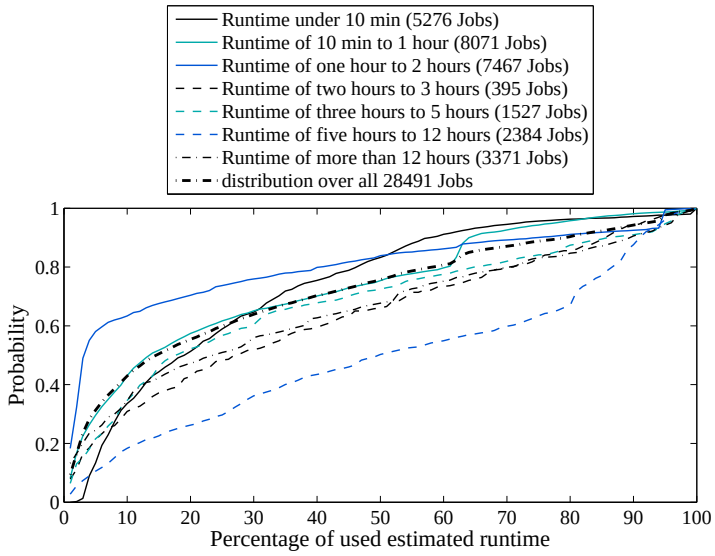
For both overbooking strategies, the additional gain increased with the penalty ratio.

The heuristic profited more from the runtime-estimation CDF based simulation with a gain between 23% and 120%. The comprehensive approach profited more from the resource analysis based simulation. In the simulation, the additional profit increased from 49% to 213%. The presentation of the SDSC-BLUE based simulation ends here, and the chapter continues with the next traces from the San-Diego Supercomputer Center.

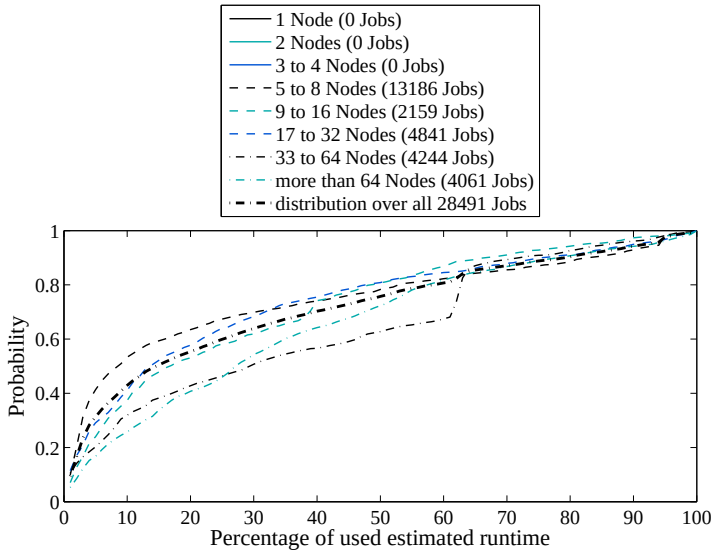
6.5 SDSC-DataStar

In this section, the results of the San Diego Supercomputer Center DataStar (SDSC-DataStar) trace simulation with 20 test runs of 1000 jobs are presented. The SDSC-DataStar cluster system had 1,664 resources. From the 96,089 jobs of the trace, the last 20,000 jobs were submitted to the simulation. The previous jobs were used to learn the statistical input; these were 28,491 entries. The remaining 48,000 jobs were not usable.

Figure 6.20 SDSC-DataStar: The CDF functions.



(a) The runtime-estimation analysis of the SDSC-DataStar trace.



(b) The resource analysis of the SDSC-DataStar trace.

Table 6.9: *The PoF max gain with SDSC-DataStar trace.*

PoF simulation	runt. est.	plus	PoF_{max}	resources	plus	PoF_{max}
Comp. backfilling	64,360			64,360		
Heuristic planning	99,290			99,210		
Comp. overbooking	114,490	78%	0.4	110,150	71%	0.75
Heur. overbooking	134,110	35%	0.6	130,680	32%	0.7

Figure 6.20(a) shows the cumulative density functions calculated with the statistical runtime-analysis of the workload trace. Figure 6.20(b) shows the cumulative density functions according to the booked resources.

For the runtime-estimation analysis, only two curves differed from the others. The one to 2 hours estimates were worse and the five to 12 hours estimates were better than the average. The SDSC-DataStar had jobs with a minimum resource request size of 8 resources. Therefore, only 5 different resource classes exist. The shapes of the resource based CDFs are similar. Only the jobs with 33 to 64 nodes were estimated a little better. Therefore, the resource analysis does not seem to be very beneficial for this simulation.

6.5.1 PoF Acceptance Test

Table 6.9 summarizes the maximum gain of the PoF based acceptance test. The table shows the average results of the 20 test runs for the runtime-estimation and resource statistics and the surplus of the overbooking strategies in percent.

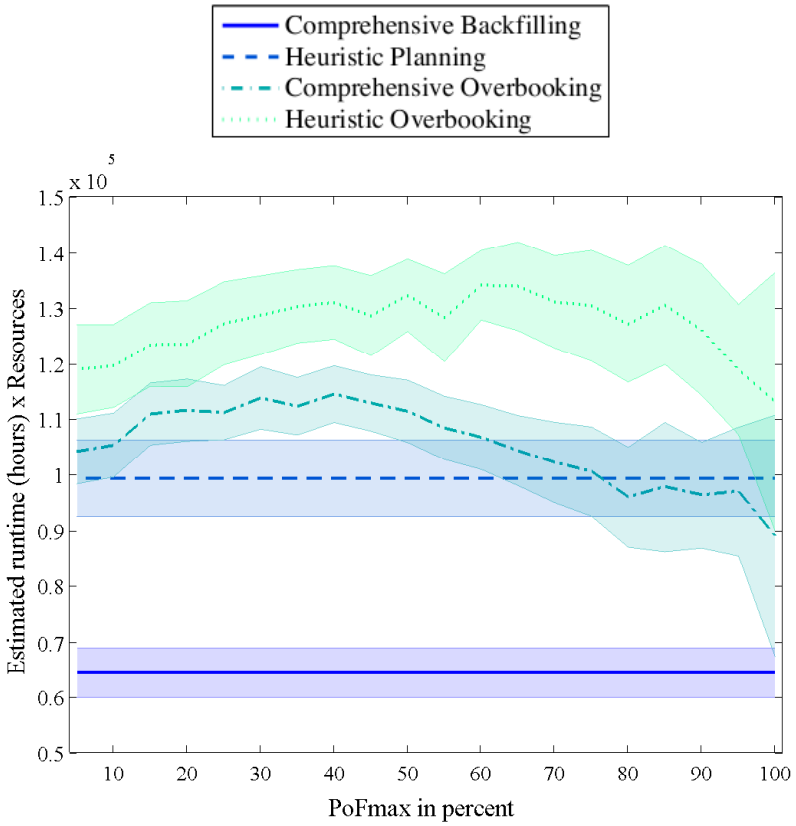
The non-overbooking strategies had an average income of 64,360 VCs for the backfilling and 99,290 VCs for the planning. This was 54% more income for the heuristic planning. For the overbooking strategies, the resulting curves show that the heuristic made a higher profit than the comprehensive approach.

For the runtime-estimation based PDFs, the heuristic approach had 134,110 VCs with aPoF_{max} of 0.6, about 20,000 VCs more profit than the comprehensive approach. Both strategies profited from an increasing PoF_{max} threshold. The highest gain of the comprehensive approach was achieved at a PoF_{max} between 0.3 and 0.4. The heuristic seemed to be more conservative and had increasing results until a PoF_{max} of 0.6.

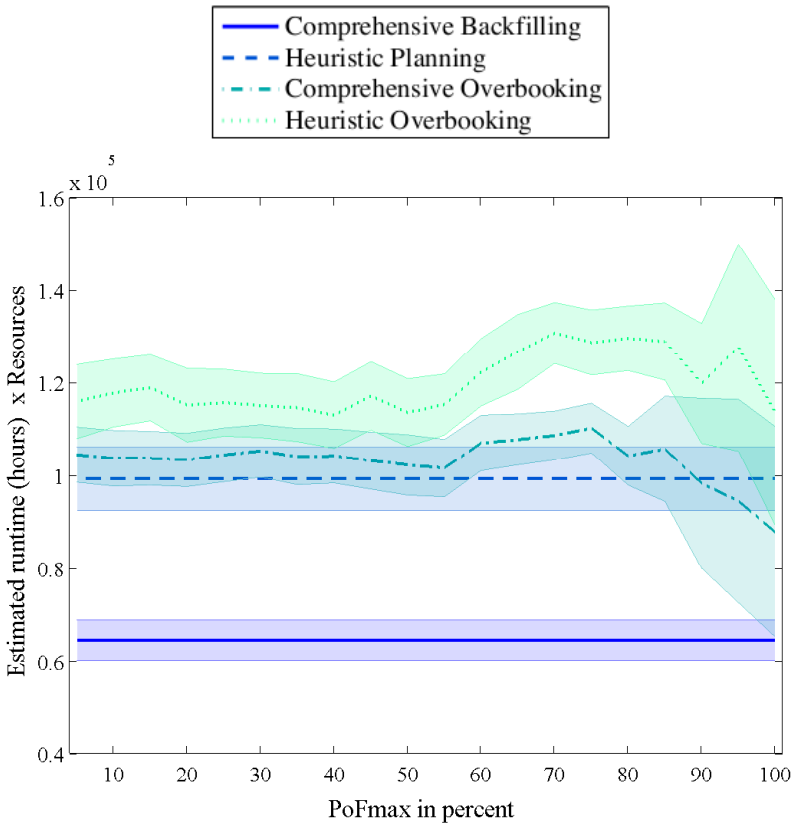
For the resource based PDF based simulation, both plots for the overbooking approach were nearly linear in a PoF_{max} range between 0.05 and 0.55 and then began to increase. The comprehensive approach had its maximum profit at a $PoF_{\text{max}} = 0.75$ and the heuristic had its maximum at a $PoF_{\text{max}} = 0.7$.

The maximum gain of the heuristic lay at 134,110 VCs for the runtime-estimation analysis and 130,680 VCs for the resource analysis. For the comprehensive approach, it was 114,490 and 110,150 VCs, respectively. The heuristic overbooking had about 20% more income.

Figure 6.21 SDSC-DataStar: The gain for the PoF acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

The profit compared to the non-overbooking strategies was for the comprehensive overbooking higher with 78% and 71%, according to 35% and 32% for the heuristic overbooking. This shows that the overbooking of the comprehensive approach can compensate the bad performance of the simple backfilling strategy.

The shape of the runtime-estimation density function indicates that the underlying function was accurate. With a low threshold, more and more additionally accepted runtime was successfully sold and if the accepted PoF became too high more and more additionally accepted jobs failed. The resource density function was not so supportive. In the beginning, no additional runtime was sold and above a short peak, the gain fell. The more accurate runtime-estimation based PDFs also allow a higher profit than the resource density functions. This corresponds with the assumptions when comparing the functions in the beginning of this section.

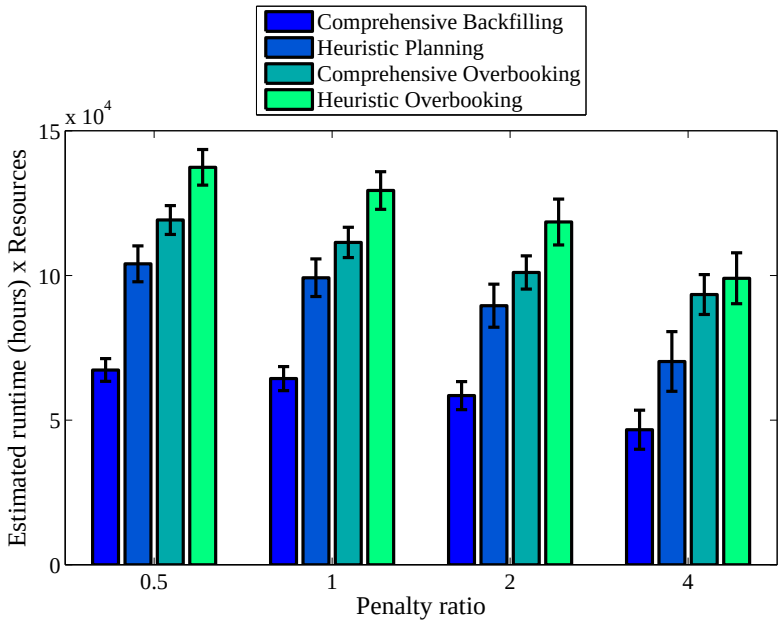
6.5.2 Risk Acceptance Test

Table 6.10 summarizes the results of the risk based acceptance test and the SDSC-DataStar simulation. The table shows the average results in VCs for the penalty ratios and the surplus of the overbooking strategies in percent.

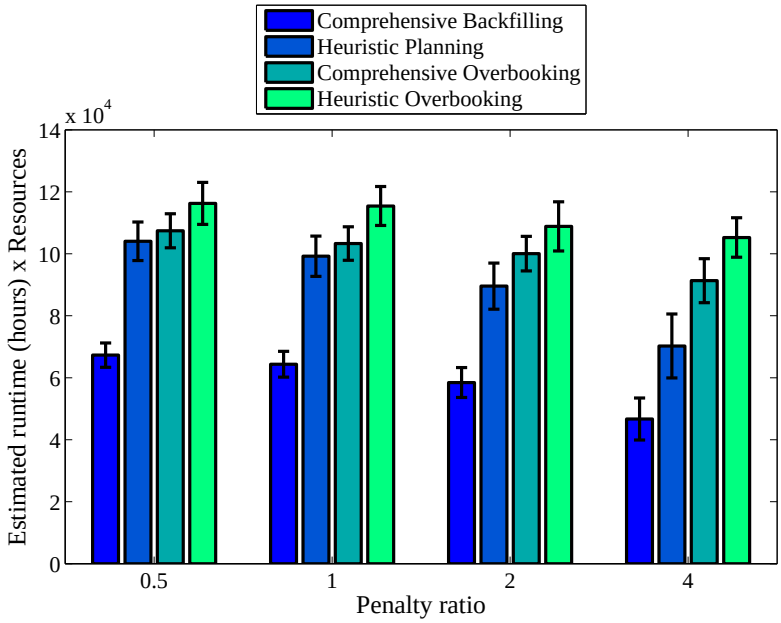
Table 6.10: *The Risk simulation results with SDSC-DataStar trace.*

Risk simulation	Penalty Ratio			
Runtime based statistics	0.5	1	2	4
Comprehensive backfilling	67,306	64,357	58,461	46,669
Heuristic planning	104,040	99,210	89,560	70,250
Comprehensive overbooking	119,180	111,430	101,030	93,410
Additional comp. profit	77%	73%	73%	100%
Heur. overbooking	137,380	129,370	118,470	99,020
Additional heuristic profit	32%	30%	32%	41%
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	67,306	64,357	58,461	46,669
Heuristic planning	104,040	99,210	89,560	70,250
Comprehensive overbooking	107,420	103,300	100,030	91,320
Additional comp. profit	60%	61%	71%	95%
Heur. overbooking	116,270	115,410	108,850	105,240
Additional heuristic profit	12%	16%	22%	50%

Figure 6.22 SDSC-DataStar: The gain for the risk acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

For the non-overbooking strategies, the income of the backfilling approach decreased only slightly. This indicates that due to the risk aware overbooking strategy fewer jobs failed because a higher risk is estimated. The income of the backfilling approach decreased from 67,306 VCs to 46,669 VCs. The heuristic decreased from 104,040 VCs to 70,250 VCs. For both strategies, the additional gain increased when the penalty ratio increased. The profit for the heuristic overbooking fell from 137,380 VCs to 99,020 VCs (for the runtime-estimations based simulation) and 116,270 VCs to 105,240 VCs (for the resource based simulation), respectively. For the heuristic overbooking, with a penalty factor of 4, this was still about 70% of the profit that was made with a penalty factor of 0.5. The gain of the comprehensive overbooking decreased from 119,180 VCs to 93,410 VCs and from 107,420 VCs to 91,320 VCs. However, this was 77% and 95% of the gain with a penalty factor of 0.5.

The total gain of the comprehensive approach was worse than the gain of the heuristic. However, the additional benefit of the comprehensive overbooking approach was higher than that of the heuristic overbooking.

For the comprehensive overbooking approach, the additional gain increased from 60% to 100%. At the top, the profit of the heuristic overbooking increased by 50%.

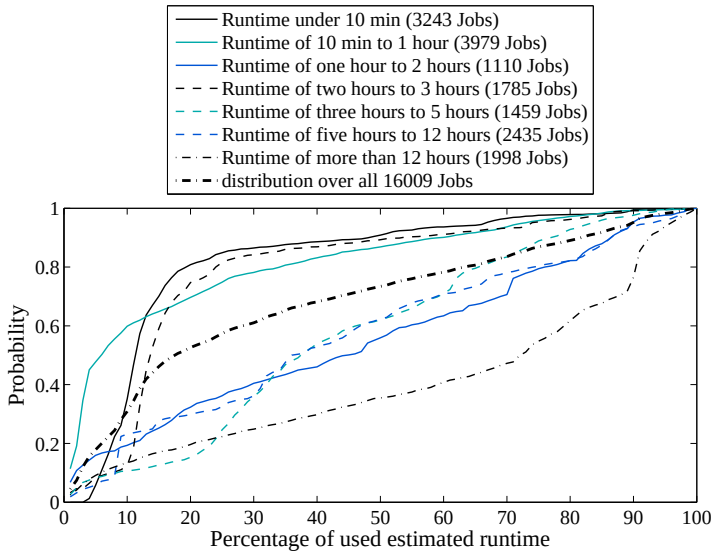
In this simulation, the security factor added to the risk, 4 for the comprehensive and 2 for the heuristic overbooking, seemed to be well chosen because the additional profit was almost stable. This simulation showed that it is possible to get a predictable, additional profit with the overbooking strategies. For the other simulations, this also means that the security factors, as well as the input PDFs, should be adjusted with actual monitoring information to allow predictable results.

The presentation of the SDSC-DataStar simulation ends here, and the last simulation based on the trace of the San Diego Supercomputer Center follows.

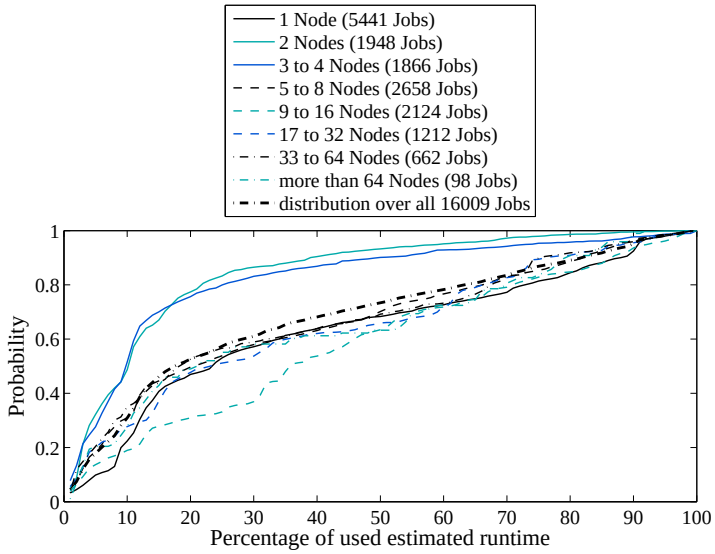
6.6 SDSC-SP2

This section presents the results for the San Diego Supercomputer Center (SDSC) SP2 (SDSC-SP2) trace. The SDSC-SP2 cluster system had 128 resources. From the 59,725 jobs of the trace, the last 20,000 jobs were submitted to the system. The previous jobs were used to learn the statistical input; these were 16,009 entries. The remaining 20,000 jobs were not usable because users runtime-estimations or other important information were missing.

Figure 6.23 SDSC-SP2: The CDF functions.



(a) The runtime-estimation analysis of the SDSC-SP2 trace.



(b) The resource analysis of the SDSC-SP2 trace.

Figure 6.23 (a) shows the cumulative density functions calculated with the statistical runtime-analysis of the workload trace. Figure 6.23 (b) shows the cumulative density functions according to the booked resources.

The runtime-estimation analysis shows more distributed plots. The two groups of jobs with an estimated duration of one hour were estimated worst. Jobs between two and three hours also had bad estimations.

Jobs with a longer estimated duration used more of their booked runtime. The jobs from one to 2 hours, five to 12 hours, and three to 5 hours were increasingly better estimated. The best estimation had the jobs with more than 12 hours runtime.

The resource based CDFs have a similar shape. However, jobs with two to four nodes differ here; their estimation quality was worse than the average. Jobs with 9 to 16 nodes were slightly better estimated than the average.

The more divergent runtime-estimation based PDFs might produce the better overbooking results. The following simulation results will examine this.

6.6.1 PoF Acceptance Test

Like for all other traces, the PoF based acceptance test is interpreted first. Table 6.11 summarizes the results of the PoF acceptance test simulation. The table shows the average results of the 20 test runs for the runtime-estimation and resource statistics and the surplus of the overbooking strategies in percent. Again, the non-overbooking strategies showed that the heuristic was more productive than the backfilling approach. The heuristic planning earned about 24,000 VCs, while the backfilling earned 17,000 VCs.

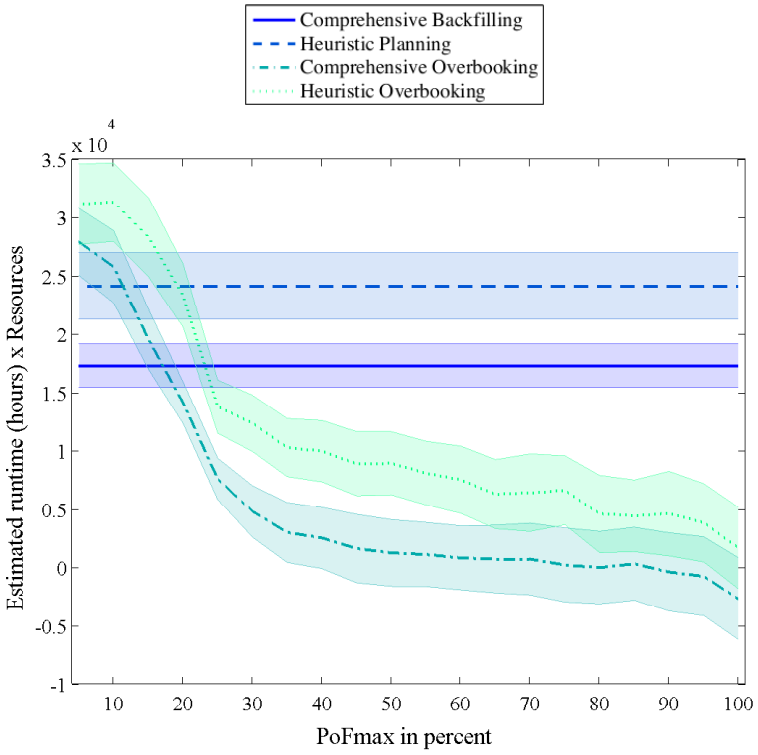
Table 6.11: *The PoF max gain with SDSC-SP2 trace.*

PoF simulation	runtime	profit	PoF_{max}	resources	profit	PoF_{max}
Comp. backfilling	17,280			17,280		
Heuristic planning	24,090			24,090		
Comp. overbooking	27,860	61%	0.05	27,530	59%	0.55
Heur. overbooking	31,290	29%	0.1	31,550	30%	0.55

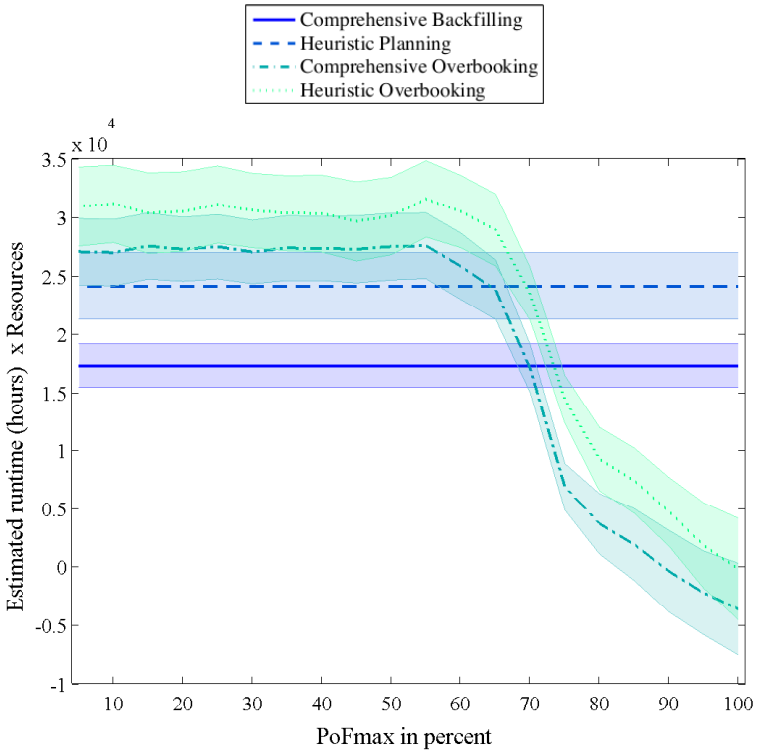
The curves for the runtime-estimation and resource simulation are shaped differently. The overbooking results of the runtime-estimation based simulation decreased almost instantly with an increasing PoF_{max} threshold over 0.05. Contrary, the plot of the resource-based simulation is almost linear up until a threshold of 0.6 and then decreases. However, both curves indicate that the underlying statistical functions were not very accurate. Due to the, at first, not changing shapes and then directly decreasing gain, the statistics were not promising for the overbooking approaches.

The runtime-estimation based PDFs should be treated with most caution. The input PDFs seem to indicate that the jobs would use less runtime than they really do. In practice, an update of the statistics would be necessary to operate successfully.

Figure 6.24 SDSC-SP2: The gain for the PoF acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

Table 6.12: *The Risk simulation results with SDSC-SP2 trace.*

Risk simulation	Penalty Ratio			
Runtime based statistics	0.5	1	2	4
Comprehensive backfilling	18,126	17,276	15,574	12,171
Heuristic planning	26,417	24,093	19,444	10,147
Comprehensive overbooking	14,424	14,099	17,865	20,577
Additional comp. profit	-20%	-18 %	14%	69%
Heuristic overbooking	20,129	9,295	12,887	24,675
Additional heuristic profit	-23%	-61%	-33%	143%
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	18,126	17,276	15,574	12,171
Heuristic planning	26,417	24,093	19,444	10,147
Comprehensive overbooking	28,168	27,233	24,892	21,801
Additional comp. profit	55%	57%	59%	79%
Heuristic overbooking	3,1427	30,598	28,222	25,575
Additional heuristic profit	18%	27%	45%	152%

The heuristic overbooking in the runtime-estimation PDF based simulation had a maximum gain of 31,290 VCs with a PoF_{max} between 0.05 and 0.1. The comprehensive overbooking earned 27,860 VCs. With a higher PoF_{max}, the gain decreased. At a PoF_{max} between 0.25 and 0.3, both curves fell through the lines of their corresponding non-overbooking strategies.

The simulation results based on the resource based CDFs were shaped very differently. The gain did not change for a PoF_{max} between 0.05 and 0.6. Only with a PoF_{max} higher than 0.65, the gain decreased. Again and thus for both analyses, the heuristic had better results with about 31,550 VCs compared to the maximum of 27,530 VCs of the comprehensive overbooking. For all PoF_{max} thresholds, the heuristic was better than the comprehensive overbooking.

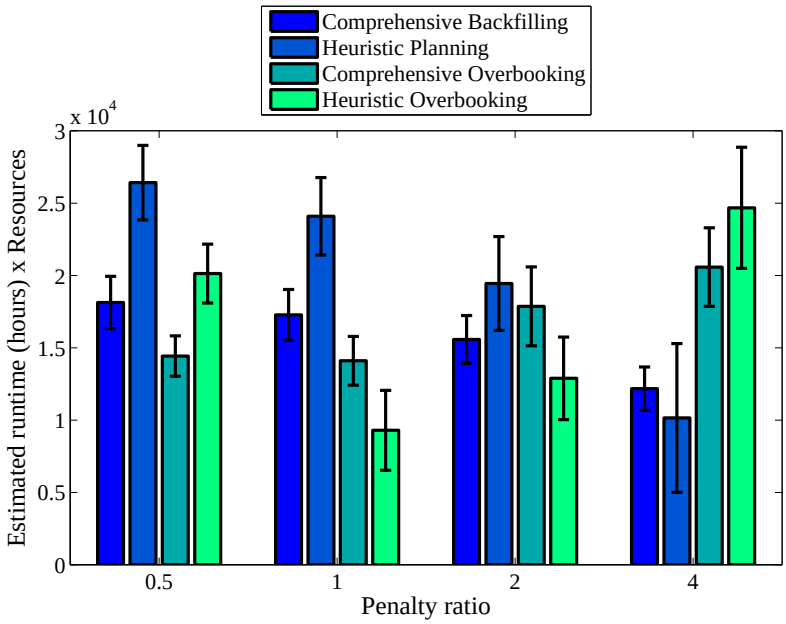
Here, the resource based PDFs seemed to produce a more reliable estimate. This means that changing estimation behaviors are more dangerous if the past jobs had a very divergent behavior. In practice, the input statistics should always be adapted with the last job results. In addition if it is not possible to update the statistics, the general average assumption might produce a more reliable overbooking result.

6.6.2 Risk Acceptance Test

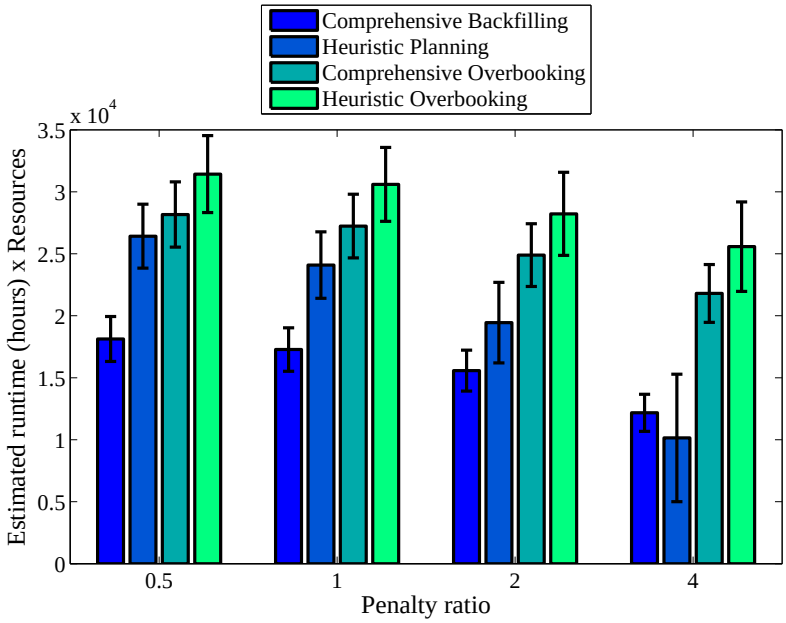
Table 6.12 summarizes the results of the risk based acceptance test. The table shows the average results of the 20 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent. The negative percent values show that the overbooking strategies had a worser result than the corresponding non-overbooking.

For the non-overbooking strategies, the heuristic was better than backfilling with a low penalty ratio but decreased fast with a higher ratio. With a penalty ratio of

Figure 6.25 SDSC-SP2: The gain for the risk acceptance test.



(a) The combined gain of the runtime-estimation PDF.



(b) The combined gain of the resource based PDF.

four, the backfilling approach was better. This means that the heuristic accepted many additional jobs, and a lot of these additional jobs failed.

The runtime-estimation functions were not very accurate and had a bad impact on the overbooking results. This result was similar to the PoF simulation. If the risk was calculated too low, too many jobs that failed were accepted. One can see that the results of the comprehensive and heuristic overbooking were worse than the results of the non-overbooking strategies. The best way to avoid this would be an update of the underlying statistics. A better statistical input should allow us to detect jobs with a higher risk, which could then be declined; thus, fewer jobs would fail. It was possible to work profitable even with the bad CDF functions. This is implicated by the results for the penalty ratio of 4. Here, the overbooking had a positive result because the higher risk due to the higher penalties prevented jobs with a too high PoF from being accepted. To get a better result, a higher security factor must be added to the calculated risk. In practice, the monitoring of actual jobs should reveal that the predictions are faulty. At the end, the additional gain increased from a negative value to a profit of 69% and 143%.

For the simulation with the resource based CDFs, the overbooking strategies were always better than the basic ones. The gain increased from 55% to 79% for the comprehensive approach and from 18% to 152% for the heuristic. This means that the behavior of the classes of different resources was more stable and reliable than the behavior of the classes for the runtime-estimations.

6.7 Summary

The last section of this chapter summarizes the most important facts of the results of the simulations with the PWA traces. The section begins with a summary of the results of the PoF acceptance test simulation, continues with the results of the risk based acceptance test, and concludes with a discussion of the runtime that was used by the planning algorithms.

6.7.1 PoF Results

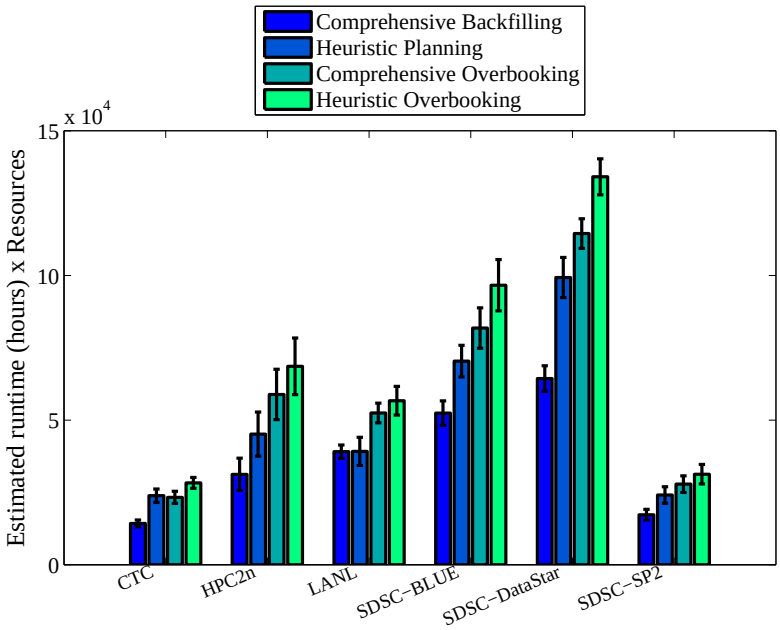
Figure 6.26 and Table 6.13 summarize the maximum gain of all six simulations with the runtime-estimation and resource analysis from the PWA. Two different kinds of input functions for the simulation were applied. One was based on grouped estimated runtime PDFs and the other one on grouped resource PDFs. They revealed if it is beneficial to take different sources as input for a statistical analysis.

Table 6.13: *Additional gain in % compared to the underlying non-overbooking strategy for the PoF acceptance test and the PWA traces.*

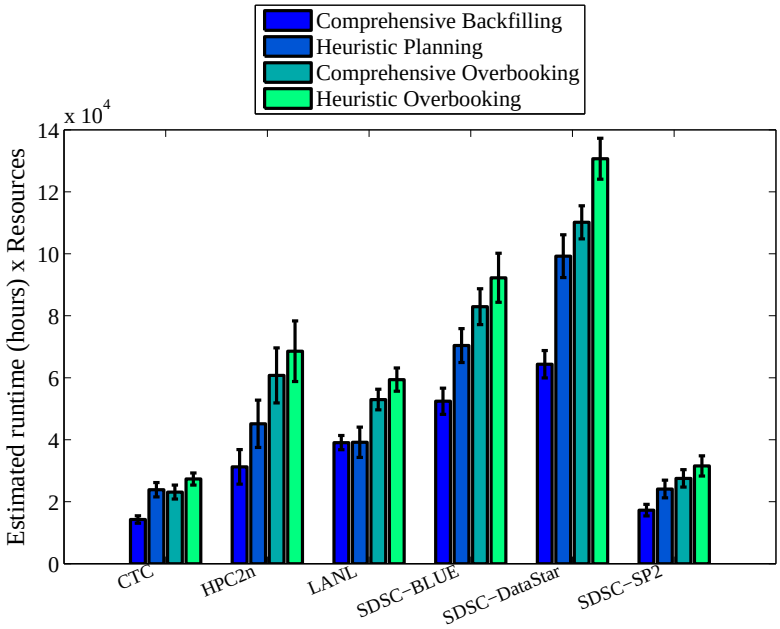
Trace		runtime	$PoF_{\max}$	resources	$PoF_{\max}$
CTC	compreh.	63%	0.1	63%	0.6
	heuristic	19%	0.15	15%	0.6
HPC2n	compreh.	88%	0.9	94%	0.9
	heuristic	52%	1	52%	1
LANL	compreh.	34%	0.05	36%	0.05
	heuristic	45%	0.1	52%	0.05
SDSC-BLUE	compreh.	56%	0.6	58%	0.2
	heuristic	37%	0.75	31%	0.85
SDSC-DataStar	compreh.	78%	0.4	71%	0.75
	heuristic	35%	0.6	32%	0.7
SDSC-SP2	compreh.	61%	0.05	59%	0.55
	heuristic	29%	0.1	30%	0.55

In Figure 6.26, one can see that the heuristic approaches always provided a higher peak profit than the comprehensive approaches. This is caused by the fact that the heuristic was more flexible than the backfilling and the comprehensive overbooking. The main reason is that the backfilling does not reschedule. This means, jobs are always given the complete runtime. The comprehensive overbooking allows later jobs to start earlier if the preceding jobs ended before the estimated runtime. In addition, the comprehensive overbooking is inflexible because the jobs had to run on the resources that were selected at negotiation. The heuristic overbooking can start the jobs on any resource of the cluster. This provides a higher flexibility as well as a higher utilization and more profit.

Figure 6.26 Summary of the $PoF_{\max}$ threshold simulations.



(a) Maximum gain of the runtime-estimation PDF based simulations.



(b) Maximum gain of the resource based PDF based simulations.

However, the comprehensive overbooking was more reliable. Table 6.13 shows that the additional profit compared to the backfilling approach was higher than the additional profit of the heuristic overbooking. The results of the comprehensive overbooking approach also fluctuated less.

Summary The simulations showed that it is possible to increase the profit with overbooking. The impact of the different input statistics on the results of overbooking was lower than expected. One can see that many shapes of the figures are relatively similar. Evaluating other kinds of groups might allow us to find more suitable input statistics.

Regardless of the overbooking strategy, the runtime-estimation based PDFs provided, on average, better results in 6 out of 12 simulations, while the resource based PDFs gave better results for the other 6. The additional gain of the strategies lay between 15% and 94%.

6.7.2 Risk Results

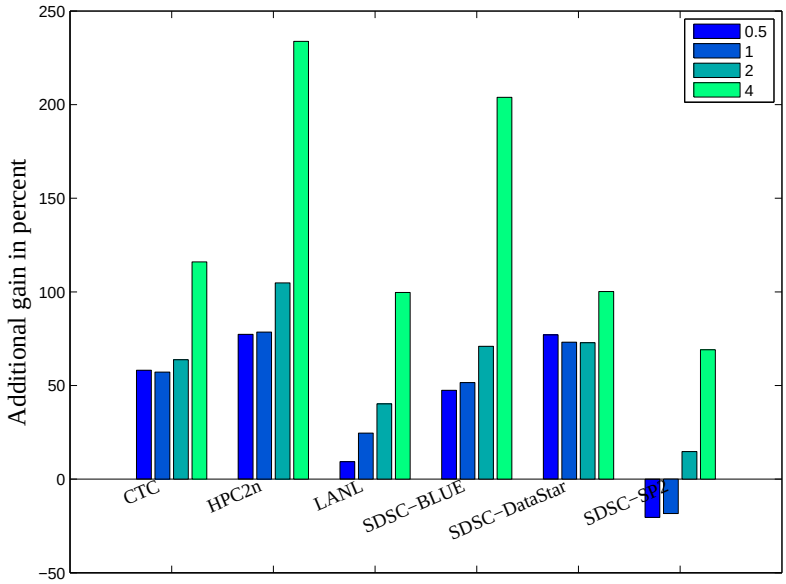
Figures 6.27 and 6.28 and Table 6.14 show the maximum gain for the simulations with the risk acceptance test. The risk results indicate that the comprehensive overbooking is more reliable. It had a higher additional gain compared to the backfilling. The heuristic overbooking had a higher total profit.

Table 6.14: *Additional gain in % compared to the underlying non-overbooking strategy for the risk acceptance test and the PWA traces.*

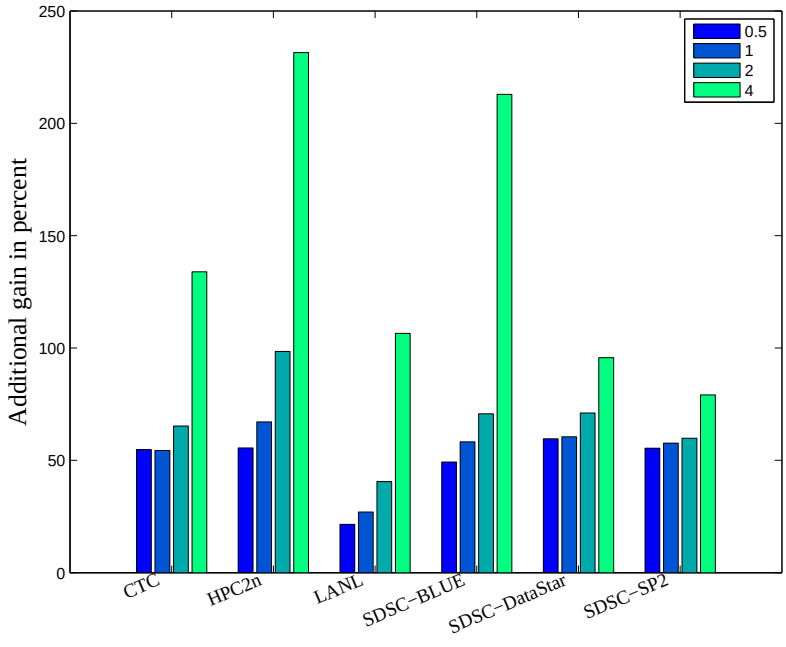
Trace		runtime	ratio	resources	ratio
CTC	compreh.	160%	4	134%	4
	heuristic	20%	1	19%	4
HPC2n	compreh.	234%	4	231%	4
	heuristic	61%	4	60%	4
LANL	compreh.	100%	4	107%	4
	heuristic	116%	2	84%	2
SDSC-BLUE	compreh.	204%	4	231%	4
	heuristic	120%	4	68%	4
SDSC-DataStar	compreh.	100%	4	95%	4
	heuristic	41%	4	50%	4
SDSC-SP2	compreh.	69%	4	79%	4
	heuristic	143%	4	152%	4

There is an outlier in the Figure 6.28 for the LANL trace. The negative bar for the penalty ratio of 4 does not mean that the gain is negative. It is caused by the negative profit of the heuristic planning and backfilling, while the gain of the corresponding overbooking strategies is positive. Therefore, the maximum gain in Table 6.14 is given for the LANL trace at a penalty ratio of 2. However,

Figure 6.27 Summary of the Risk simulations with comprehensive Overbooking.

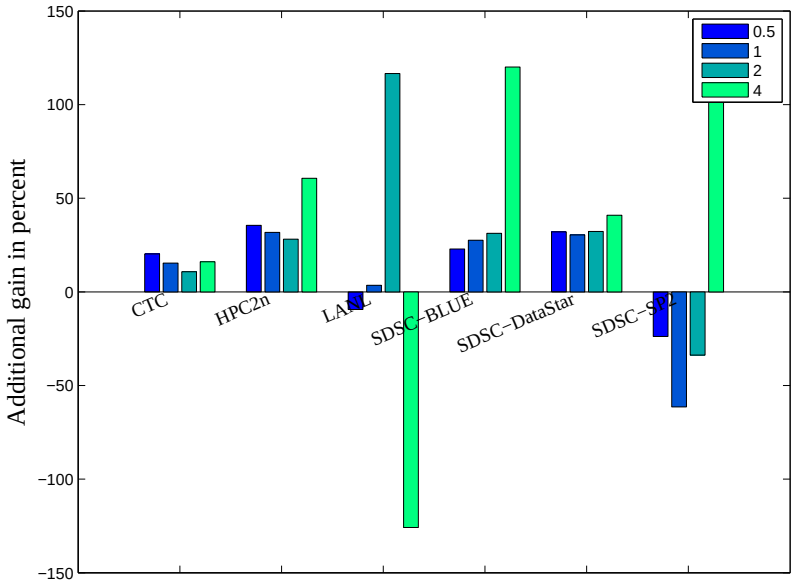


(a) Runtime-estimation PDF based simulations

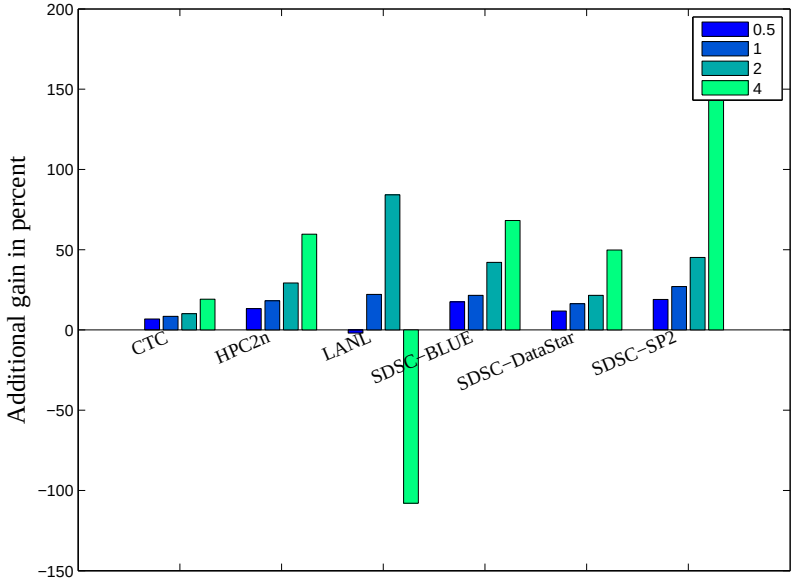


(b) Resource PDF based simulations

Figure 6.28 Summary of the Risk simulations with heuristic Overbooking.



(a) Runtime-estimation PDF based simulations



(b) Resource PDF based simulations

at a ratio of 4 the income of the non-overbooking heuristic was negative, while the overbooking heuristic remains positive. Therefore, the -107% and -125% additional win at a ratio of 4 means that the result was still positive.

Summarizing the comprehensive overbooking, it tended to be more reliable. Therefore, the profit of this overbooking method always increased with a higher penalty ratio. This indicates that with a higher risk due to a higher penalty ratio the comprehensive overbooking accepted fewer jobs and these jobs had a higher probability to be successful. Therefore, less jobs failed, and the income increased even with higher penalty ratios. However, this also might indicate that the heuristic approach could profit from a higher security factor. For these simulations, the opportunity had to be four times higher than the risk to ensure that a job could be accepted. In practice, a monitoring tool should compare the calculated failures of the jobs to the real failures. If the amount of failures is higher than calculated, the security factor should be increased.

Summarizing the heuristic results, they show a little different picture. Here, the additional gain did not always increase with a higher penalty factor. The heuristic overbooking algorithm produced a more reliable result with the resource based PDFs than with the runtime-estimation functions. On the one hand, the runtime-estimation functions had more often a negative additional gain. On the other hand, runtime-estimation functions had a higher profit if they were positive.

Comparing the statistics for the resource function based simulation, the gain always increased with a higher penalty factor. For the runtime-estimation simulation, the results tended to shrink in the beginning and increased in the end. This might indicate that the risk factor of 2 should be adapted in praxis according to the relation of fee and penalty.

Summary The results of the risk acceptance test based overbooking were dependent on the security factor and the quality of the input statistics. While the statistics of the runtime-estimation analysis produced a slightly better profit, they also produced negative results. This shows that a regular update of the probability density functions, as well as the security factor, is important. A better statistical input would reflect a higher risk; thus, fewer jobs would be accepted.

The importance of the security factor was also shown for the jobs of the SDSC-SP2 trace. For the runtime-estimation analysis, the results of the overbooking approach for a low penalty ratio were worse than the ones of the heuristic planning without overbooking. This shows that the security factor was too low. As a consequence, too many jobs were accepted that failed in the end. However, it was possible to work profitable with the SDSC-SP2 trace. This was shown by the results for the penalty ratio of 4. In this case, the overbooking simulation of the SDSC-SP2 trace had a positive result.

6.7.3 Runtime

The Section 3.4 provided already a brief theoretical runtime analysis of the overbooking algorithms. When we assumed that the number of jobs n would be

dominant, the theoretical approach shows that both approaches would have the same complexity class and need quadratic runtime $O(n^2)$.

However in the discussion about practical issues, we found that when a user negotiates online with a cluster system to add a job, not the time for scheduling all jobs offline but the duration for calculating the probability and applying the acceptance test for the user's job is important.

For the case that one job is submitted, the heuristic seems to be faster than the comprehensive approach because the runtime only depends on the number of planned p in the system $O(p)$. The comprehensive overbooking approach has a runtime that is dependent on the number of jobs in the system p and the number of resources of the cluster k as well as the time needed for the FFT calculations $O(t \log(t))$. Following, the complexity would be $O(t \log(t) pk)$.

To evaluate whether this has an impact in practice, this simulation measured the runtime of the overbooking algorithms. The results are presented in the following.

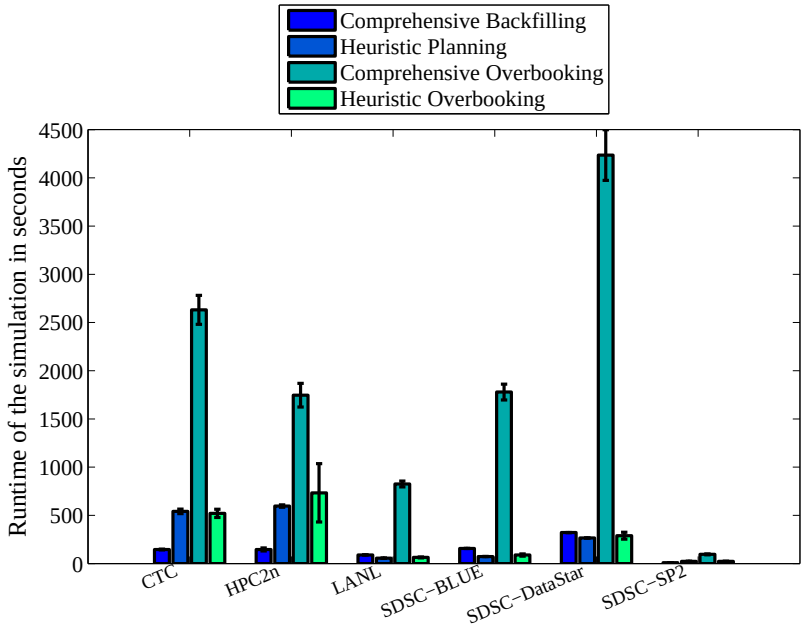
The placing strategy that was used in these simulations was similar to best fit. Related work shows that heuristic approaches, implementing this strategy, have a runtime that is dependent quadratic to the power of the jobs and also dependent to the number of resources [Burk 04]. However, these implementations did not consider overbooking and were not affected by the PDF convolutions. Beneath the results of the overbooking mechanisms, the runtime of the simulation runs were additionally measured to get an idea about the dependence of the runtime on this factors.

Figures 6.29 and 6.30 illustrate the average runtime. Due to the similarity of the approaches, the runtime is only shown as a summary. One can immediately see that the comprehensive overbooking used far more runtime than the other approaches. The runtime did not depend on the risk or PoF acceptance test or the strategies.

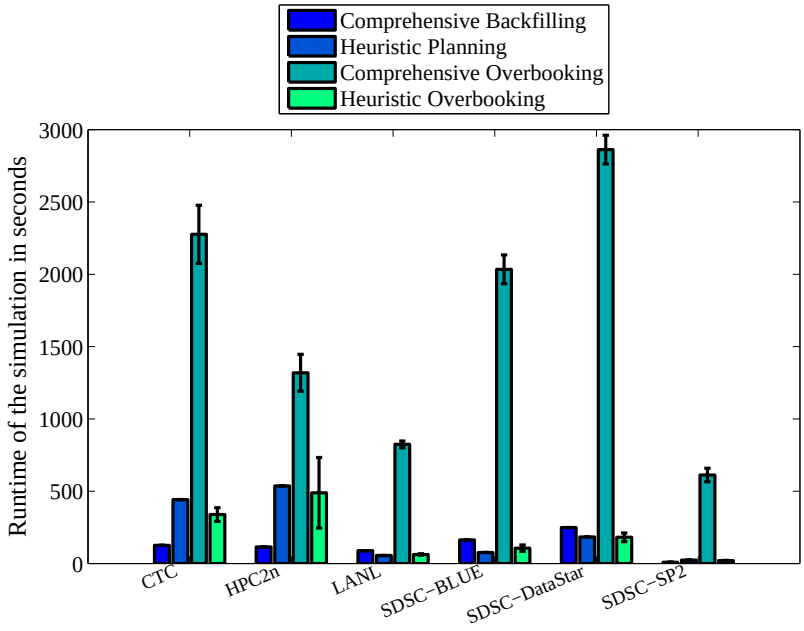
The overbooking strategies work on a best-fit basis. They first try to place the job in a plan with the full estimated runtime, and if that is not possible, with overbooking. Due to the many jobs and resources, the comprehensive overbooking has to check many possible gaps. In addition, the comprehensive overbooking approach has to calculate the convolutions of the jobs and the combined functions for every possible gap in which jobs were running beforehand. Therefore, many fast Fourier transformations have to be done. The simulations showed that this takes far more time than the other strategies. The backfilling approach without overbooking was often faster than the heuristic planning and overbooking. This was caused by the fact that the backfilling assigns the full estimated runtime and does not replan, while the heuristics replan after each job's end.

The heuristic planning and the heuristic overbooking have a similar runtime. The heuristic overbooking is very fast because it does not do convolutions anymore. The heuristic only calculates the PoS with the given maximum runtime of the gap. In some traces, the heuristic overbooking took more time than the heuristic planning because building the sum of a PDF was more time consuming than the heuristic planning. In other traces, the heuristic overbooking was faster. This

Figure 6.29 Runtime of the PoF simulations.

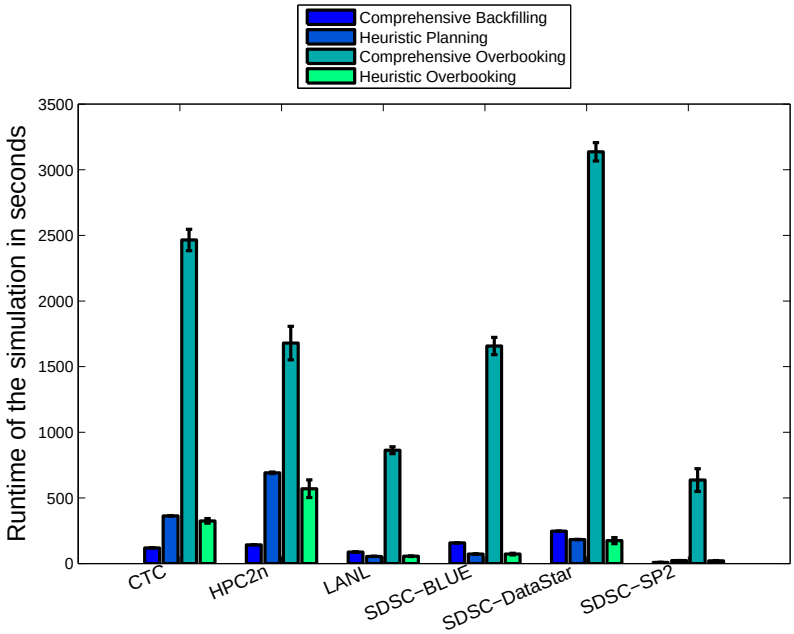


(a) Runtime-estimation PDF based simulations

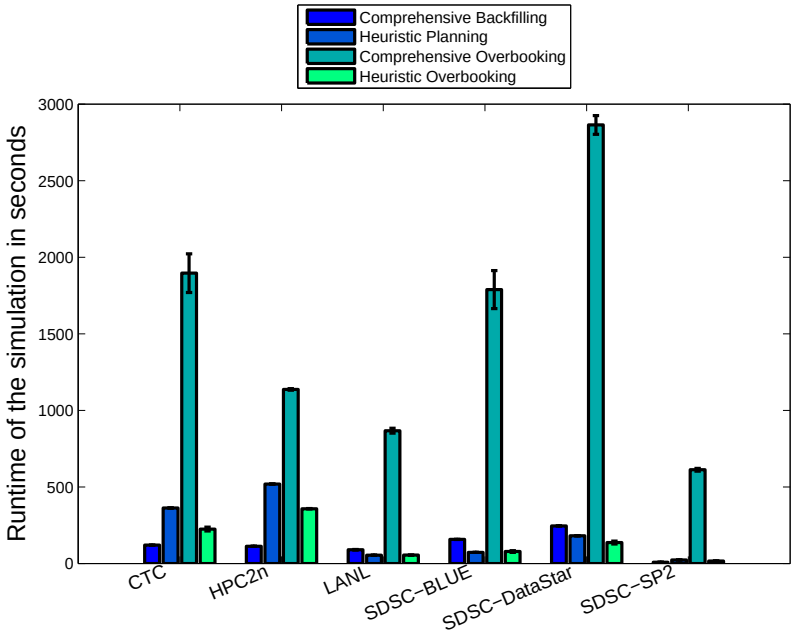


(b) Resource PDF based simulations

Figure 6.30 Runtime of the Risk simulations.



(a) Time Simulations



(b) Resource Simulations

might be because the additionally accepted jobs reduce the gaps in the schedule and, following, fewer gaps have to be checked.

However, the same strategies had different runtimes for the different simulations and the confidence intervals indicate that they will not overlap. This shows that the runtime is dependent on the simulated environment, more precisely the size k of the cluster that is simulated. A look at the simulated clusters shows that the clusters with more resources used more runtime in the simulations. This is because more resources offered more possible gaps for a job that had to be checked. Following, and as assumed in Section 3.4.3, the simulations showed that time needed the FFT calculations and the size of the cluster k have an influence on the runtime.

The following simulations with the Arminius trace should show if the statistics also have an influence on the runtime. Therefore, the amount of resources will be the same for all simulation runs with Arminius.

7 Evaluation of the Arminius Cluster

This chapter presents the results of the evaluation based on a trace of the Arminius cluster of the PC^2. This trace allowed simulations for a broader area of statistical analyses because more detailed input data was available and the environment was known. These advantages allowed to extend the evaluation of the overbooking algorithms by two additional kinds of statistical input functions, the used application environments and individual users. The contents of this chapter are the results of the simulations according to the four analyses of runtime-estimation, resources, applications, and users.

Contents

7.1 Runtime-Estimation Analysis 144
7.1.1 PoF Acceptance Test 144
7.1.2 Risk Acceptance Test 146
7.2 Resources Analysis 149
7.2.1 PoF Acceptance Test 149
7.2.2 Risk Acceptance Test 152
7.3 Application Analysis 157
7.3.1 PoF Acceptance Test 157
7.3.2 Risk Acceptance Test 160
7.4 Submission-User Analysis 163
7.4.1 PoF Acceptance Test 163
7.4.2 Risk Acceptance Test 165
7.5 Summary 169
7.5.1 PoF Results 169
7.5.2 Risk Results 170
7.5.3 Runtime 171

The Arminius cluster had 200 nodes with dual core CPUs and 4 GB RAM. For the simulation of the jobs from 2007 were used to learn the statistics. These were 33,318 jobs. The jobs from 2008 were the input and 60 test batteries of 1000 jobs were submitted. Therefore, the results of this evaluation base on 60,000 jobs instead of 20,000 that were applied by the PWA simulations. The Arminius statistics are described in more detail in Chapter 4.

7.1 Runtime-Estimation Analysis

The presentation of the results of the Arminius simulation starts with the runtime-estimation analysis. Figures 4.2 and 4.1 from Section 4 show the cumulative and probability density functions according to the eight estimated-runtime classes. According to the simulations of the PWA traces, this section starts with the results of the PoF threshold acceptance tests and continues with the risk based acceptance tests.

7.1.1 PoF Acceptance Test

This section discusses the results of the simulation with the acceptance test based on the PoF_{max} threshold. Table 7.1 summarizes the maximum gain of the runtime-estimation statistics based simulation. The table shows the average results of the 60 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent.

Table 7.1: *The PoF max gain with Arminius and the runtime-estimation PDF*

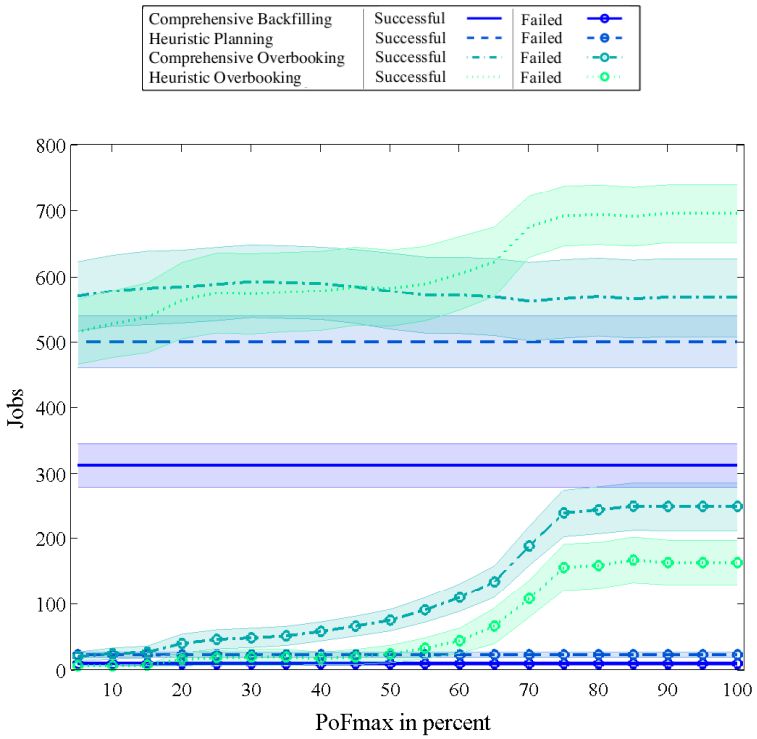
PoF simulation	VCs	additional gain	PoF_{max}
Comprehensive backfilling	13,944		
Heuristic planning	16,022		
Comprehensive overbooking	16,140	16%	0.05
Heuristic overbooking	19,813	24%	0.5

Figure 7.1 shows the successful and failed jobs in (a) and their differences in (b). The transparent areas around the average results of the 60 simulation runs are the 95% confidence intervals.

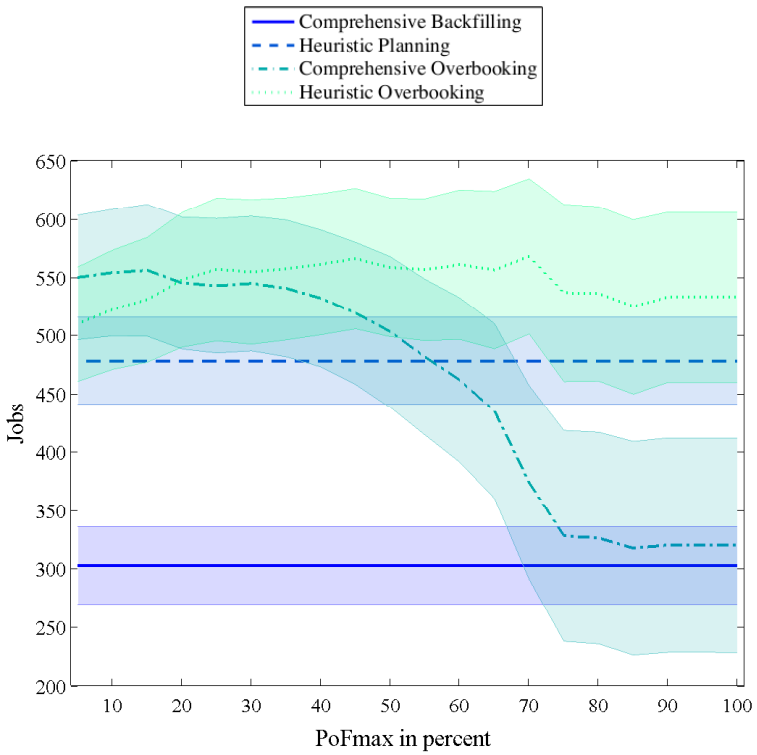
As already discussed in the last chapter, the curves of the non-overbooking strategies in Figure 7.1(a) remained on the same level because the increasing PoF_{max} threshold does not affect them. For both strategies, the number of successful jobs increased slightly in the beginning until a $PoF_{\text{max}} = 0.2$. After that, the result-line was almost constant. For the heuristic, the number of successful jobs increased on average from 500 to nearly 700 jobs. The comprehensive approach remained at about 600 of 1000 possible jobs. For the heuristic, the number of failed jobs increased little until a $PoF_{\text{max}} = 0.5$ and then faster until a $PoF_{\text{max}} = 0.75$. From this threshold, the number of failed jobs, as well as the number of successful jobs, does not further increase. This means that with a $PoF_{\text{max}} = 0.75$ the simulated cluster was completely utilized.

The corresponding number of successful minus failed jobs is displayed in Figure 7.1(b). Even with 60 simulation runs instead of the 20 from the simulations of the PWA traces, the huge confidence intervals indicate how uncertain the average results of the highly different single simulations are. The fluctuations of the results additionally underlined that single simulation runs could be very different from the mean. However, due to the higher number of the 60 simulation runs, the

Figure 7.1 Arminius: Jobs with PoF acceptance test and runtime-estimation PDFs.



(a) The successful and failed jobs of Arminius.



(b) The difference of successful and failed jobs.

Table 7.2: *The Risk simulation results with Arminius and the runtime-estimation PDFs*

Risk simulation	Penalty Ratio			
Runtime-estimation CDFs	0.5	1	2	4
Comprehensive backfilling	15,835	13,944	10,160	2,593
Heuristic planning	18,054	16,022	11,960	3,834
Comprehensive overbooking	17,861	15,360	11,483	4,204
Comprehensive gain	13%	10%	13%	62%
Heuristic overbooking	21,389	18,582	16,239	10,360
Heuristic gain	18%	16%	36%	170%

average results were smoother than the serrated shape of the curves from some PWA traces.

The comprehensive approach tended to be better with very low PoF_{max} thresholds and then decreased until a threshold of $PoF_{\text{max}} = 0.75$. At this threshold, the cluster seemed to be completely utilized. It had an average value of 500 to 600 jobs for all thresholds and seemed to be more constant and predictable.

Figure 7.2 presents the resulting gain in virtual coins. Here, the huge confidence intervals indicate the uncertainty of the average result of the 60 test runs as well.

Regarding the penalties, as shown in Figure 7.2(a), the results of all 4 strategies were nearly equal up to a PoF_{max} threshold of 0.5. From this point on, the penalties increased for the overbooking approaches until the $PoF_{\text{max}} = 0.75$. At this point, the cluster system generally seemed to be fully utilized. Figure 7.2(b) illustrates the combined gain. The huge confidence intervals show the high probability that the single simulation results differ.

However, one can see that the heuristic planning produced a better result than the comprehensive overbooking.

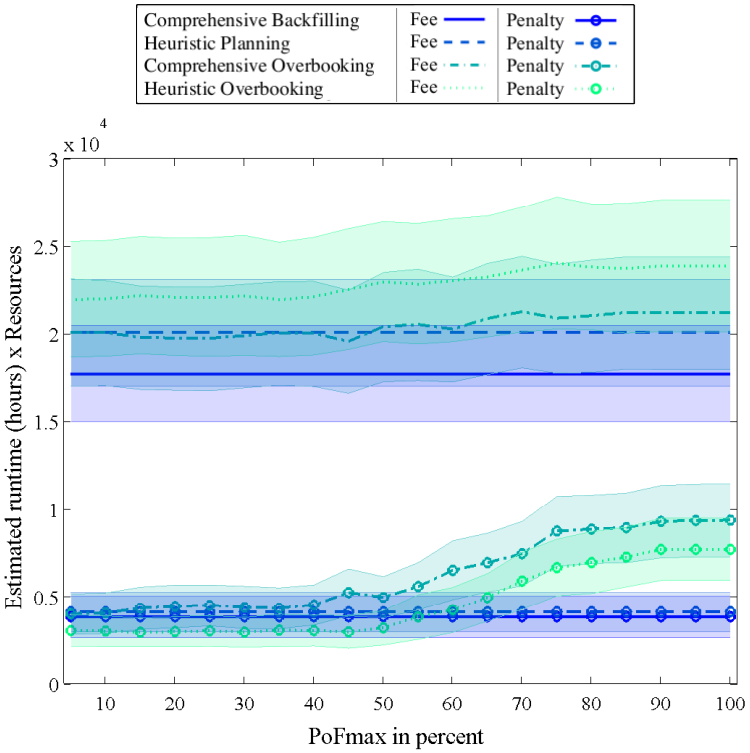
In the beginning, both overbooking approaches had a constant average result until a PoF_{max} of about 0.5. From this point on, the profit fell. The decrease stopped for the comprehensive approach at a PoF_{max} of 0.75 and at a PoF_{max} of 0.9 for the heuristic. This is interesting because the utilization of the simulation seemed to be saturated at a PoF_{max} of 0.75. For the heuristic however, some jobs still failed until a PoF_{max} of 0.9. The maximum gain for the heuristic overbooking had nearly 20,000 VCs. The heuristic planning had, with 16,000 VCs, the same results as the comprehensive overbooking.

7.1.2 Risk Acceptance Test

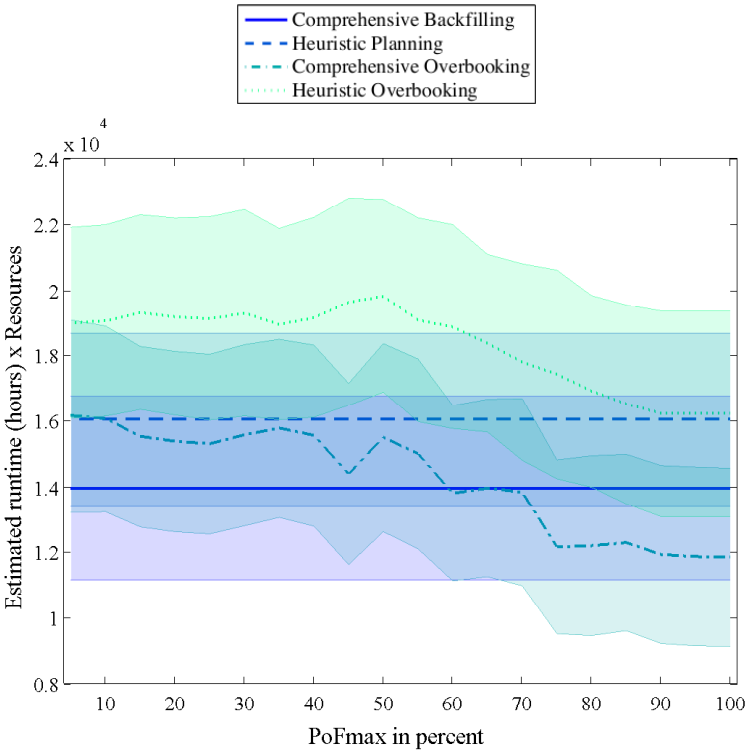
Table 7.2 summarizes the average results and the surplus of the overbooking strategies and Figure 7.3 illustrates the number of jobs resulting from the simulation with the risk acceptance test.

Figure 7.3(a) shows the number of successful and failed jobs. It underlines that the strategies that employ overbooking could successfully schedule more jobs than the non-overbooking strategies. From the non-overbooking strategies, the

Figure 7.2 Arminius: Profit and penalties with PoF acceptance test and runtime-estimation PDFs.

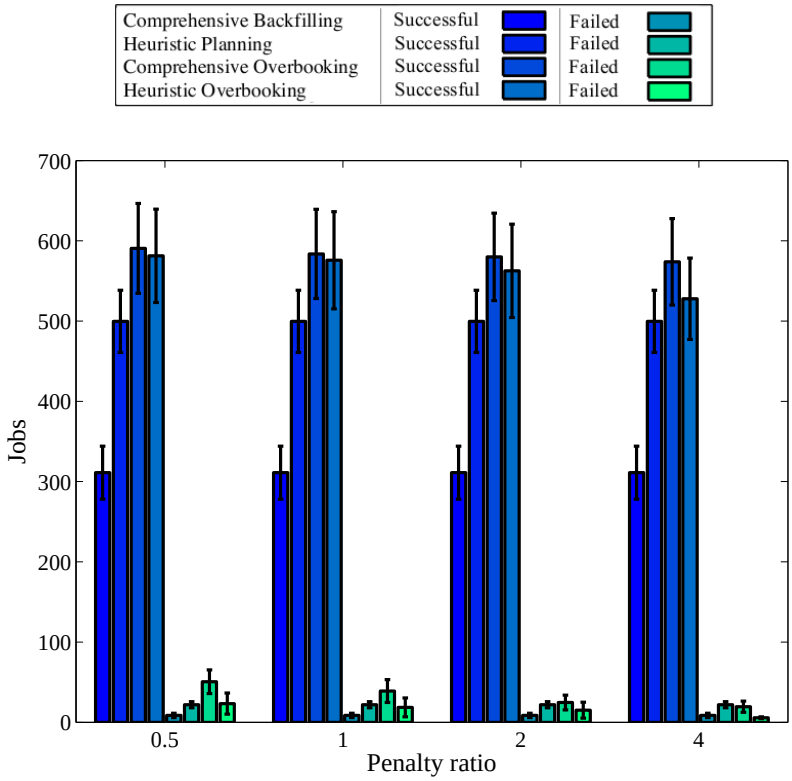


(a) The fees and penalties of the Arminius trace.

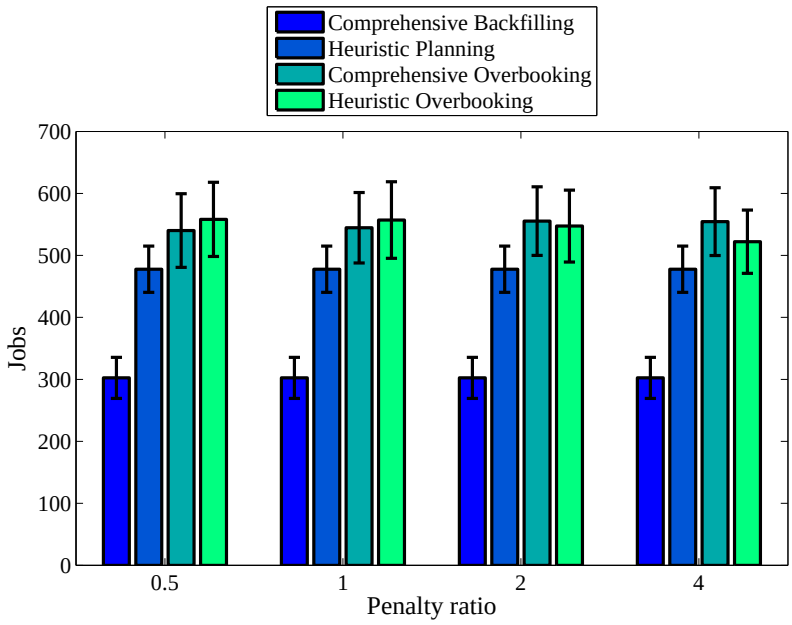


(b) The gain.

Figure 7.3 Arminius: Jobs with Risk acceptance test and runtime-estimation PDFs.



(a) The successful and failed jobs of the Arminius trace.



(b) The difference of successful and failed jobs.

heuristic had the most jobs. Looking at the successful jobs of the overbooking strategies, the heuristic seems to have a slightly decreasing number, while the comprehensive approach seems to have a constant result. However, a look at the failed jobs shows that for both strategies their number decreased with a higher penalty ratio due to the increasing risk.

Figure 7.3(b) illustrates the difference of successful minus failed jobs. Again, one can see that the non-overbooking approaches were not affected by the risk assessment, while the overbooking strategies had a different results.

Figure 7.4 shows the resulting profit and penalties for the simulation. While Figure 7.4(a) shows that the number of failed jobs was very small compared to the successful jobs, Figure 7.4(a) illustrates that the corresponding penalties were high.

For the non-overbooking strategies, all jobs failed because of node outages. Their penalties were doubled with each doubled penalty ratio. The overbooking strategies, in this simulation mainly the heuristic, profit from the risk based acceptance test. Fewer jobs failed and the penalties increased at a lower rate.

For the heuristic, Figure 7.4(b) indicates that the number of accepted jobs decreased; this allowed more jobs to be successfully executed. Therefore, the profit was better compared to the other strategies. The additional gain of the heuristic increased from 18% at a penalty ratio of 0.5 to 170% with a penalty ratio of 4. The additional gain of the comprehensive overbooking increased from 13% to 62% with the same parameters.

7.2 Resources Analysis

This section continues with the presentation of the results of the Arminius trace simulations and the applied resource statistics input. Figures 4.4 and 4.3 of Section 4.2.2 show the input functions.

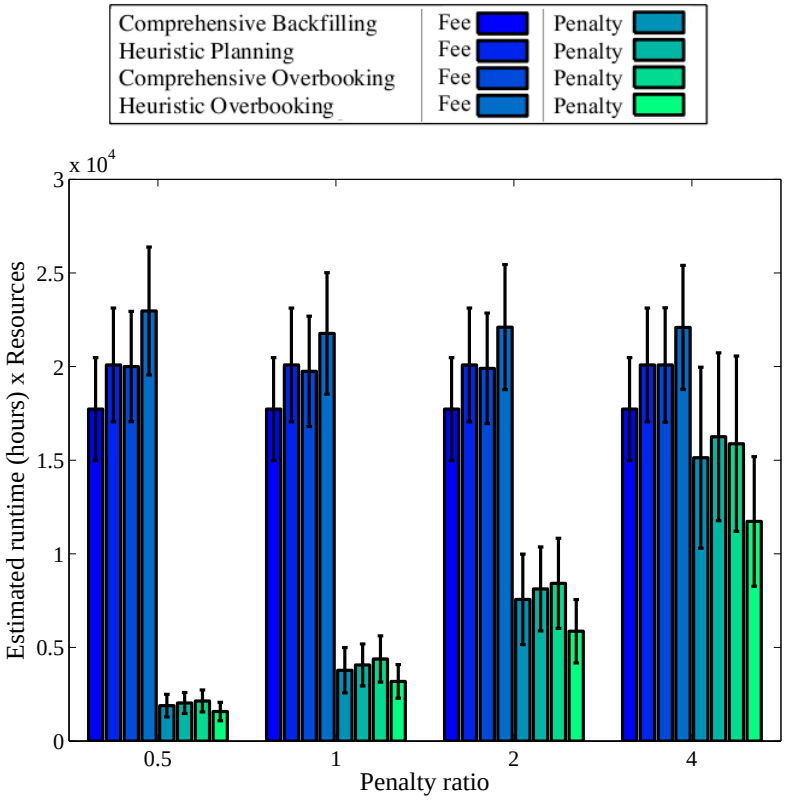
7.2.1 PoF Acceptance Test

The results of the simulation with the $PoF_{\max}$ threshold acceptance test are discussed here. Table 7.3 summarizes the average results of the 60 test runs for the resource statistics and the surplus of the overbooking strategies in percent with the given $PoF_{\max}$.

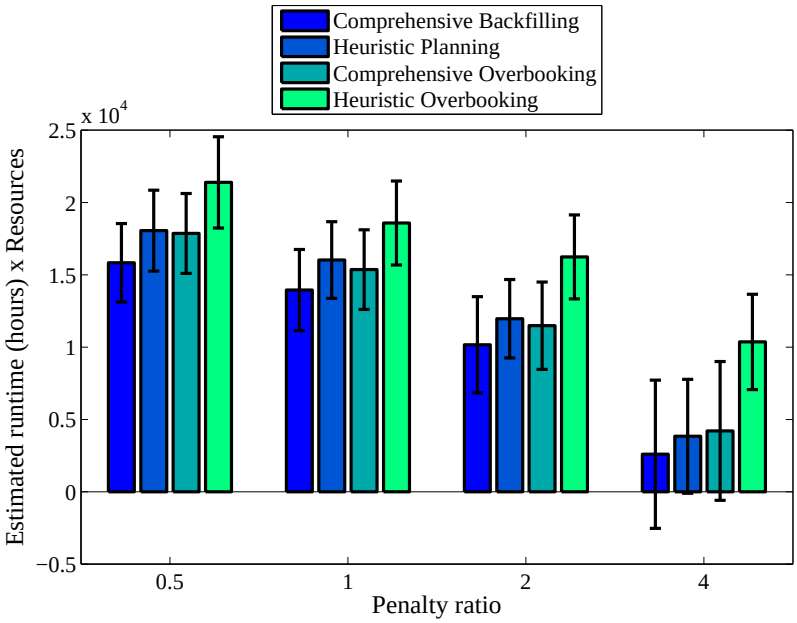
Table 7.3: *The PoF max gain with Arminius and the resource based PDF*

PoF simulation	VCs	additional gain	$PoF_{\max}$
Comprehensive backfilling	13,944		
Heuristic planning	16,022		
Comprehensive overbooking	16,158	16%	0.15
Heuristic overbooking	19,556	22%	0.4

Figure 7.4 Arminius: Profit and penalties with Risk acceptance test and runtime-estimation PDFs.

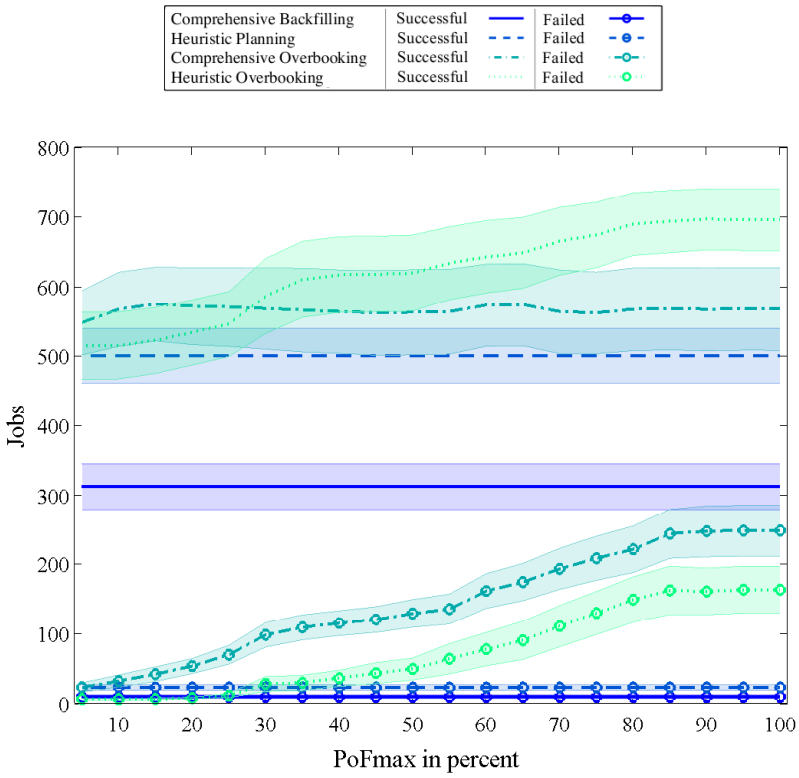


(a) The fees and penalties of the Arminius trace.

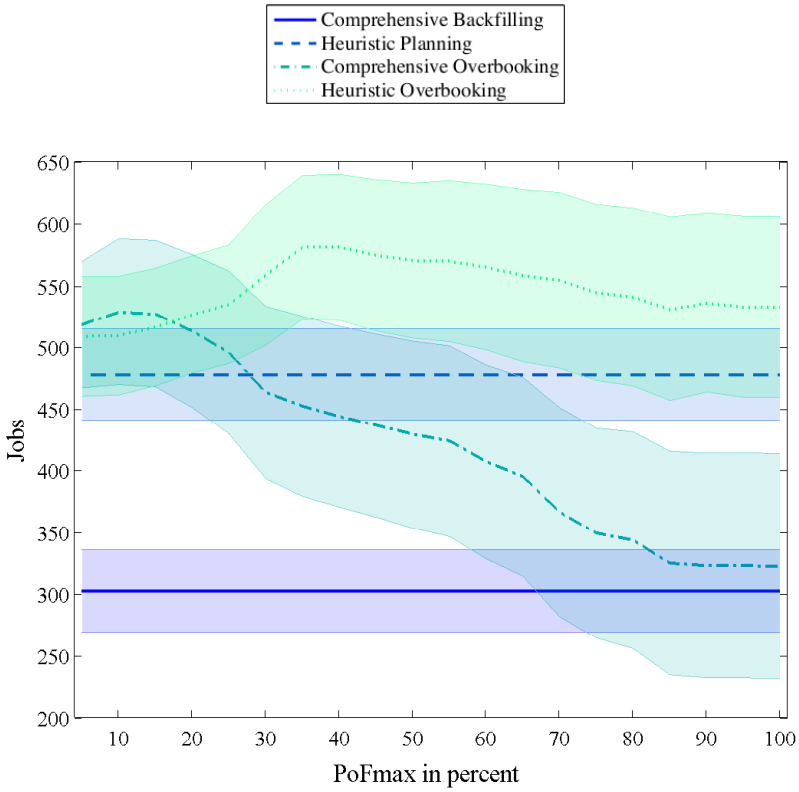


(b) The gain.

Figure 7.5 Arminius: Jobs with PoF acceptance test and resource based PDFs.



(a) The successful and failed jobs of the Arminius trace.



(b) The difference of successful and failed jobs.

Figure 7.5 illustrates the successful and failed jobs. Again, the 95% confidence intervals show that the average values could overlap. Even with 60 test runs, the results of the Arminius trace differed more than the results of the PWA traces. A reason for this strange behavior might be that the simulation learned the job behaviors from the year 2007 and used the jobs from 2008 for the evaluation. The PWA traces had a shorter time span. This gave the users less time and chance to change their behavior.

Due to the longer time span of the Arminius traces, the average result of 2007 could differ from the results of the 2008 jobs. Thus, simulation runs in the end of 2008 used an outdated statistical input. In practice, a steady update process has to incorporate the changing job behaviors into the PoF calculation.

Figure 7.5(a) illustrates that the comprehensive overbooking had, up to a PoF_{max} threshold of 0.3, a higher number of successful jobs. For a higher PoF_{max} threshold, the heuristic overbooking was more successful. For the comprehensive overbooking, the number of failing jobs increased steadily, while the number of successful jobs remained at the same level.

However, the number of failed jobs dominates even the heuristic overbooking results. Figure 7.5(b) illustrates that the comprehensive overbooking had the maximum number of jobs at a PoF_{max} of 0.15, while the heuristic had the best number at a PoF_{max} of 0.35.

Figure 7.6 shows the corresponding gain. The fees and penalties increased over the complete threshold range. The confidence intervals for the gathered fees and penalties overlap in Figure 7.6(a). The combined gain in Figure 7.6(b) illustrates that the overbooking results were best with a PoF_{max} of up to 0.15 (comprehensive approach) and 0.4 (heuristic approach) and decrease thereafter.

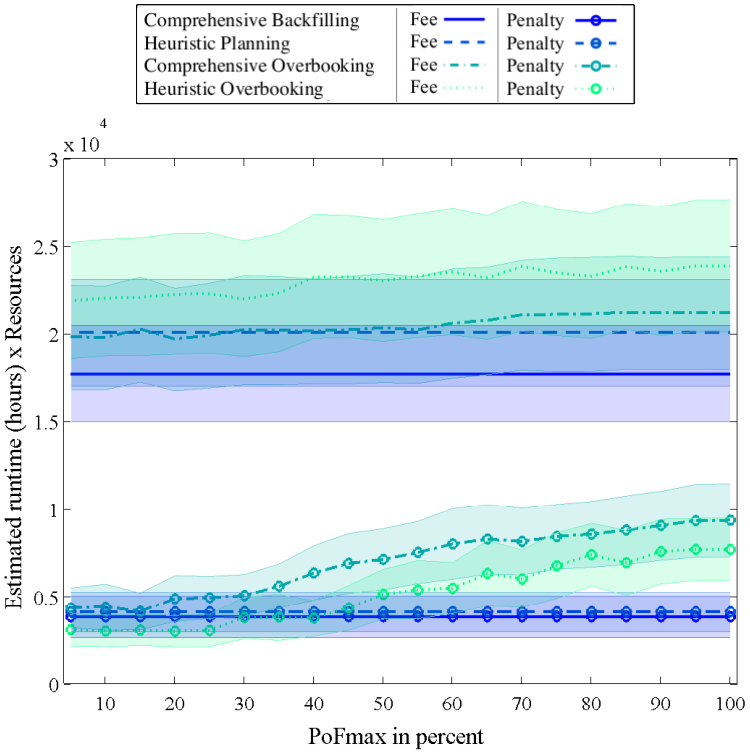
Due to its inflexibility, the comprehensive overbooking was only able to get the same result as the heuristic planning. The heuristic planning did not overbook jobs. The maximum gain of the comprehensive overbooking was, with about 16,000 VCs, 16% higher than the gain to the backfilling. The heuristic overbooking had a maximum gain of nearly 20,000 VCs. This was an additional 22% gain compared to the heuristic planning and an additional 20% gain compared to the comprehensive overbooking.

7.2.2 Risk Acceptance Test

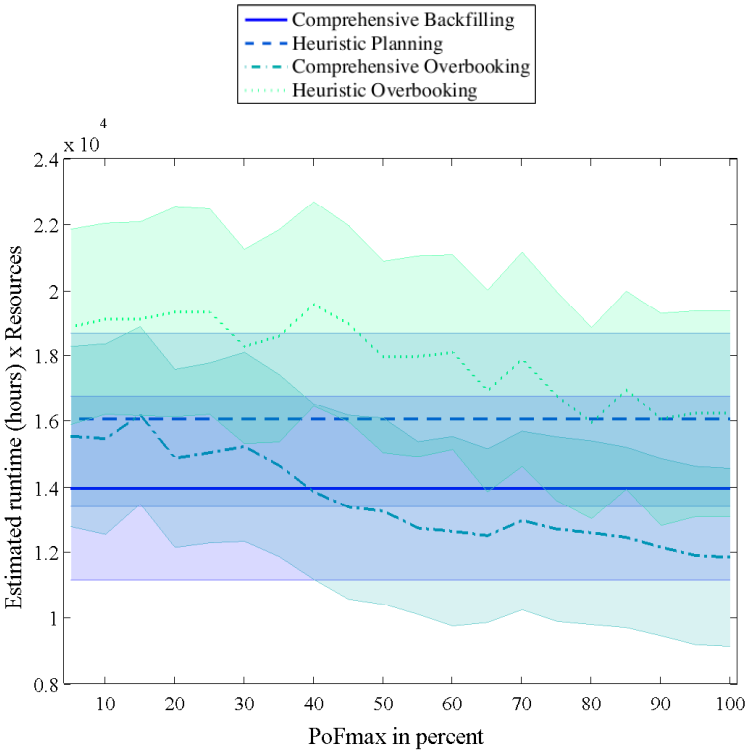
Table 7.4 summarizes the result of the risk acceptance test and the resource simulation of the Arminius trace. The table shows the average results of the 60 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent.

Figure 7.7(a) shows the successful and failed jobs and Figure 7.7(b) the difference of the successful and failed. The non-overbooking strategies did not change. Comparing these resource simulation results to the runtime-estimation simulation results, the number of failed jobs was lower than the comprehensive results and higher than the heuristic.

Figure 7.6 Arminius: Profit and penalties with PoF acceptance test and resource based PDFs.

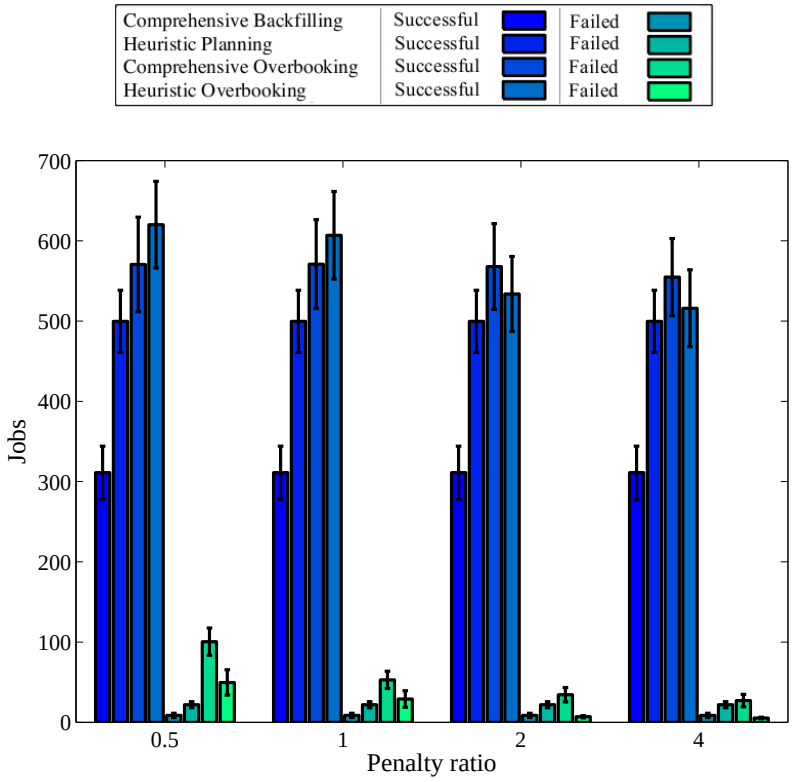


(a) The fees and penalties of the Arminius trace.

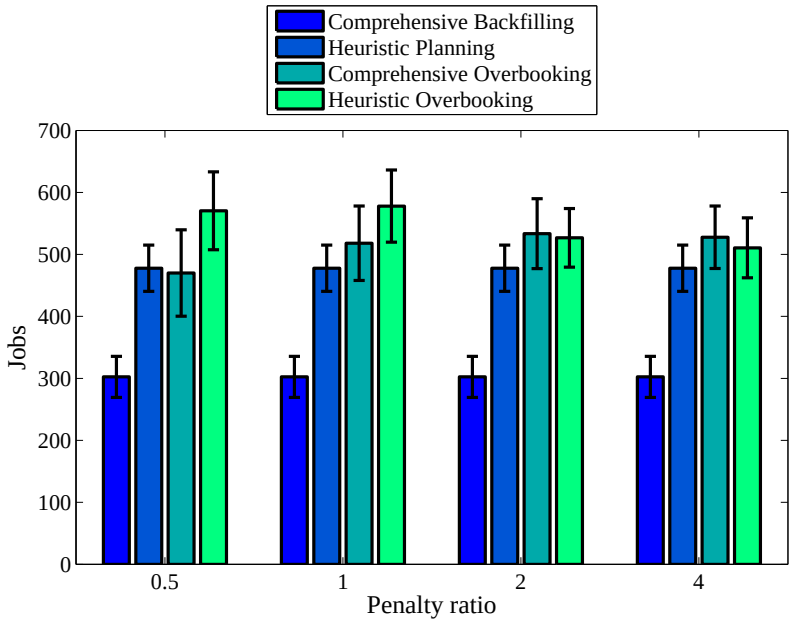


(b) The gain.

Figure 7.7 Arminius: Jobs with Risk acceptance test and resource based PDFs.



(a) The successful and failed jobs of the Arminius trace.



(b) The difference of successful and failed jobs.

Table 7.4: *The Risk simulation results with Arminius and the resource based PDFs*

Risk simulation	Penalty Ratio			
Resource based statistics	0.5	1	2	4
Comprehensive backfilling	15,835	13,944	10,160	2,593
Heuristic planning	18,054	16,022	11,960	3,834
Comprehensive overbooking	17,174	14,830	11,539	3,136
Comprehensive gain	8%	6%	14%	21%
Heuristic overbooking	20,496	19,181	16,305	10,240
Heuristic gain	14%	20%	36%	167%

The difference of successful minus failed jobs in Figure 7.7(b) illustrates that the overbooking strategies reduced the amount of failed jobs with a higher risk. Compared to the heuristic planning, the comprehensive overbooking had a lower result with a penalty ratio of 0.5 and a higher result with a penalty ratio of 1. Starting at a penalty ratio of 2, the comprehensive overbooking had a higher ratio than the heuristic overbooking. When comparing both overbooking strategies, the comprehensive overbooking was too lazy when the risk was low and became much more careful when the risk was higher.

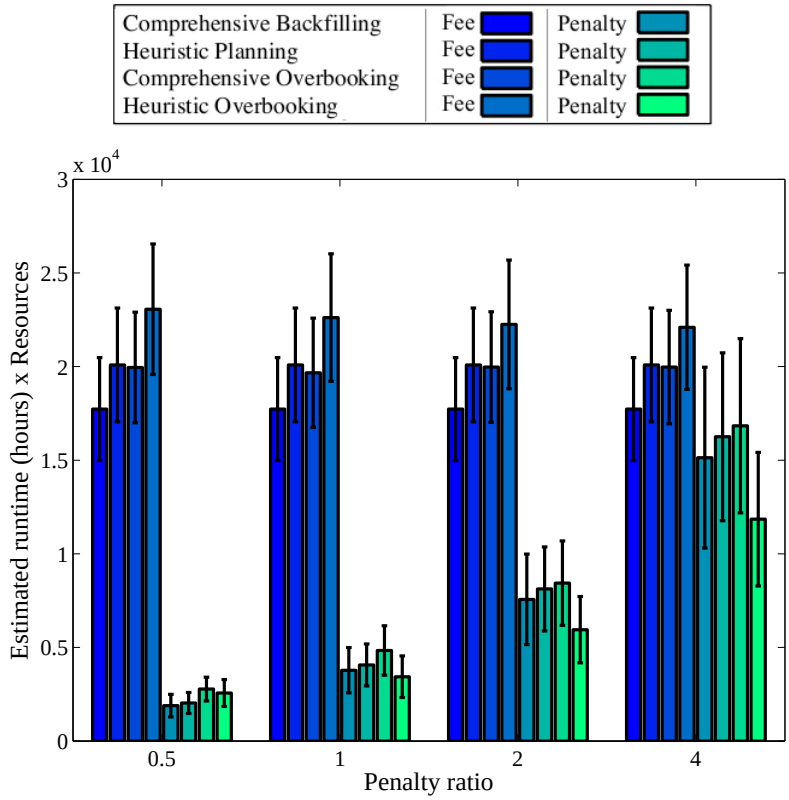
Figure 7.8(a) shows that the shape of the fees and penalty results remained, while the bars of the penalties were closer to the bars of the fees.

The distance between the failed and the successful jobs is bigger than the distance between the fees and penalties for a penalty ratio of 1 or 2. This means that less but very large jobs failed

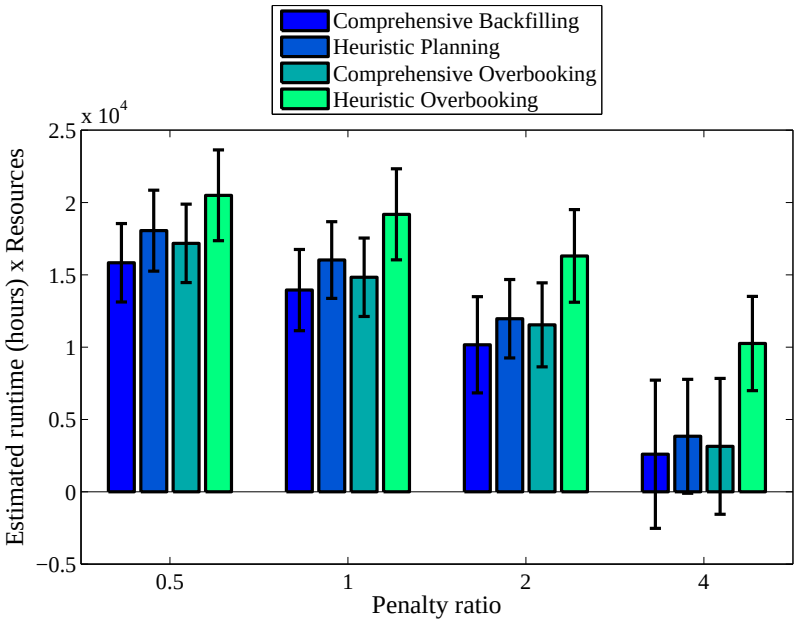
Therefore, Figure 7.8(b) shows that the additional gain for the heuristic increased from 14% to 167%. For the comprehensive overbooking, the gain decreased from 8% to 6% for a penalty ratio between 0.5 and 1. For a penalty ratio of 2, it increased to 14% and to 21% for a penalty ratio of 4. The additional gain of the heuristic compared to the comprehensive overbooking increased from 3,000 VCs or 18% at a penalty ratio 0.5 (difference 17,174 VCs [comp.] to 20,496 VCs [heur.]) to 7,000 VCs or 300% at a ratio of 4 (the difference from 3,136 VCs [comp.] to 10,240 VCs [heur.]).

The confidence intervals show that the backfilling, heuristic planning, and comprehensive overbooking could have negative results with these settings. In practice, this means that the strategies were not careful enough, and the input statistics or the security factor need to be adapted in order to work profitable.

Figure 7.8 Arminius: Profit and penalties with Risk acceptance test and resource based PDFs.



(a) The fees and penalties of the Arminius trace.



(b) The gain.

7.3 Application Analysis

A new kind of statistical input was used for the application statistics based simulation. The simulation used the statistical analysis according to different requested applications. Figures 4.6 and 4.5 in Section 4.2.3 show the input CDFs and PDFs.

7.3.1 PoF Acceptance Test

The evaluation starts with the acceptance test based on the PoF_{max} threshold. Table 7.5 summarizes the maximum gain of the application analysis based simulation. A discussion of the results is shown in the following.

Table 7.5: *The PoF max gain with Arminius and the application PDF*

PoF simulation	VCs	additional gain	PoF_{max}
Comprehensive backfilling	13,944		
Heuristic planning	16,022		
Comprehensive overbooking	15,292	10%	0.05
Heuristic overbooking	19,170	20%	0.1

Figure 7.9 shows the jobs of the simulation. In the beginning and like outlined in Figure 7.9(a), the amount of successful jobs was almost the same for the two overbooking strategies. From a PoF_{max} threshold of 0.15 on, they increased. The comprehensive overbooking results remained at 550 successful jobs after a PoF_{max} threshold of 0.4. The heuristic overbooking increased further to over 700 jobs and remained at this level between a PoF_{max} threshold of 0.7 and 1.

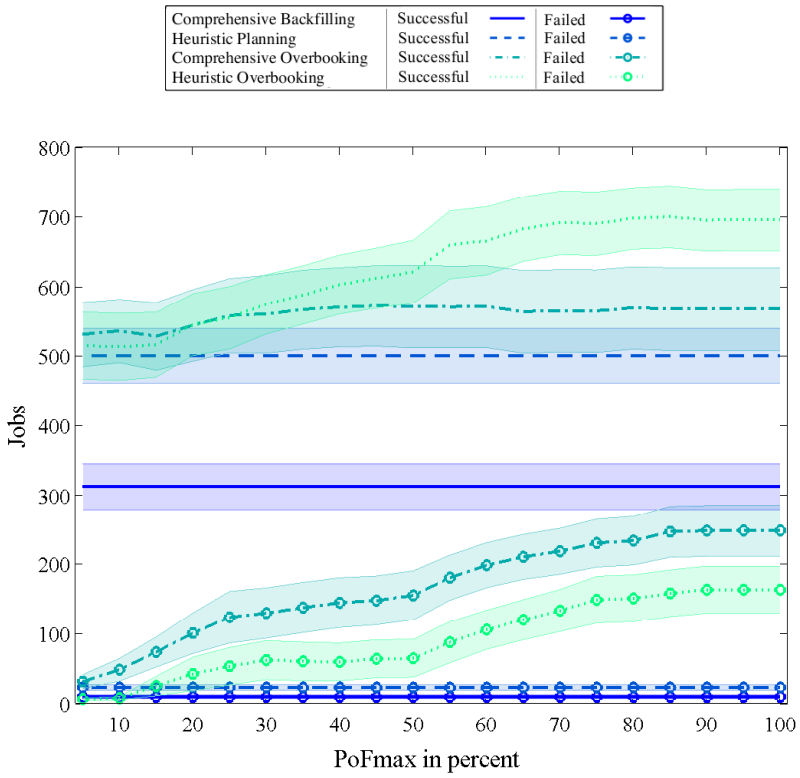
The failed jobs increased for both overbooking strategies from the beginning. At the end, the comprehensive overbooking had with 250 the most failed jobs. The heuristic had 150 failed jobs with a PoF_{max} threshold of 1.

Figure 7.9(b) shows the difference of successful - failed jobs. The 95% confidence intervals around the lines of the two overbooking strategies overlap up until a $PoF_{\text{max}} = 0.5$. The curves for the heuristic increase up to a $PoF_{\text{max}} = 0.55$, while the comprehensive overbooking decrease from the beginning.

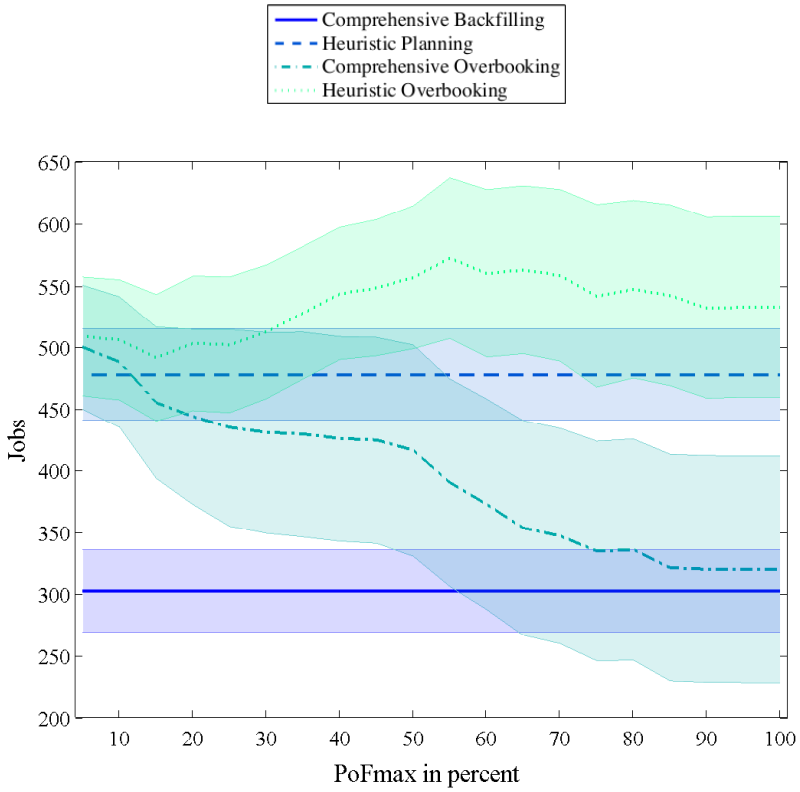
Figure 7.10 shows the resulting fees, penalties, and the combined gain. The overlapping confidence intervals indicate that the results of the single test runs can differ from the average results.

The fees of both overbooking strategies slowly increased with a higher PoF_{max} threshold. The penalties grew at a higher rate. Therefore, Figure 7.10(b) shows that the gain of the simulation decreased with a higher PoF_{max}. The maximum gain of the comprehensive overbooking was 15,292 VCs with a PoF_{max} of 0.05; this means 10% extra. The maximum average gain of the heuristic overbooking was 19,170 VCs with a PoF_{max} of 0.1. This was about a 20% extra income.

Figure 7.9 Arminius: Jobs with PoF acceptance test and application PDFs.

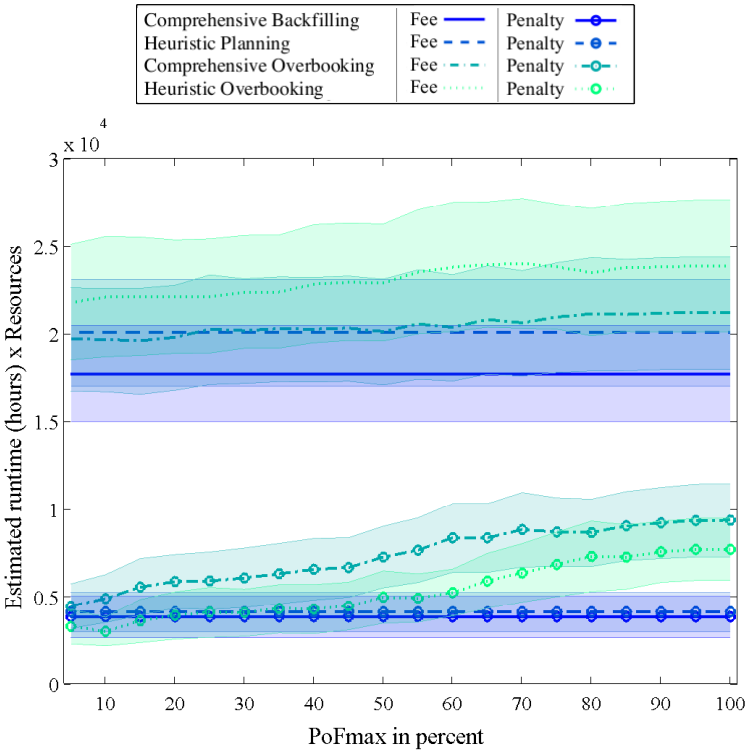


(a) The successful and failed jobs of the Arminius trace.

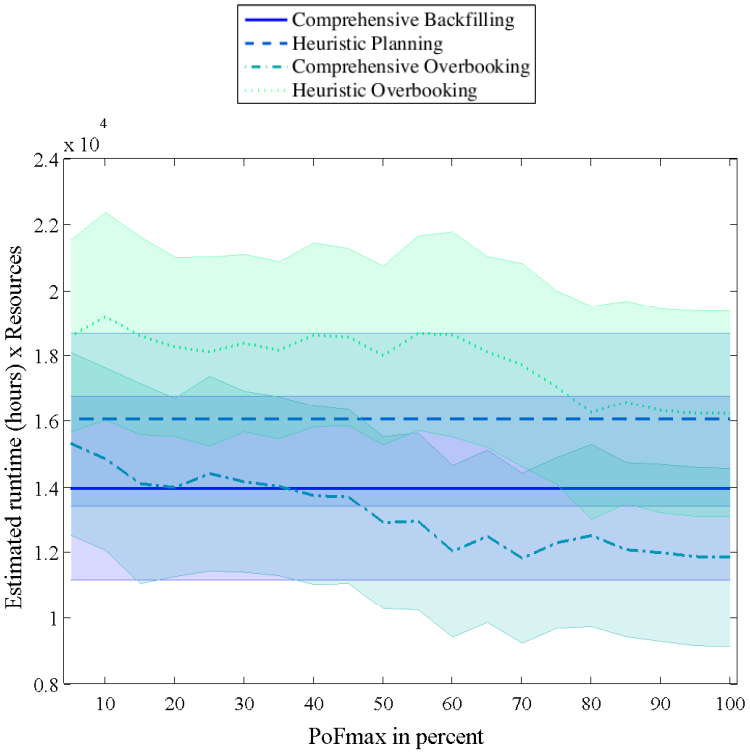


(b) The difference of successful and failed jobs.

Figure 7.10 Arminius: Profit and penalties with PoF acceptance test and application PDFs.



(a) The fees and penalties of the Arminius trace.



(b) The gain.

Table 7.6: *The Risk simulation results with Arminius and the application PDFs*

Risk simulation	Penalty Ratio			
Application based statistics	0.5	1	2	4
Comprehensive backfilling	15,835	13,944	10,160	2,593
Heuristic planning	18,054	16,022	11,960	3,834
Comprehensive overbooking	17,212	13,966	9,750	2,071
Comprehensive gain	9%	0%	-4%	-20%
Heuristic overbooking	20,450	18,230	14,382	10,987
Heuristic gain	13%	14%	20%	187%

7.3.2 Risk Acceptance Test

Table 7.6 summarizes the results of the risk acceptance test and the application statistics. The table shows the average results of the 60 test runs according to the penalty ratios and the surplus of the overbooking strategies in percent.

Figure 7.11(a) shows the successful and failed jobs for the resource simulation and Figure 7.11(b) shows the job difference.

Similar to the resource analysis, the number of failed jobs of the overbooking strategies was high with a low penalty ratio. The number of failed jobs decreased when the penalty ratio grew.

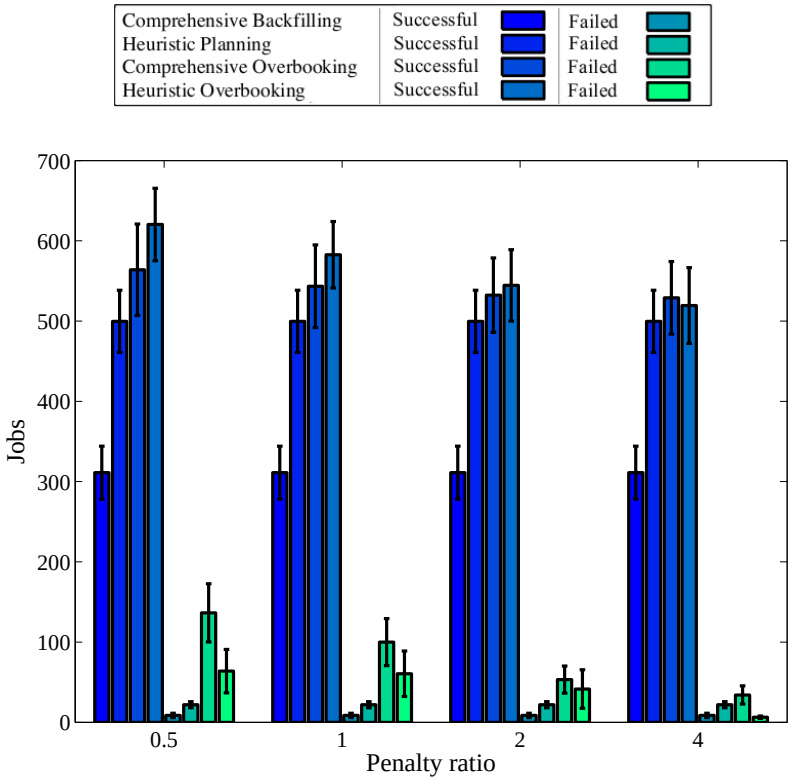
Figure 7.11(b) illustrates the difference of successful minus failed jobs. The overbooking strategies gathered an average number of 500 jobs with a penalty ratio of 1 and 2.

Figure 7.12(a) shows the increasing penalties. The shape of the bars is similar to the results of the resource analysis. Therefore, Figure 7.12(b) illustrates that the additional gain increased monotonically from 13% to 187%, with a higher penalty ratio for the heuristic.

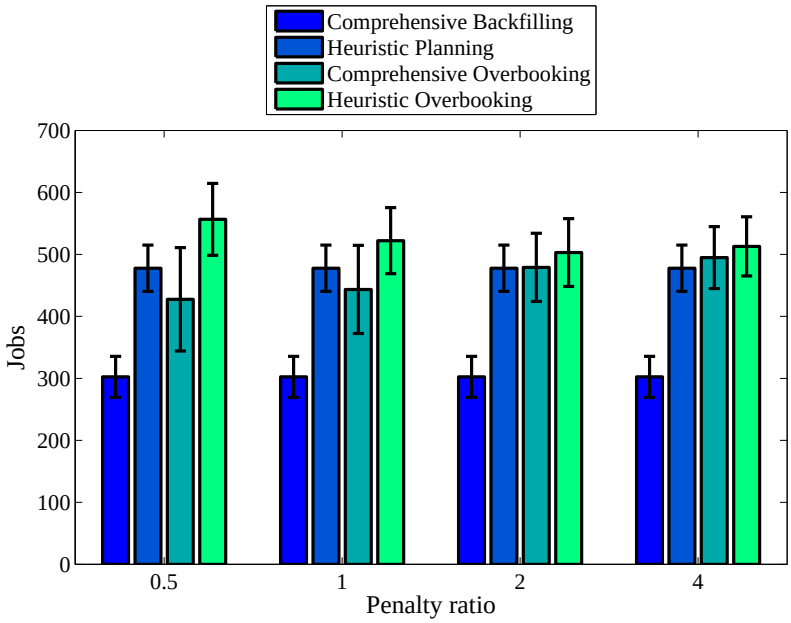
Due to the high amount of SLA violations, even the non-overbooking strategies have almost no gain with a penalty ratio of four. This means that nearly one-fourth of the sold computing time got lost due to resource failures.

The heuristic was able to achieve a very high profit and the comprehensive overbooking performed badly. Starting at a penalty ratio of 1, the comprehensive overbooking had a worse result than the backfilling. In addition, the gain of the comprehensive overbooking decreased with an increasing penalty ratio. The strategy had a very bad performance with the input statistics. Either the selected application classes must be updated to allow suitable risk calculations, or the applications are generally not able to support overbooking. To allow this simulation approach to be successful, we need a higher security factor. The security factor that was added in the simulation to the calculated risk value to ensure that only jobs with very good opportunity were selected was 4. This security factor of 4 was suitable for most simulations, here however, the factor was not high enough. Alternatively, the statistical input must be updated.

Figure 7.11 Arminius: Jobs with Risk acceptance test and application PDFs.

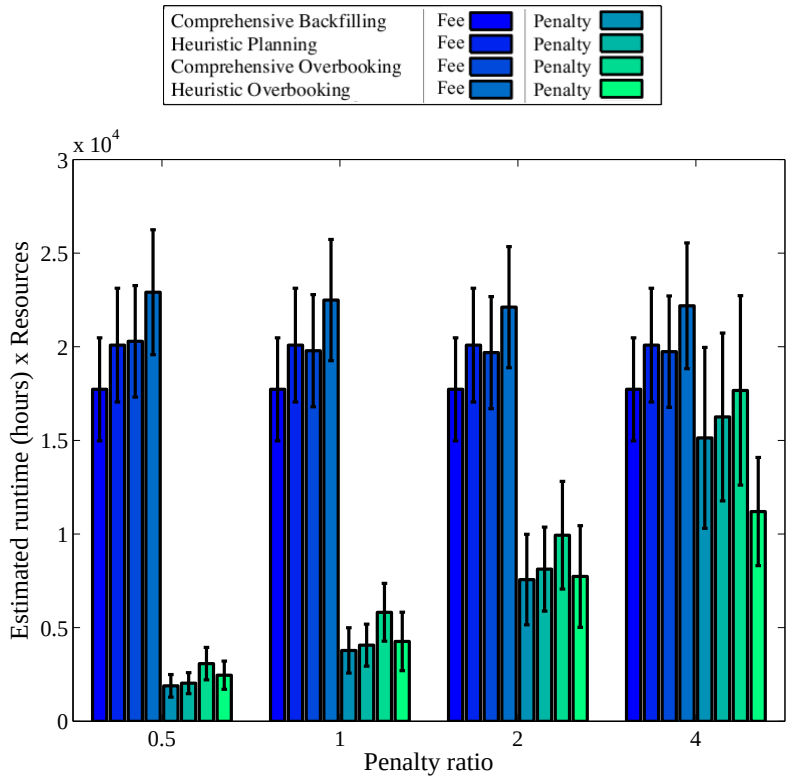


(a) The successful and failed jobs of the Arminius trace.

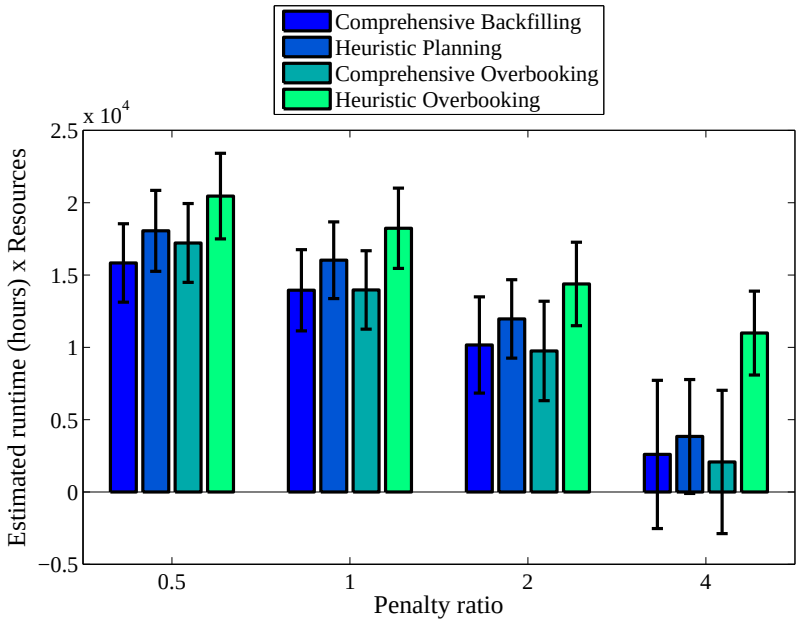


(b) The difference of successful and failed jobs.

Figure 7.12 Arminius: Profit and penalties with Risk acceptance test and application PDFs.



(a) The fees and penalties of the Arminius trace.



(b) The gain.

The confidence intervals in Figure 7.12(b) show that for a penalty ratio of 4 the results of the backfilling, heuristic planning, and comprehensive overbooking can lead to a loss of money. Only the heuristic overbooking works profitable.

Overall, the overbooking simulation based on the applications of Arminius had the worst result of all the simulation runs of the Arminius traces. At first glance, this means that the applications did not seem to be a fitting basis for overbooking. However, the simulation did not use statistics for single applications but for CCS workers. This workers often were application environments and complete sets of applications were assigned too them. The simulation shows that these applications were too different to be analyzed in groups build by CCS workers. Related work indicates that a runtime prediction for applications is beneficial [Gibb 97, Smit 98, Tsaf 07]. This means that applications are predictable and, therefore, can be used for overbooking. As a consequence, another way of grouping the applications should be evaluated.

7.4 Submission-User Analysis

The results of the simulation with PDF classes according to different users are presented in this section. This simulation was the last one in the evaluation of the Arminius traces. It was based on the statistical PDF classes according to different users that submitted many jobs. Figures 4.7 and 4.8 show the CDFs and PDFs for the top 10 users of Arminius in 2007. When new jobs from these users arrived, their personal statistics were used. For all other users, a general statistic was applied. The evaluation of the user statistics is discussed in Section 4.2.4.

7.4.1 PoF Acceptance Test

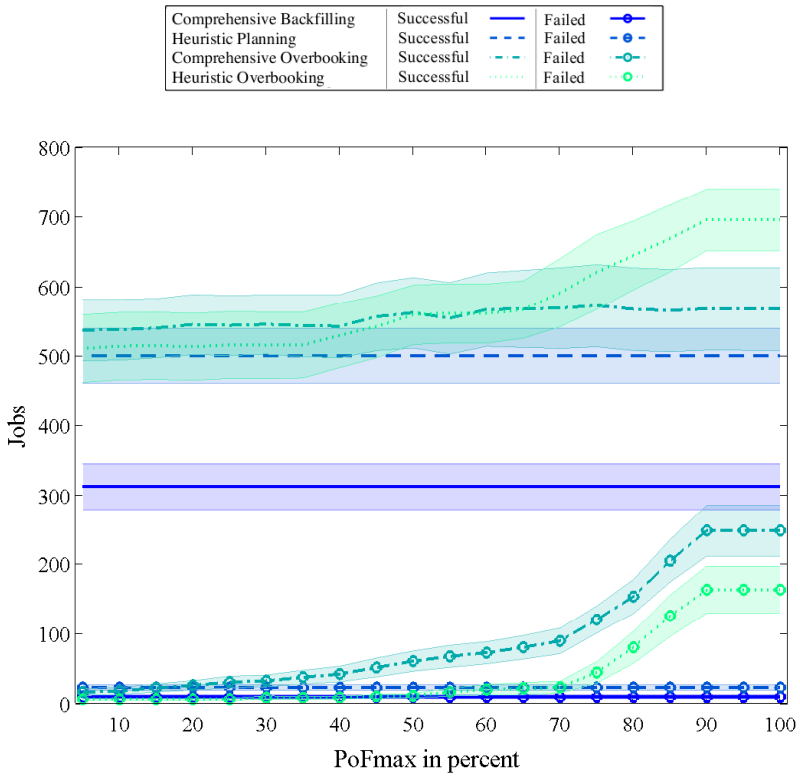
This section presents the results of the simulation with the acceptance test based on the PoF_{max} threshold. Table 7.7 summarizes the maximum gain of the User PDF based simulation, it contains the average results and the surplus of the overbooking strategies.

Table 7.7: *The PoF max gain with Arminius and the User PDF*

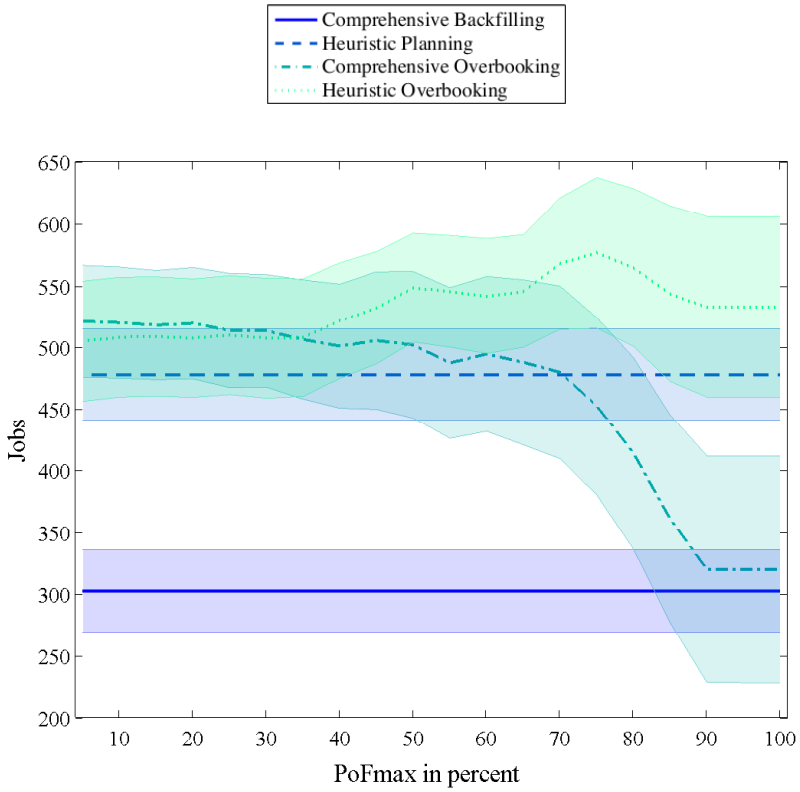
PoF simulation	VCs	additional gain	PoF_{max}
Comprehensive backfilling	13,944		
Heuristic planning	16,022		
Comprehensive overbooking	16,526	19%	0.35
Heuristic overbooking	20,520	28%	0.5

Figure 7.13 shows the successful and failed jobs in 7.13(a) and the differences in 7.13(b). The transparent areas are the 95% confidence intervals around the average results of 60 simulation runs. In this simulation, they are smaller than in the other test runs. This indicates that the use of user-based statistics might be

Figure 7.13 Arminius: Jobs with PoF acceptance test and user PDFs.



(a) The successful and failed jobs of the CTC trace.



(b) The difference of successful and failed jobs.

Table 7.8: *The Risk simulation results with Arminius and the user-based PDFs*

Risk simulation	Penalty Ratio			
User-based statistics	0.5	1	2	4
Comprehensive backfilling	15,835	13,944	10,160	2,593
Heuristic planning	18,054	16,022	11,960	3,834
Comprehensive overbooking	18,163	15,836	12,117	4,221
Comprehensive gain	15%	19%	17%	10%
Heuristic overbooking	22,080	19,324	16,829	9,540
Heuristic gain	22%	20%	41%	149%

the most useful one because the user-based analyses allowed the most predictable results.

The comprehensive overbooking had a slowly increasing amount of successful jobs over the entire PoF_{max} range. The number of jobs of the heuristic overbooking increased at a higher rate. At a PoF_{max} threshold of 0.9, the heuristic was saturated. The number of failed jobs of the comprehensive overbooking began to increase at a PoF_{max} of 0.25 and was saturated at a PoF_{max} of 0.9. For the heuristic, the number of failed jobs started to increase at a PoF_{max} of 0.7 and was saturated at a PoF_{max} of 0.9. This indicates that the cluster was fully utilized.

Figure 7.13 shows the resulting jobs. The confidence intervals are smaller than in the other simulations but they still overlap until a PoF_{max} of 0.8. The job difference of both overbooking strategies was the same until a PoF_{max} of 0.35. From there on, the heuristic's results were better compared to the comprehensive overbooking.

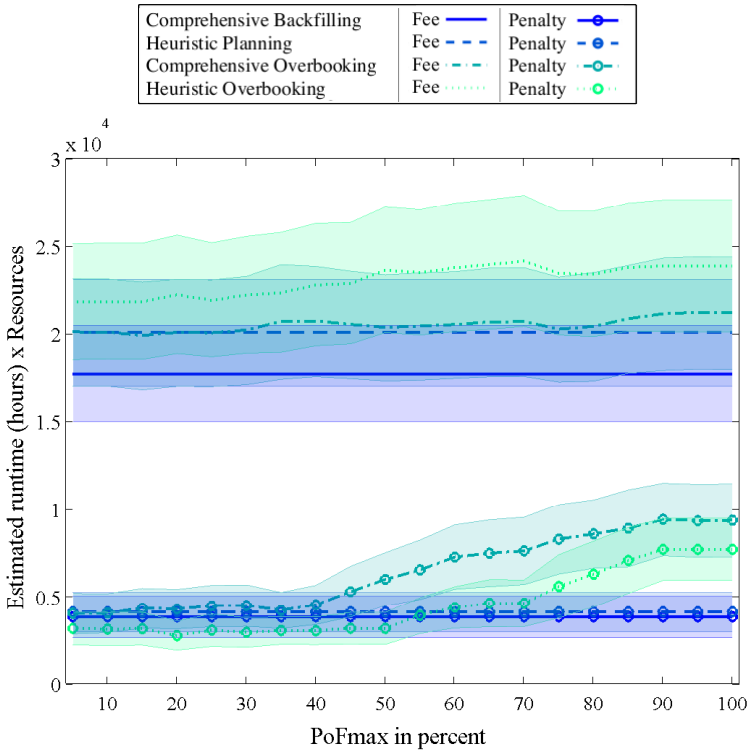
Figure 7.14 shows the resulting gain in resource hours or virtual coins. Figure 7.14(a) shows that the fees of all strategies overlap in their confidence intervals. The penalties of the comprehensive overbooking grew starting at a PoF_{max} of 0.4. The penalties of the heuristic overbooking grew starting at a PoF_{max} of 0.5.

Therefore, Figure 7.14(b) shows that the heuristic overbooking had its peak gain of 20,500 VCs with a PoF_{max} of 0.5, and the comprehensive overbooking had a peak gain of 16,500 VCs with a PoF_{max} of 0.35. For a PoF_{max} between 0.05 and 0.4, the result of the comprehensive overbooking was equal to the heuristic planning; thereafter, its result became worse. On average, the heuristic overbooking was better than the heuristic planning. From a PoF_{max} of 0.9 on, it had nearly the same result.

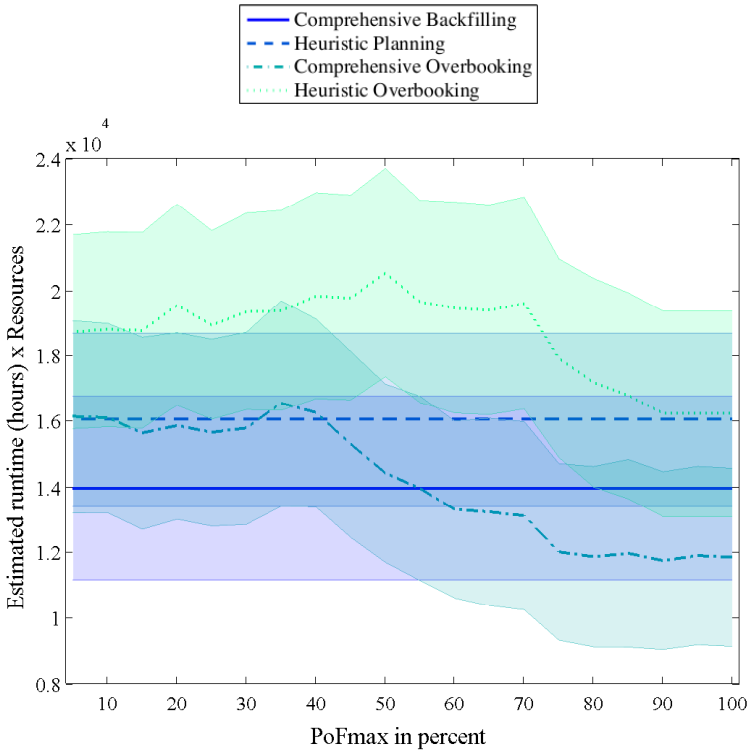
7.4.2 Risk Acceptance Test

Table 7.8 summarizes the results of the risk acceptance test of the last simulation. It was based on the statistical analysis of the user behaviors. The table shows the average results of the 60 test runs according to the penalty ratios and the surplus of the overbooking strategies.

Figure 7.14 Arminius: Profit and penalties with PoF acceptance test and user PDFs.

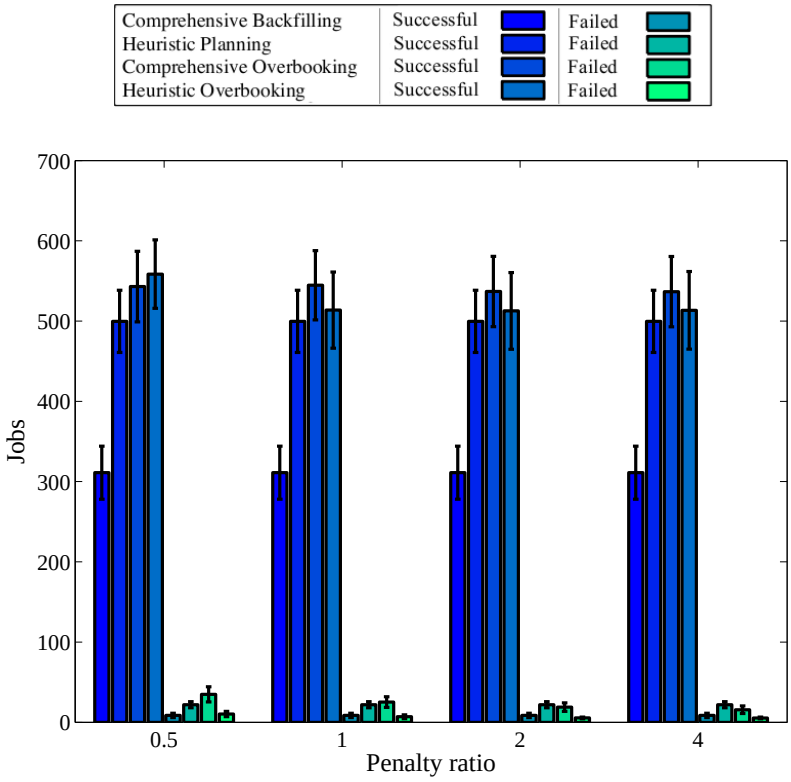


(a) The fees and penalties of the Arminius trace.

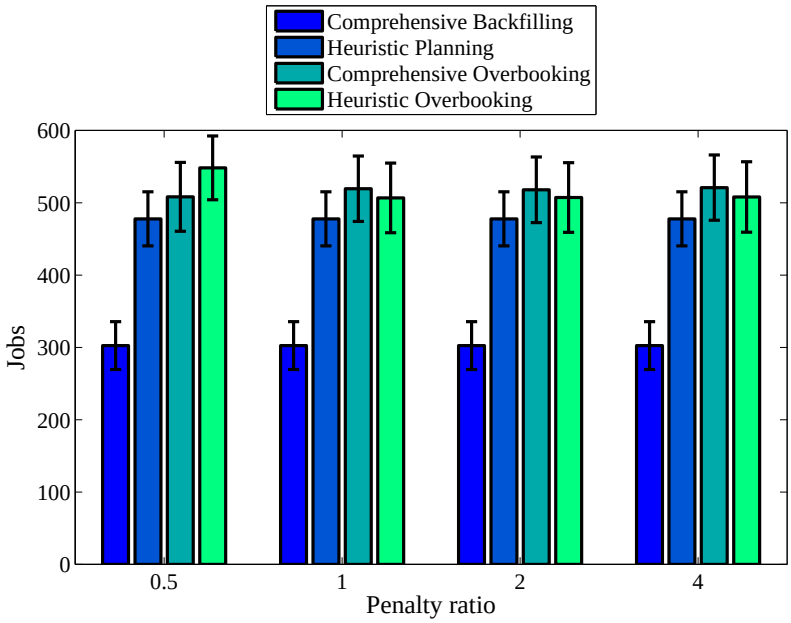


(b) The gain.

Figure 7.15 Arminius: Jobs with Risk acceptance test and user PDFs.

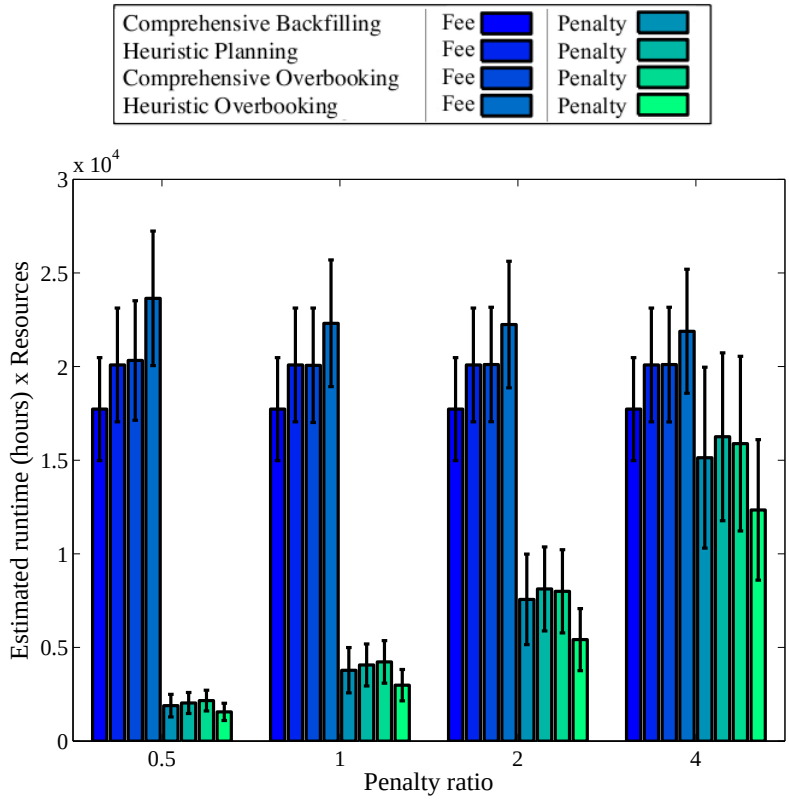


(a) The successful and failed jobs of the Arminius trace.

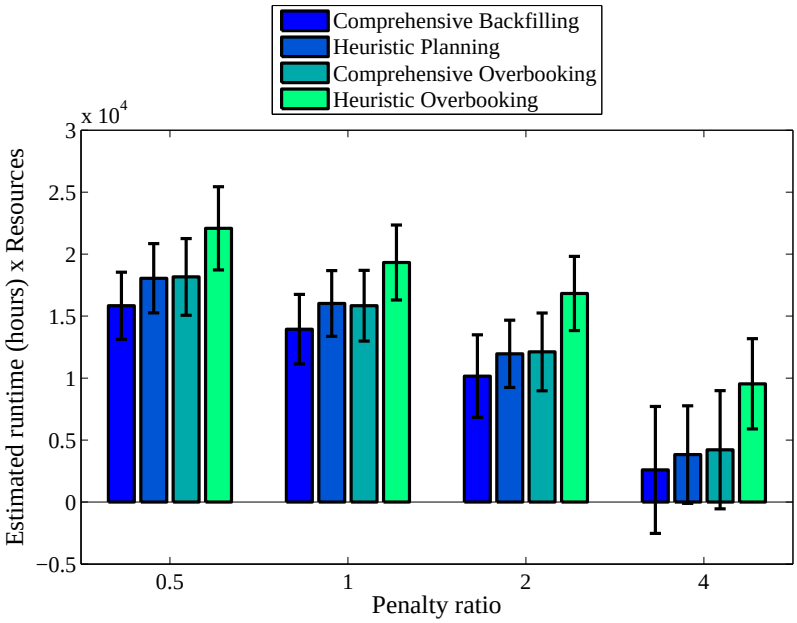


(b) The difference of successful and failed jobs.

Figure 7.16 Arminius: Profit and penalties with Risk acceptance test and user PDFs.



(a) The fees and penalties of the Arminius trace.



(b) The gain.

Figure 7.15 summarizes the resulting jobs. Figure 7.15(a) shows that even with low penalty ratio between 0.5 and 1, the number of failed jobs was low for the overbooking strategies. Accordingly with a higher penalty ratio, the number of failed jobs decreased only slightly. The number of successful jobs decreased at a higher rate. Figure 7.15(b) shows that especially the heuristic had a lower resulting job-difference for a higher penalty ratio.

Figure 7.16 summarizes the resulting gain. On the one hand with a higher penalty ratio, the heuristic overbooking had a smaller fee than the other strategies. However, it also looses less money, as illustrated in Figure 7.16(a).

The additional gain of the comprehensive approach was best with 15,836 VCs or a gain of 19% at a penalty ratio of 1. The heuristic overbooking was best at a penalty ratio of 4, with 9,540 VCs or a gain of 149%.

Like in all other simulations, the heuristic outperforms the comprehensive overbooking. This was due to a much more flexible, underlying heuristic planning strategy.

According to the confidence intervals, with a high penalty ratio of 4 the non-overbooking strategies and comprehensive overbooking can also have negative results. This was the last simulation run of the Arminius trace.

7.5 Summary

This section summarizes the most important facts of the simulations with the job traces of the Arminius cluster. It starts with the results of the PoF acceptance test and continues with a summary of the risk acceptance test. Finally, it presents the runtime of the simulations.

7.5.1 PoF Results

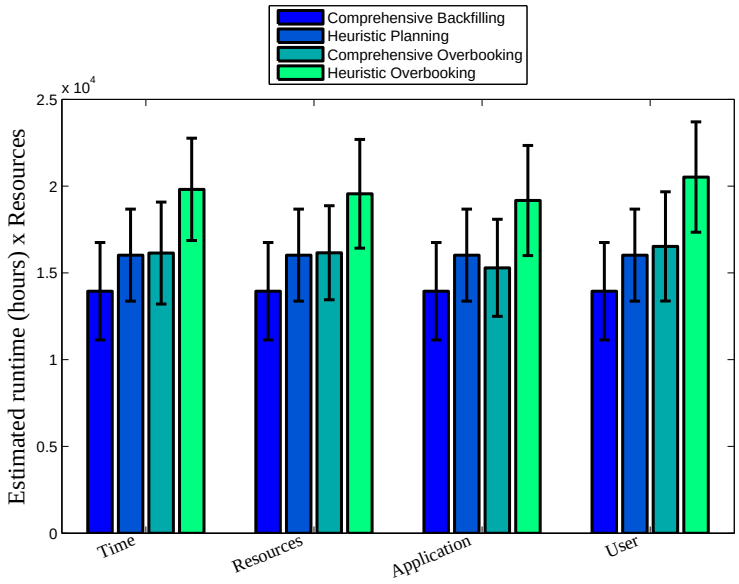
Figure 7.17 summarizes the maximum gain of the simulations based on the runtime-estimation, resources, application, and user analysis. The simulations confirmed that it is possible to increase the profit with overbooking.

Table 7.9: *Additional gain in % compared to the underlying non-overbooking strategy for the PoF acceptance test and the Arminius simulation.*

	Runtime	$PoF_{\max}$	Res.	$PoF_{\max}$	Appl.	$PoF_{\max}$	User	$PoF_{\max}$
Compreh.	16%	0.05	16%	0.15	10%	0.05	19%	0.35
Heuristic	24%	0.5	22%	0.4	20%	0.1	28%	0.5

For all simulations, the heuristic overbooking had the best results. The advantage of the heuristic overbooking lies in the heuristic planning that outperforms the backfilling. In three of the simulations, the comprehensive overbooking was better

Figure 7.17 Summary of the Arminius $PoF_{\max}$ threshold simulations.



(a) The maximum gain of the PoF acceptance test based simulations.

than the heuristic planning. In one case, the heuristic planning was better even without overbooking.

The impact of the different input statistics on the results of overbooking was measurable but low, about 4% additional gain was in between the single simulation results. The best result was possible with the statistical density functions based on the user's estimations with 20,520 VCs gain, followed by the runtime-estimation statistic with nearly 19,813 VCs gain, and the resource statistics with 19,556 VCs. The application statistics produced the worst results with 19,170 VCs at the most.

Except for the application simulation, the shapes of the bars were similar. This means that if the input statistics allow a certain estimation quality, the results of the overbooking were based on the scheduling strategy and the chosen $PoF_{\max}$ threshold. The maximum gain can be reached with an adequate threshold. Only if the input statistics are of low quality, the maximum additional gain cannot be reached. The optimal threshold allows the strategy to achieve the best result, and in such a case, the accuracy of the underlying statistics is not important.

7.5.2 Risk Results

Table 7.10 summarized the maximal additional gain in % compared to the underlying non-overbooking strategy with the given penalty ratios. Figure 7.18 shows the maximum gain of the simulations with the risk acceptance test in percent.

The results of the simulations of this test battery showed that the heuristic overbooking was more reliable than the comprehensive. The heuristic overbooking

Table 7.10: *Additional gain in % compared to the underlying non-overbooking strategy for the risk acceptance test and the Arminius simulation.*

	Runtime	ratio	Resources	ratio	Appl.	ratio	User	ratio
Compreh.	62%	4	21%	4	9%	0.5	19%	1
Heuristic	170%	4	167%	4	187%	4	149%	4

had, for all simulations, by far the best results with a penalty ratio of 4. The comprehensive overbooking had negative results with a penalty ratio of 2 and 4 (with the application PDFs), while the backfilling and the application statistics have a positive one.

This demonstrates that the application statistics were not usable for the risk acceptance test. Here, either an update or a change of the statistics is necessary, or the security factor has to be increased.

7.5.3 Runtime

The Section 3.4 provided already a brief theoretical runtime analysis of the overbooking algorithms that showed that the complexity classes would be the same $O(n^2)$ if the amount of jobs n would be dominant.

However, a user is not interested in the complexity class but in the fact that he gets a direct feedback for a job negotiation.

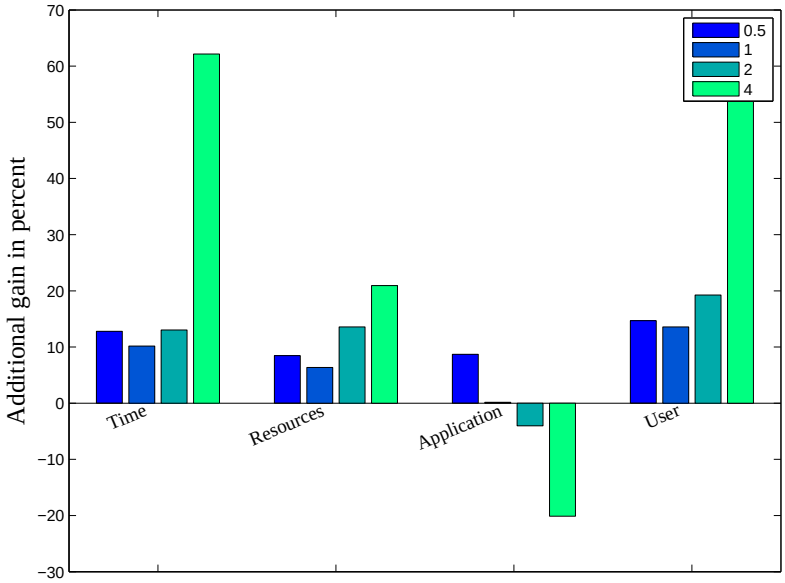
In this case, the heuristic seems to be faster than the comprehensive approach because the runtime only depends on the number of planned p in the system $O(p)$. The comprehensive overbooking approach has a runtime that is dependent on p, the number of resources of the cluster k, and the time needed for the FFT calculations. Following, the complexity would be $O(t \log(t) pk)$.

To evaluate whether this difference has an impact in practice, this simulation measured the runtime of the algorithms. Section 6.7.3 already discussed the runtime of the simulation with the PWA traces.

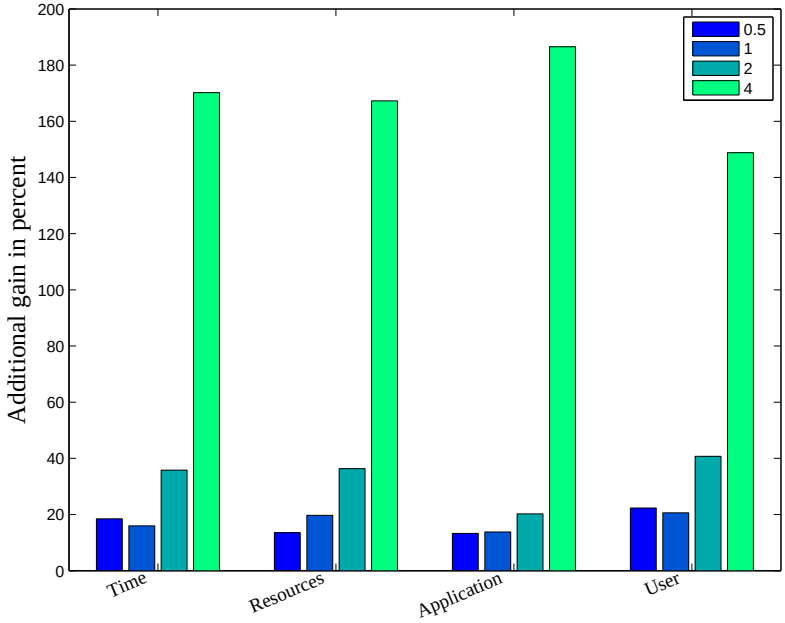
This section presents the results for the Arminius trace with the additional applied application and user statistics. In the summary of the simulation runtimes of Section 6.7.3, we saw different runtimes for different simulations. The question occurred whether or not the different runtimes only depended on the cluster size and the FFT transformations or also on the input statistics. A look at the runtimes of this set of simulations should reveal if this is the case since the environment was the same for all simulations in this chapter. Figures 7.19 and 7.20 show the average runtime of the simulations of the four approaches for the Arminius trace.

Compared to the PWA traces, the comprehensive overbooking still used more runtime than the other approaches even though the distance to the heuristic overbooking was smaller. The heuristic overbooking used about half the runtime

Figure 7.18 Summary of the Risk simulations and the Arminius traces.

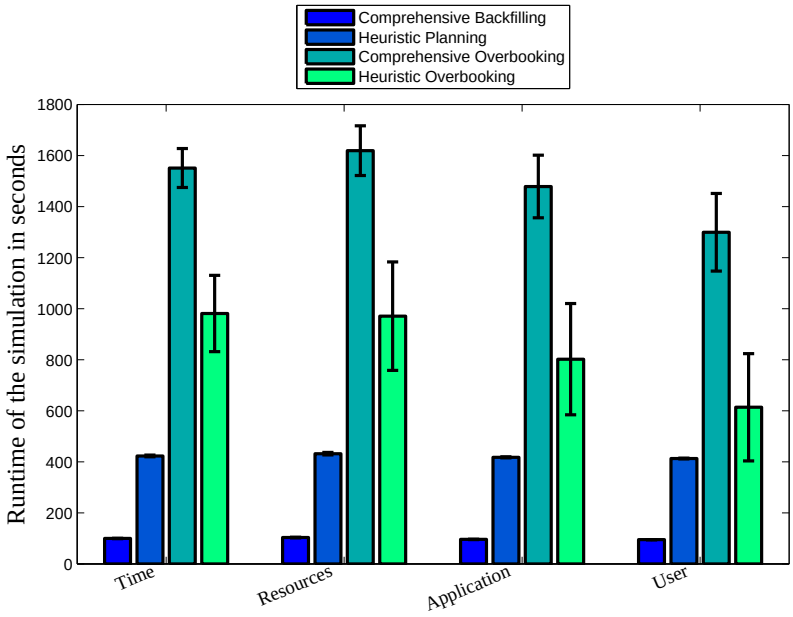


(a) The results of the comprehensive overbooking.



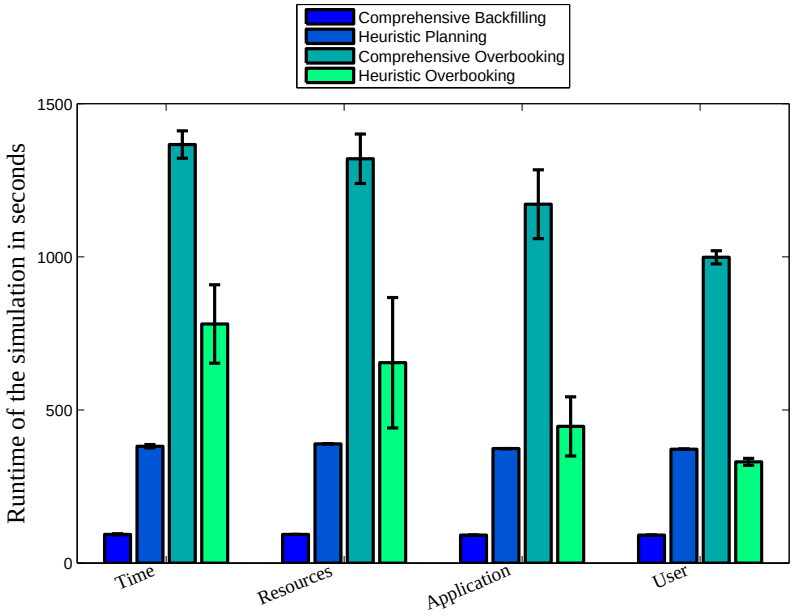
(b) The results of the heuristic overbooking.

Figure 7.19 Runtime of the PoF simulations.



(a) Runtime of the PoF simulations.

Figure 7.20 Runtime of the Risk simulations.



(a) Runtime of the Risk simulations.

of the comprehensive overbooking. Compared to the PWA traces, only the HPC2n trace had a similar distance.

The runtime of the four strategies was similar for all simulations in this chapter. Only the user-statistics based simulation was a bit faster than the average. Comparing this results to the runtime of the simulations shown in Section 6.7.3, it means that the input statistics had little influence on the runtime. The time of the FFT transformations and the size of the cluster system, and therewith the amount of resources and possible gaps, influenced the runtime most. The risk or the PoF acceptance test or the strategies were not very time consuming.

8 Conclusion

The conclusion first discusses the results of the evaluation of the overbooking approaches. The second part touches the remaining issues and new questions for future research that came up during this work.

Contents

8.1 Discussion 175

8.2 Future Work 178

The evaluation of the overbooking approaches in Chapters 6 and 7 demonstrates that the commercial selling of computing power can benefit from overbooking. One important prerequisite for a successful application of overbooking is the availability of reliable probability density functions. They describe the quality of the users' runtime estimations. Two strategies were developed and applied: one comprehensive overbooking approach that allows predictable results and one heuristic overbooking that proved to be faster, more flexible, and simply more profitable. The simulation used workload traces from the parallel workload archive and the PC^2's Arminius cluster. The evaluation of the algorithms showed that the profit of a provider increases and comparing all simulations, the heuristic overbooking has the best results. The additional gain depends on the load of the cluster and the quality of the runtime estimations. For the PoF acceptance test

- the minimum additional gain for the Arminius trace and the comprehensive overbooking with the worker statistics was 10%, and
- the maximum additional gain for the HPC2n trace and the comprehensive overbooking with resource statistics was 94%.

For the risk acceptance test with the HPC2n trace, the time statistics, and the penalty factor of 4, the additional gain was up to 234%. However, the risk acceptance test also shows that with bad statistics and a too offensive approach, the application of overbooking can lead to money loss. With the SDSC-SP2 trace, the input statistics, and a penalty ratio of 0.5, the simulation showed a loss of 61%.

8.1 Discussion

The simulation was supposed to evaluate whether the application of overbooking is profitable. Therefore, it was necessary to compare the comprehensive and heuristic overbooking strategy, find suitable statistics, and evaluate the runtime. In the following, these three points are discussed.

The *comprehensive overbooking* approach was designed to support a very accurate PoF calculation. It is able to calculate a job's PoF based on node specific failure predictions and the previously scheduled jobs. Therefore, the prediction is very accurate. The ability to accurately calculate PoFs supports a reliable overbooking process. The disadvantage of this approach is that the algorithm has to map the jobs on resources at SLA negotiation time. During the start time, a better choice of resources is often possible. As a consequence, the comprehensive strategy has many small gaps and a more scattered schedule. In the simulation, this disadvantage turns out to be an advantage when a higher risk or PoF was accepted for negotiation. In this case, only small jobs can be accepted. Loosing one of them does not lead to a high penalty.

Comparing the simulation results shows that the comprehensive overbooking is more trustworthy and reliable. In addition, if we compare the additional gain of the overbooking mechanisms to the underlying, not overbooking, strategies, the additional gain is higher for the comprehensive overbooking. This indicates that overbooking is more helpful for scheduling strategies that have a lower utilization. Obviously, one can more easily improve a result if there is more room for improvement. However, the simulations also show a very long runtime for the comprehensive overbooking.

To overcome this disadvantage, a *heuristic overbooking* was developed. It does not decide where the job should run at SLA negotiation time. The heuristic can start accepted jobs on any free resources and allows replanning due to the shorter runtime. The heuristic overbooking uses an overall resource count to calculate the probability of the job's success. On the one hand, this approach proved to be more flexible because a job can run on every node. On the other hand, this strategy does not remember the jobs that ran before. Therefore, the heuristic cannot include statistics of these jobs into the PoF calculation. As a result, the heuristic's PoF estimations are not as accurate as those of the comprehensive approach. However, ignoring the previous jobs is like assuming that they all use their full runtime. Nevertheless, it is very likely that the jobs will run shorter and the new job will likely start earlier and get more runtime. This probability is not included into the PoF calculation. Therefore, the resulting PoF estimation is in tendency too high and the algorithm tends to a PoF overestimation. However, to overestimate the PoF is better than underestimate it because it is better to be cautious and decline a few jobs more than to accept too many jobs that will fail.

In all simulation runs, the heuristic overbooking had a higher peak profit than the comprehensive overbooking, with the PoF acceptance test. The simulation shows that the heuristic has fewer but bigger gaps. Therefore, it is able to accept bigger jobs. Consequently, the heuristic overbooking has a higher profit than the comprehensive overbooking. The disadvantage of the heuristic is that the results can fluctuate. This can be seen in the simulation of the HPC2n and the SDSC-BLUE trace. They indicate that the risk to lose extremely big jobs was often misestimated. However, big jobs are those jobs that the provider wants to fulfill, not only because of money loss but also because of a reputation-loss.

In most simulation runs, the additional gain of the comprehensive overbooking is higher compared to the according, not overbooking strategies. However, if

backfilling and planning have the same results then the heuristic overbooking is better. The LANL simulation shows this. There, the backfilling and heuristic planning have the same results, and the heuristic overbooking is better than the comprehensive overbooking.

The *quality of the underlying statistics* was supposed to be the most important factor for the overbooking result. Therefore, the simulation implemented two different analyses for the PWA traces and four for the Arminius. However, the application of the different groups of statistics seems less aiding than expected. Due to the different PoF thresholds that were applied by the simulations, a similar peak performance was achieved with many different statistics. For the PWA simulation runs, the one half had better results with the time statistics, and the other half had better results with the resource statistics. For Arminius simulations, the user statistics allowed the best results and the worker the worst.

However, the quality of the statistics definitely has an impact on the PoF calculation. The shape of the PoF acceptance test curves illustrates this. The plots illustrate very different results. For supportive statistics, utilization and results of the PoF acceptance test simulation increase in the beginning because new jobs are successfully accepted; afterwards, they decrease because the acceptance test is too aggressive. The sample simulations with supportive statistics are:

- CTC runtime-estimation and resource statistics.
- SDSC-BLUE runtime-estimation and resource statistics.
- SDSC-DataStar runtime-estimation statistics.
- Arminius runtime-estimation, resource and user statistics.

In other cases, the statistics were not very helpful because the classes were badly chosen or the statistics were outdated. An outdated statistic can lead to a risk underestimation or risk overestimation. Statistics that lead to an underestimation of the PoF and the risk show directly decreasing plots. The examples for a simulation run with a risk underestimation are the simulation of the

- LANL runtime-estimation and resource statistics.
- SDSC-SP2 traces and runtime-estimation statistics.

Statistics that lead to a PoF and risk overestimation show curves with no changes in the resulting plots because no further jobs are accepted even with higher threshold due to the PoF overestimation. Examples for simulation runs with risk overestimation are:

- HPC2n runtime-estimation and resource statistics.
- SDSC-DataStar.

They never get a negative result.

However, the simulation shows that a monitoring and update process is important. Even if many simulations tend to be overestimated, better statistics allow better results. In practice, a monitoring and feedback system should allow steady updates. The PoF estimation shows that with a low threshold all overbooking approaches work well. If the overbooking approaches are carefully applied, overbooking

is profitable. In all simulations with a maximum $PoF_{\max}$ of 0.1, the results are positive. This also indicates that a very high security factor will probably lead to a profitable overbooking, for the risk acceptance tests.

The *runtime* of the algorithms is best for the not overbooking strategies. This fact is easily understandable because PoF estimations do not need to be calculated. The backfilling is the fastest strategy because it does not use rescheduling. The runtime of the comprehensive overbooking is the worst one due to the many convolutions that need to be done for the PoF and risk estimation process. The heuristic overbooking is faster. If comparing the strategies in the simulations, it becomes clear that the runtime strongly depends on the size of the simulated cluster. There are only small differences between the statistics of the same cluster.

Overall, the simulations showed that overbooking is applicable in a commercial environment. The comprehensive overbooking allows more predictable results, while the heuristic is faster and allows a higher profit. However, the underlying statistics should be updated to allow accurate PoF and risk estimations. If accurate statistics were available and the overbooking was applied carefully, it was successful in all simulation runs.

8.2 Future Work

The main task of a scientific work is to find answers to so far unsolved questions. Often on the way to uncover the results, new question arise. During the work of developing and evaluating the overbooking algorithms for academical or commercial resource providers, many new questions and ideas came up that are worth analyzing in further evaluations and may introduce new overbooking abilities. Some of them were already mentioned in the discussion of the evaluation, others are related to an adoption of the overbooking algorithms to other fields of application.

Future work considers:

- An improved monitoring of jobs,
- The use of commercial job traces for more appropriate statistical inputs,
- The application of advance reservations,
- The search for correlations between user estimation qualities and other job parameters,
- The application of system generated runtime predictions,
- The usage of checkpointing and migration, and
- The extension of overbooking to virtualization, and Cloud computing.

Monitoring of Jobs The first point that should be extended is the update process of the statistics. In practice, a monitoring and feedback system should allow steady updates. This should avoid outdated data. After a successful or failed job execution, the results should be included into the input statistics. A function should be developed that ages the monitoring information such that older moni-

toring information will have less impact on the statistics. At some point, outdated monitoring information should be removed from the statistics.

Application of Advance Reservations The actual simulation environments only deal with jobs that are submitted with SLAs and are free placeable in a plan in between their release and deadline. In practice however, users are tending to query for advance reservations. They have a defined beginning and defined end. For example, the users can take them for interactive use during the working hours of the day. These advance reservations cannot be overbooked or moved in the plan. The actual simulations avoid these kinds of jobs. However, they are used quite frequently and are a challenge because the overbooked jobs have to be placed around. Future simulations should reveal the impact of advance reservations on overbooking.

Use Other Sources of Job Traces to Create the Input Statistics The simulations presented in this dissertation base on the traces of scientific compute clusters. However additionally, the ability of overbooking for commercial providers should be evaluated. Therefore, an extended evaluation on the basis of SLA based job traces would be beneficial. The underlying assumption is that users tend to estimate better if they have to pay for the compute cycles. There already are commercial providers, like Oracle or Amazon, where customers buy resources, but the traces or even monitoring information is not available at this time.

Find Correlations of the Runtime Estimation and Other Parameters For future work, it would also be interesting to determine if there are other parameters that correlate with the user runtime estimation quality. Finding good correlations should allow the calculation of high quality PoF estimations. Beneath the monitoring issue, the quality of the estimations could be increased further with better correlations.

System Generated Runtime Estimations A further step towards a more profitable overbooking could be the creation of system generated runtime estimations for the jobs. These estimations should provide a lower bound for the job's runtime. This means that, regardless of the PoF threshold, only jobs that can be planned with this minimum runtime boundary would be accepted. This should prevent many SLA violations.

Checkpointing and Migration Fault tolerance mechanisms like checkpointing and migration can also lead to a less risky overbooking process. Checkpointing is the process of taking a transparent snapshot of a (parallel) job's memory, messages, registers, and CPU status, e.g. the program counter. The checkpoint can be stored in a file system available in the network [Hove 06, Hein 05b, Hein 05a, Hove 05]. Incomplete jobs could then be restarted in the next free gap of the schedule, before the job's completion time. Migration is the process of creating a job's checkpoint and transferring it to other nodes on the cluster, or even to another cluster, and restarting the job there from this checkpoint. With the checkpointing and migration mechanisms, jobs can be used in cooperation with planned FT mechanisms. They are then able to restart from checkpoint in case the gap is too short [Voss 06, Batt 07a, Batt 08b].

Virtualization Techniques Additionally, future work should examine how overbooking could be applied if the providers are using virtualization techniques. This step would close the gap to the cloud approaches and allow us to migrate jobs that took more runtime than the gap length allowed. Compute clouds already use virtualization to simplify data center management. Virtualization allows the live-migration of jobs and possesses other abilities, like the possibility to dynamically change the resources of a VM during runtime [Clar, Wald 02, Chis 07]. Additionally if enough other resources are available at the end of a job's time slot, the job can be moved to the free resources, and an SLA violation might be prevented. The costs of the virtualization due to migration time and of lost performance due to virtualization overhead could then be gauged with the additional gain by avoiding SLA violations. In addition, the possibility to overbook resources not only vertically but also horizontally enables other possible applications of overbooking. However, they also lead to a new complexity in estimating the benefits.

The mentioned subjects could be investigated and might provide better PoF and risk-prediction abilities, offer new means to avoid job kills, or allow us to exploit further dimensions for the overbooking of jobs.

Here, this dissertation comes to its end. This work had the task to reveal whether or not the application of overbooking is useful for SLA based scheduling systems. The simulations showed that the developed algorithms are promising. However, the quality of the underlying estimation statistics is important for the successful application of overbooking by a resource provider.

List of Figures

1.1 Overbooking example: How likely is it that the new job fits into the free time-slot? . . . 3
1.2 Comprehensive Overbooking: Calculating the probability by using node and other jobs' information. . . . 4
1.3 Heuristic Overbooking: Other jobs are not considered, only the free nodes are counted. . . . 5
1.4 AssessGrid architecture. . . . 7
1.5 The DGSI delegation scenarios. Grid schedulers from different domains can cooperate using activity and resource delegation. . . . 9

3.1 The PDF derived from all jobs in the examined cluster. . . . 38
3.2 The CDF for several time slots. . . . 39
3.3 Exemplary job schedule. . . . 40
3.4 The job can only be accepted using overbooking. . . . 42
3.5 The new job overlaps with the planned jobs on resources 1 and 3. 43
3.6 Heuristic Overbooking: The free and occupied nodes are counted for all planed jobs. . . . 48

4.1 The resulting PDFs from the runtime-estimation analysis. . . . 61
4.2 The resulting PDFs from the runtime-estimation analysis. . . . 62
4.3 The resulting PDFs from the statically analysis about requested nodes. . . . 63
4.4 The resulting CDFs from the statically analysis about requested nodes. . . . 64
4.5 The resulting PDFs from the statistical analysis of requested workers. . . . 65
4.6 The resulting CDFs from the statically analysis about requested workers. . . . 66
4.7 The resulting PDFs from the statistical analysis of the users. . . . 68
4.8 The resulting CDFs from the statistical analysis of the users. . . . 69

5.1 The simulation event processing. . . . 79
5.2 The process for the first fit job placing. . . . 82
5.3 The process for the best fit job placing. . . . 83

6.1 The runtime-estimation analysis of the CTC trace. . . . 87
6.2 The resource analysis of the CTC trace. . . . 88
6.3 CTC: Jobs with PoF acceptance test and runtime-estimation based PDFs. . . . 90

6.4 CTC: Profit and penalties with PoF acceptance test and runtime-estimation based PDFs. 91
6.5 CTC: Jobs with PoF acceptance test and resource based PDFs. . 94
6.6 CTC: Profit and penalties with PoF acceptance test and resource based PDFs. 95
6.7 CTC: Jobs with risk acceptance test and runtime-estimation based PDFs. 98
6.8 CTC: Profit and penalties with risk acceptance test and runtime-estimation based PDFs. 99
6.9 CTC: Jobs with risk acceptance test and resource based PDFs. . 101
6.10 CTC: Profit and penalties with risk acceptance test and resource based PDFs. 102
6.11 HPC2n: The CDF functions. 103
6.12 HPC2n: The gain for the PoF acceptance test. 105
6.13 HPC2n: The gain for the risk acceptance test. 107
6.14 LANL: The CDF functions. 109
6.15 LANL: The gain for the PoF acceptance test. 111
6.16 LANL: The gain for the risk acceptance test. 113
6.17 SDSC-BLUE: The CDF functions. 115
6.18 SDSC-BLUE: The gain for the PoF acceptance test. 117
6.19 SDSC-BLUE: The gain for the risk acceptance test. 119
6.20 SDSC-DataStar: The CDF functions. 121
6.21 SDSC-DataStar: The gain for the PoF acceptance test. 123
6.22 SDSC-DataStar: The gain for the risk acceptance test. 125
6.23 SDSC-SP2: The CDF functions. 127
6.24 SDSC-SP2: The gain for the PoF acceptance test. 129
6.25 SDSC-SP2: The gain for the risk acceptance test. 131
6.26 Summary of the $PoF_{\max}$ threshold simulations. 134
6.27 Summary of the Risk simulations with comprehensive Overbooking. 136
6.28 Summary of the Risk simulations with heuristic Overbooking. . 137
6.29 Runtime of the PoF simulations. 140
6.30 Runtime of the Risk simulations. 141

7.1 Arminius: Jobs with PoF acceptance test and runtime-estimation PDFs. 145
7.2 Arminius: Profit and penalties with PoF acceptance test and runtime-estimation PDFs. 147
7.3 Arminius: Jobs with Risk acceptance test and runtime-estimation PDFs. 148
7.4 Arminius: Profit and penalties with Risk acceptance test and runtime-estimation PDFs. 150
7.5 Arminius: Jobs with PoF acceptance test and resource based PDFs. 151
7.6 Arminius: Profit and penalties with PoF acceptance test and resource based PDFs. 153
7.7 Arminius: Jobs with Risk acceptance test and resource based PDFs. 154
7.8 Arminius: Profit and penalties with Risk acceptance test and resource based PDFs. 156

7.9 Arminius: Jobs with PoF acceptance test and application PDFs. . 158
7.10 Arminius: Profit and penalties with PoF acceptance test and application PDFs. 159
7.11 Arminius: Jobs with Risk acceptance test and application PDFs. 161
7.12 Arminius: Profit and penalties with Risk acceptance test and application PDFs. 162
7.13 Arminius: Jobs with PoF acceptance test and user PDFs. 164
7.14 Arminius: Profit and penalties with PoF acceptance test and user PDFs. 166
7.15 Arminius: Jobs with Risk acceptance test and user PDFs. 167
7.16 Arminius: Profit and penalties with Risk acceptance test and user PDFs. 168
7.17 Summary of the Arminius PoF_{max} threshold simulations. 170
7.18 Summary of the Risk simulations and the Arminius traces. . . . 172
7.19 Runtime of the PoF simulations. 173
7.20 Runtime of the Risk simulations. 173

List of Tables

2.1 SLA implementations by different projects 17
2.1 SLA implementations by different projects (con.) 18

3.1 Job attributes 34

4.1 Runtime of Arminius' jobs in 2007 in seconds 60
4.2 Runtime-estimation classes and the belonging jobs 61
4.3 Resource classes and the belonging jobs 63
4.4 Workers and the belonging jobs 66
4.5 Top 10 users and their jobs 67
4.6 Summary of the user survey. 70

5.1 Job Creation Model 74
5.2 The used traces for the PWA simulations 76

6.1 The PoF_{max} gain with the CTC trace. 88
6.2 The risk simulation results with CTC trace. 97
6.3 The PoF max gain with HPC2n trace. 104
6.4 The Risk simulation results with HPC2n trace. 106
6.5 The PoF max gain with LANL trace. 110
6.6 The Risk simulation results with LANL trace. 112
6.7 The PoF max gain with SDSC-BLUE trace. 116
6.8 The Risk simulation results with SDSC-BLUE trace. 118
6.9 The PoF max gain with SDSC-DataStar trace. 122
6.10 The Risk simulation results with SDSC-DataStar trace. 124
6.11 The PoF max gain with SDSC-SP2 trace. 128
6.12 The Risk simulation results with SDSC-SP2 trace. 130
6.13 Additional gain in % compared to the underlying non-overbooking strategy for the PoF acceptance test and the PWA traces. 133
6.14 Additional gain in % compared to the underlying non-overbooking strategy for the risk acceptance test and the PWA traces. 135

7.1 The PoF max gain with Arminius and the runtime-estimation PDF 144
7.2 The Risk simulation results with Arminius and the runtime-estimation PDFs 146
7.3 The PoF max gain with Arminius and the resource based PDF . 149
7.4 The Risk simulation results with Arminius and the resource based PDFs 155
7.5 The PoF max gain with Arminius and the application PDF . . . 157
7.6 The Risk simulation results with Arminius and the application PDFs 160
7.7 The PoF max gain with Arminius and the User PDF 163

7.8 The Risk simulation results with Arminius and the user-based PDFs 165
7.9 Additional gain in % compared to the underlying non-overbooking strategy for the PoF acceptance test and the Arminius simulation. 169
7.10 Additional gain in % compared to the underlying non-overbooking strategy for the risk acceptance test and the Arminius simulation. 171

Bibliography

[Amir 00] Y. Amir, B. Awerbuch, A. Barak, R. S. Borgstrom, and A. Keren. "An opportunity cost approach for job assignment in a scalable computing cluster". *IEEE Transactions on Parallel Distributed Systems*, Jul. 2000.

[Amir 98] Y. Amir, B. Awerbuch, and R. S. Borgstrom. "A cost-benefit framework for online management of a metacomputing system". In: *Proceedings of the 1st International Conference Information and Computational Economy, Charleston, SC*, Oct. 25, 1998.

[Andr 04a] A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj, and D. Snelling. "Open Issues in Grid Scheduling". *The UK e-Science Report UKeS-2004-03*, 2004.

[Andr 04b] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. "Web services agreement specification (WS-Agreement)". *Global Grid Forum (GGF)*, 2004.

[Asik 09] Ö. Aşık and E. Özcan. "Bidirectional best-fit heuristic for orthogonal rectangular strip packing". *Annals of Operations Research*, Vol. 172, No. 1, pp. 405–427, 2009.

[Asse 08] AssessGrid Team. "AssessGrid: Advanced Risk Assessment and Management for Trustable Grids". Project Website: `http://assessgrid.eu/`, 2008.

[Assu 09] M. de Assunção, A. di Costanzo, and R. Buyya. "Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters". In: *Proceedings of the 18th International ACM Symposium on High Performance Parallel and Distributed Computing (HPDC)*, pp. 141–150, ACM, 2009.

[Bake 80] B. Baker, E. Coffman Jr, and R. Rivest. "Orthogonal packings in two dimensions". *SIAM Journal on Computing*, Vol. 9, p. 846, 1980.

[Bake 81] B. Baker, D. Brown, and H. Katseff. "A 5/4 algorithm for two-dimensional packing". *Journal of Algorithms*, Vol. 2, No. 4, pp. 348–368, 1981.

[Bake 83] B. Baker and J. Schwarz. "Shelf Algorithms for Two-Dimensional Packing Problems". *SIAM Journal on Computing*, Vol. 12, p. 508, 1983.

[Bar 09] A. Bar-Noy, S. Guha, Y. Katz, J. Naor, B. Schieber, and H. Shachnai. "Throughput maximization of real-time scheduling with batching". *ACM Journal Transactions on Algorithms (TALG)*, Vol. 5, No. 2, pp. 1–17, 2009.

[Barh 03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the art of virtualization". *ACM SIGOPS Operating Systems Review*, Vol. 37, No. 5, pp. 164–177, 2003.

[Barz 07] C. Barz, T. Eickermann, M. Pilz, O. Wäldrich, L. Westphal, and W. Ziegler. "Co-allocating compute and network resources-bandwidth on demand in the VIOLA testbed". In: *Proceedings of the CoreGRID Symposium, CoreGRID Series, Towards Next Generation Grids*, pp. 193–202, 2007.

[Batt 07a] D. Battré, M. Hovestadt, O. Kao, A. Keller, and K. Voss. "Increasing fault tolerance by introducing virtual execution environments". In: *Proceedings of the 1. GI/ITG KuVS Fachgespräch Virtualisierung*, 2007.

[Batt 07b] D. Battré, O. Kao, and K. Voss. "Implementing WS-Agreement in a Globus Toolkit 4.0 Environment". In: *Proceedings of the Workshop Usage of Service Level Agreements in Grids in conjunction with The 8th IEEE International Conference on Grid Computing (GRID2007)*, 2007.

[Batt 08a] D. Battré, K. Djemame, I. Gourlay, M. Hovestadt, O. Kao, J. Padgett, K. Voss, and D. Warneke. "Assessgrid strategies for provider ranking mechanisms in risk–aware grid systems". *Grid Economics and Business Models*, pp. 226–233, 2008.

[Batt 08b] D. Battré, M. Hovestadt, O. Kao, A. Keller, and K. Voss. "Virtual Execution Environments for ensuring SLA-compliant Job Migration in Grids". In: *Proceedings of the IEEE International Conference on Services Computing SCC'08*, pp. 571–572, IEEE, 2008.

[Batt 08c] D. Battré, G. Birkenheuer, V. Deora, M. Hovestadt, O. Rana, and O. Wäldrich. "Guarantee and Penalty Clauses for Service Level Agreements". In: *Proceedings of the Cracow Grid Workshop (CGW)*, 2008.

[Batt 08d] D. Battré, G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voss. "Applying Risk Management to Support SLA Provisioning". In: *Proceedings of the Cracow Grid Workshop (CGW)*, 2008.

[Bayu 99] A. Bayucan, R. Henderson, C. Lesiak, B. Mann, T. Proett, and D. Tweten. "Portable batch system: External reference specification". Tech. Rep., MRJ Technology Solutions, 1999.

[Baza 10] Bazaar Team. "Bazaar". Project Website: `https://www.ce-egee.org/weblog/bazaar`, 2010.

[Benn 96] J. Bennett, G. Bohoris, E. Aspinwall, and R. Hall. "Risk analysis techniques and their application to software development". *European Journal of Operational Research*, Vol. 95, No. 3, pp. 467–475, 1996.

[Berk 87] J. Berkey and P. Wang. "Two-dimensional finite bin-packing algorithms". *Journal of the Operational Research Society*, pp. 423–429, 1987.

[Birk 06a] G. Birkenheuer, K. Djemame, I. Gourlay, O. Kao, J. Padgett, and K. Voss. "Using WS-Agreement for Risk Management in the Grid". In: *the First WS-Agreement Workshop (OGF18)*, 2006.

[Birk 06b] G. Birkenheuer, S. Döhre, M. Hovestadt, O. Kao, and K. Voss. "On Similarities of Grid Resources for Identifying Potential Migration Targets". In: *Proceedings of the Cracow Grid Workshop (CGW)*, 2006.

[Birk 07] G. Birkenheuer, P. Majlender, H. Nitsche, K. Voss, and E. Weber. "Gather and Prepare Monitoring Data for Estimating Resource Stability". In: *Proceedings of the Cracow Grid Workshop (CGW)*, 2007.

[Birk 08a] G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voss. "Overbooking in Planning Based Scheduling Systems". In: *Proceedings of the 2008 International Conference on Grid Computing and Applications (GCA)*, Las Vegas, Nevada, USA, May 2008.

[Birk 08b] G. Birkenheuer, A. Brinkmann, H. Dömer, S. Effert, C. Konersmann, O. Niehörster, and J. Simon. "Virtual Supercomputer for HPC and HTC". In: *Proceedings of the Gemeinsamer Workshop der GI/ITG Fachgruppen Betriebssysteme und KuVS: Virtualized IT infrastructures and their management*, Oct. 2008.

[Birk 09a] G. Birkenheuer, A. Brinkmann, and H. Karl. "The gain of overbooking". In: *Proceedings of the 14th Workshops on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Rome, Italy, May 2009.

[Birk 09b] G. Birkenheuer, A. Carlson, A. Fölling, M. Högqvist, A. Hoheisel, A. Papaspyrou, K. Rieger, B. Schott, and W. Ziegler. "Connecting Communities on the Meta-Scheduling Level: The DGSI Approach!". In: *Proceedings of the Cracow Grid Workshop (CGW)*, pp. 96 –103, 2009.

[Birk 10] G. Birkenheuer, A. Brinkmann, and H. Karl. "Risk Aware Overbooking for Commercial Grids". In: *Proceedings of the 15th Workshops on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Atlanta, USA, Apr. 23 2010.

[Birk 11a] G. Birkenheuer and A. Brinkmann. "Reservation-based overbooking for HPC clusters". In: *Proceedings of the IEEE International Conference on Cluster Computing (Cluster)*, Austin, USA, Sep. 2011.

[Birk 11b] G. Birkenheuer, A. Brinkmann, M. Högqvist, A. Papaspyrou, B. Schott, D. Sommerfeld, and W. Ziegler. "Infrastructure Federation Through Virtualized Delegation of Resources and Services". *Journal of Grid Computing*, pp. 1–23, 2011. 10.1007/s10723-011-9192-1.

[Bock 08] W. Bock, G. Macek, T. Oberndorfer, and R. Pumsenberger. "Praxisbuch ITIL". *Erfolgreiche Zertifizierung nach ISO*, Vol. 20000, No. 2, 2008.

[Boeh 89] B. Boehm. "Software risk management". *ESEC'89*, pp. 1–19, 1989.

[Bolt 10] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehörster, and A. Brinkmann. "Non-intrusive Virtualization Management Using libvirt". In: *Proceedings of the 2010 Conference on Design, Automation and Test in Europe (DATE)*, Dresden, Germany, Mar. 2010.

[Bred 98] J. Bredin, D. Kotz, and D. Rus. "Utility Driven Mobile-Agent Scheduling". Tech. Rep., CS-TR98-331, Dartmouth College, Hanover, NH, Oct. 3, 1998.

[Brin 10] A. Brinkmann, D. Battré, G. Birkenheuer, O. Kao, and K. Voss. "Risikomanagement für verteilte Umgebungen". *Forschungs Forum Paderborn (FFP)*, 2010.

[Broo 00] J. Brooke, M. Foster, S. Pickles, K. Taylor, and T. Hewitt. "Minigrids: Effective test-beds for grid application". In: *Proceedings of the 1st IEEE/ACM International Workshop GridComputing(GRID2000), Bangalore, India*, Dec. 17, 2000.

[Brow 80] D. Brown. "Improved BL Lower Bound". *Information Processing Letters*, Vol. 11, No. 1, pp. 37–39, 1980.

[Burk 04] E. Burke, G. Kendall, and G. Whitwell. "A new placement heuristic for the orthogonal stock-cutting problem". *Operations Research*, pp. 655–671, 2004.

[Buyy 00] R. Buyya, D. Abramson, and J. Giddy. "Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid". In: *Proceedings of the 4th International Conference High Performance Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China*, p. 283, May 2000.

[Buyy 02] R. Buyya and M. Murshed. "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing". *Journal of Concurrency and Computation: Practic Experience (CCPE), vol. 14*, 2002.

[Buyy 04] R. Buyya and S. Venugopal. "The gridbus toolkit for service oriented grid and utility computing: An overview and status report". In: *Proceedings of the 1st IEEE International Workshop Grid Economics and Business Models (GECON 2004), Seoul, Korea,*

Apr. 23, 2004.

[Buyy 05] R. Buyya, D. Abramson, and S. Venugopal. "The grid economy". In: *Proceedings of the IEEE*, pp. 698–714, IEEE, 2005.

[Carl 08] C. Carlsson, O. Weissmann, A. Bauer, I. Gourlay, J. Kinnunen, M. Heikkilä, and J. Mezier. "Advanced Risk Assessment". Tech. Rep., 2008.

[Carl 87] J. Carlier. "Scheduling jobs with release dates and tails on identical machines to minimize the makespan". *European Journal of Operational Research*, Vol. 29, No. 3, pp. 298–306, 1987.

[Caro 04] E. Caron, P. Chouhan, and F. Desprez. "Deadline scheduling with priority for client-server systems on the grid". In: *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pp. 410–414, IEEE Computer Society, 2004.

[Chan 08] W. Chan, F. Chin, D. Ye, G. Zhang, and Y. Zhang. "On-line scheduling of parallel jobs on two machines". *Journal of Discrete Algorithms*, Vol. 6, No. 1, pp. 3–10, 2008.

[Chaz 06] B. Chazelle. "The bottomn-left bin-packing heuristic: An efficient implementation". *Journal of IEEE Transactions on Computers*, Vol. 100, No. 8, pp. 697–707, 2006.

[Chen 04] M. Chen, Y. Wu, G. Yang, and X. Liu. "Efficiently Rationing Resources for Grid and P2P Computing". In: *Proceedings of the IFIP International Conference Network and Parallel Computing (NPC)*, pp. 133–136, 2004.

[Chis 07] D. Chisnall. *The Definitive Guide to the Xen Hypervisor*. Prentice Hall, Upper Saddle River, NJ, USA, 2007.

[Chun 82] F. Chung, M. Garey, and D. Johnson. "On packing two-dimensional bins". *SIAM Journal on Algebraic and Discrete Methods*, Vol. 3, No. 1, pp. 66–76, 1982.

[Clar] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. "Live Migration of Virtual Machine".

[Coff 80] E. Coffman Jr, M. Garey, D. Johnson, and R. Tarjan. "Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms". *SIAM Journal on Computing*, Vol. 9, p. 808, 1980.

[Comm 91] R. Comment and G. Jarrell. "The relative signalling power of Dutch-auction and fixed-price self-tender offers and open-market share repurchases". *Journal of Finance*, pp. 1243–1271, 1991.

[Corm 01] T. Cormen, L. C.E., and R. R.L. *Introduction to algorithms*. The MIT press, 2001.

[Crit 08] Critix Systems. "Whitepaper: Clearly better virtualization with Citrix XenServer". Project Website: `http://www.citrix.de/produkte/schnellsuche/xenserver/list/white-papers/`, 2008.

[Csir 97] J. Csirik and G. Woeginger. "Shelf algorithms for on-line strip packing* 1". *Information Processing Letters*, Vol. 63, No. 4, pp. 171–175, 1997.

[Csir 98] J. Csirik and G. Woeginger. "On-line packing and covering problems". *Lecture Notes in Computer Science, Volume 1442/1998, 147-177, DOI: 10.1007/BFb0029568*, 1998.

[Czaj 02] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. "SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems". In: *Proceedings of Job scheduling strategies for parallel processing (JSSPP)*, pp. 153–183, Springer, 2002.

[D An 06] F. D' Andria, J. Martrat, G. Laria, P. Ritrovato, and S. Wesner. "An enhanced strategy for SLA management in the business context of new mobile dynamic VO". In: *Proceedings of the 4th eChallenges Conference, Barcelona, Spain*, 2006.

[Dean 08] J. Dean and S. Ghemawat. "MapReduce: Simplified data processing on large clusters". *Communications of the ACM*, Vol. 51, No. 1, pp. 107–113, 2008.

[Dimi 07] T. Dimitrakos. "Design patterns for SOA and Grid". *BEinGRID Meta-Deliverable AC1s*, Vol. 1, 2007.

[Djem 06] K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voss. "Introducing Risk Management into the Grid". In: *E-Science '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, p. 28, IEEE Computer Society, Washington, DC, USA, 2006.

[Djem 08] K. Djemame, J. Padgett, I. Gourlay, K. Voss, D. Battré, and O. Kao. "Economically Enhanced Risk-aware Grid SLA Management". In: *Proceedings of eChallenges e-2008 Conference, Stockolm, Sweden*, 2008.

[Down 97] A. Downey. "Using Queue Time Predictions for Processor Allocation". In: *Proceedings of Job Scheduling Strategies for Parallel Processing: IPPS'97 Workshop, Geneva, Switzerland, Apr. 5*, 1997.

[Down 99] A. Downey and D. Feitelson. "The elusive goal of workload characterization". *ACM SIGMETRICS Performance Evaluation Review*, Vol. 26, No. 4, pp. 14–29, 1999.

[Elmr 05] E. Elmroth and J. Tordsson. "An interoperable, standards-based grid resource broker and job submission service". In: *Proceedings of the First International Conference on e-Science and Grid Computing*, pp. 9–pp, IEEE, 2005.

[Eyma 07] T. Eymann, W. Streitberger, and S. Hudert. "CATNETS–Open Market Approaches for Self-organizing Grid Resource Allocation". *Grid Economics and Business Models*, pp. 176–181, 2007.

[Fahr 07] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H. Truong, *et al.* "Askalon: A development and grid computing environment for scientific workflows". *Workflows for e-Science*, pp. 450–471, 2007.

[Fall 06] N. Fallenbeck, H. Picht, M. Smith, and B. Freisleben. "Xen and the art of cluster scheduling". In: *Proceedings of the 1st International Workshop on Virtualization Technology in Distributed Computing (VTDC)*, pp. 4–4, Tampa, Florida, USA, 2006.

[Feit 05] D. Feitelson and A. Mu'alem. "On the definition of on-line in job scheduling problems". *ACM SIGACT News*, Vol. 36, No. 1, pp. 122–131, 2005.

[Feit 10] D. Feitelson *et al.* "Parallel Workloads Archive". Project Website: `http://www.cs.huji.ac.il/labs/parallel/workload/`, 2010.

[Feit 11] D. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation, Theory and practice in parallel job scheduling*. 2011.

[Feit 95] D. Feitelson and B. Nitzberg. "Job Characteristics of a Production Parallel Scientific Workload on the NASA Ames iPSC/860". In: *Proceedings of Job Scheduling Strategies for Parallel Processing: IPPS'95 Workshop, Santa Barbara, CA, USA, Apr. 25*, 1995.

[Feit 97a] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong. "Theory and practice in parallel job scheduling". In: *Proceedings of Job Scheduling Strategies for Parallel Processing*, pp. 1–34, Springer, 1997.

[Feit 97b] D. Feitelson and M. Jette. "Improved Utilization and Responsiveness with Gang Scheduling". In: *Proceedings of Job Scheduling Strategies for Parallel Processing: IPPS'97 Workshop, Geneva, Switzerland, Apr. 5,*, 1997.

[Feit 98] D. Feitelson and A. Weil. "Utilization and Predictability in Scheduling the IBM SP2 with Backfilling". In: *Proceedings of the 12th International Parallel Processing Symposium*, Jan 1998.

[Fost 08] I. Foster, Y. Zhao, I. Raicu, and S. Lu. "Cloud computing and grid computing 360-degree compared". In: *Grid Computing Environments Workshop, 2008. GCE'08*, pp. 1–10, IEEE, 2008.

[Gare 79] M. Garey and D. Johnson. *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences*. WH Freeman and Company, San Francisco, Calif, 1979.

[Gibb 97] R. Gibbons. "A Historical Application Profiler for Use by Parallel Schedulers". In: *Proceedings of Job Scheduling Strategies for Parallel Processing: IPPS'97 Workshop, Geneva, Switzerland, Apr. 5*, 1997.

[Goel 85] A. Goel. “Software reliability models: Assumptions, limitations, and applicability”. *IEEE Transactions on Software Engineering*, No. 12, pp. 1411–1423, 1985.

[Grim 07] C. Grimme, T. Langhammer, A. Papaspyrou, and F. Schintke. “Negotiation-based choreography of data-intensive applications in the C3Grid project”. In: *Proceedings of the German e-Science Conference*, 2007.

[Grim 09] C. Grimme and A. Papaspyrou. “Cooperative negotiation and scheduling of scientific workflows in the collaborative climate community data and processing grid”. *Future Generation Computer Systems*, Vol. 25, No. 3, pp. 301–307, 2009.

[Grit 06] L. Grit, D. Irwin, V. Marupadi, P. Shivam, A. Yumerefendi, J. Chase, and J. Albrecht. “Harnessing virtual machine resource control for job management”. In: *Proceedings of the 1st International Workshop on Virtualization Technology in Distributed Computing (VTDC)*, Tampa, Florida, USA, 2006.

[Hams 00] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. “Evaluation of job-scheduling strategies for grid computing”. In: *Proceedings of the Grid Computing (GRID 2000)*, pp. 191–202, Springer, 2000.

[Hass 07] P. Hasselmeyer, H. Mersch, B. Koller, H. Quyen, L. Schubert, and P. Wieder. “Implementing an SLA negotiation framework”. In: *Proceedings of the eChallenges e-2007 Conference, Hague, Netherlands*, 2007.

[Heat 01] T. Heath, R. Martin, and T. Nguyen. “The shape of failure”. In: *Proceedings of the First Workshop on Evaluating and Architecting System dependabilitY (EASY)*, 2001.

[Hein 05a] F. Heine, M. Hovestadt, O. Kao, and A. Keller. “Provision of fault tolerance with grid-enabled and SLA-aware resource management systems”. In: *Proceedings of the Parallel Computing Conference*, p. 158, 2005.

[Hein 05b] F. Heine, M. Hovestadt, O. Kao, and A. Keller. “SLA-aware job migration in grid environments”. *Advances in Parallel Computing*, Vol. 14, pp. 185–201, 2005.

[Heis 98] G. Heiser, F. Lam, and S. Russell. “Resource management in the mungi single-address-space operating system”. In: *Proceedings of the Ausatralasian Computer Science Conference, Perth, Australia*, 1998.

[Hohe 06] A. Hoheisel. “Grid Workflow Execution Service-Dynamic and interactive execution and visualization of distributed workflows”. In: *Proceedings of the Cracow Grid Workshop*, pp. 13–24, 2006.

[Hoin 09] A. Höing, G. Scherp, S. Gudenkauf, D. Meister, and A. Brinkmann. “An Orchestration as a Service Infrastructure Using Grid Technolo-

gies and WS-BPEL". *Service-Oriented Computing*, pp. 301–315, 2009.

[Hosk 06] M. Hoskins. "User-mode Linux". *Linux Journal*, Vol. 145, 2006.

[Hove 03] M. Hovestadt, O. Kao, A. Keller, and A. Streit. "Scheduling in HPC Resource Management Systems: Queuing vs. Planning". In: *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, pp. 1–20, Seattle, WA, USA, Jun. 24 2003.

[Hove 05] M. Hovestadt. "Fault tolerance mechanisms for SLA-aware resource management". In: *Proceedings of the 11th International Conference on Parallel and Distributed Systems*, pp. 458–462, IEEE, 2005.

[Hove 06] M. Hovestadt. "Operation of an SLA-aware Grid Fabric". *Journal of Computer Science*, Vol. 2, No. 6, pp. 550–557, 2006.

[Hoyl 88] S. Høyland. "Bin-packing in 1.5 dimension". *SWAT 88*, pp. 129–137, 1988.

[HPC4 08] HPC4U Team. "HPC4U: Introducing Quality of Service for Grids". Project Website: `http://www.hpc4u.org/`, 2008.

[Huri 08a] J. Hurink and J. Paulus. "Online algorithm for parallel job scheduling and strip packing". *Approximation and Online Algorithms*, pp. 67–74, 2008.

[Huri 08b] J. Hurink and J. Paulus. "Online scheduling of parallel jobs on two machines is 2-competitive". *Operations research letters*, Vol. 36, No. 1, pp. 51–56, 2008.

[Iosu 07] A. Iosup, M. Jan, O. Sonmez, and D. Epema. "On the dynamic resource availability in grids". In: *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pp. 26–33, IEEE Computer Society, 2007.

[Jack 01] D. Jackson, Q. Snell, and M. Clement. "Core algorithms of the Maui scheduler". In: *Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP)*, pp. 87–102, Springer, 2001.

[Jako 96] S. Jakobs. "On genetic algorithms for the packing of polygons". *European Journal of Operational Research*, Vol. 88, No. 1, pp. 165–181, 1996.

[Jans 10] K. Jansen, S. Kratsch, D. Marx, and I. Schlotter. "Bin packing with fixed number of bins revisited". In: *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2010) Bergen, Norway, 21-23 Jun.*, 2010.

[Joha 06] B. Johannes. "Scheduling parallel jobs to minimize the makespan". *Journal of Scheduling*, Vol. 9, No. 5, pp. 433–452, 2006.

[Joha 08] M. Johanssen. "Update on System Virtualization Management". In: *Proceedings of the 2nd International DMTF Academic Al-*

liance Workshop on Systems and Virtualization Management (SVM), pp. 125 – 134, Munich, Germany, Oct. 2008.

[John 73] D. Johnson. "Near-optimal bin packing algorithms". *dissertation*, 1973.

[John 74a] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham. "Worst-case performance bounds for simple one-dimensional packing algorithms". *SIAM Journal on Computing*, Vol. 3, p. 299, 1974.

[John 74b] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham. "Worst-case performance bounds for simple one-dimensional packing algorithms". *SIAM Journal on Computing*, Vol. 3, p. 299, 1974.

[Kacs 05] P. Kacsuk and G. Sipos. "Multi-grid, multi-user workflows in the P-GRADE grid portal". *Journal of Grid Computing*, Vol. 3, No. 3, pp. 221–238, 2005.

[Kacs 07] P. Kacsuk, Z. Farkas, and G. Hermann. "Workflow-level parameter study support for production grids". *Computational Science and Its Applications–ICCSA 2007*, pp. 872–885, 2007.

[Kacs 08] P. Kacsuk, T. Kiss, and G. Sipos. "Solving the grid interoperability problem by P-GRADE portal at workflow level". *Future Generation Computer Systems*, Vol. 24, No. 7, pp. 744–751, 2008.

[Kaly 99] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer. "Failure data analysis of a LAN of Windows NT based computers". In: *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, pp. 178–187, IEEE, 1999.

[Kann 01] S. Kannan, M. Roberts, P. Mayes, D. Brelsford, and J. Skovira. "Workload management with loadleveler". *IBM Redbooks*, Vol. 2, p. 2, 2001.

[Kans 02] K. Kansala. "Integrating risk assessment with cost estimation". *Software, IEEE*, Vol. 14, No. 3, pp. 61–67, 2002.

[Karm 08] N. Karmarkar and R. Karp. "An efficient approximation scheme for the one-dimensional bin-packing problem". In: *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science SFCS'08.*, pp. 312–320, IEEE, 2008.

[Keah 05] K. Keahey, I. Foster, T. Freeman, and X. Zhang. "Virtual workspaces: Achieving quality of service and quality of life in the Grid". *Scientific Programming*, Vol. 13, No. 4, pp. 265–275, 2005.

[Kell 01] A. Keller and A. Reinefeld. "Anatomy of a Resource Management System for HPC Clusters". *Annual Review of Scalable Computing*, Vol. 3, pp. 1–31, 2001.

[Keny 00] C. Kenyon and E. Rémila. "A near-optimal solution to a two-dimensional cutting stock problem". *Mathematics of Operations*

Research, Vol. 25, No. 4, pp. 645–656, 2000.

[Keny 02] C. Kenyon and E. Remila. "Approximate strip packing". In: *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pp. 31–36, IEEE, 2002.

[Kivi 07] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. "KVM: the Linux Virtual Machine Monitor". In: *Proceedings of the 2007 Ottawa Linux Symposium*, pp. 225 –230, Ottawa, Ontario, Canada, June 2007.

[Kiya 06] N. Kiyanclar, G. Koenig, and W. Yurcik. "Maestro-VC: A paravirtualized execution environment for secure on-demand cluster computing". In: *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid) Workshops*, Singapore, Malaysia, 2006.

[Koen 04] J. Koenig. "JBoss jBPM white paper". Project website `www.JBoss.com`, 2004.

[Krue 94] P. Krueger, T. Lai, and V. Dixit-Radiya. "Job scheduling is more important than processor allocation for hypercube computers". *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 5, pp. 488–497, 1994.

[La V 98] W. de La Vega and V. Zissimopoulos. "An approximation scheme for strip packing of rectangles with bounded dimensions". *Discrete Applied Mathematics*, Vol. 82, No. 1-3, pp. 93–101, 1998.

[Laad 10] O. Laadan and S. Hallyn. "Linux-CR: Transparent Application Checkpoint-Restart in Linux". In: *Proceedings of the Linux Symposium 2010*, 2010.

[Lali 00] S. Lalis and A. Karipidis. "An open market-based framework for distributed computing over the internet". In: *Proceedings of the 1st IEEE/ACM International Workshop Grid Computing (GRID 2000), Bangalore, India*, 2000.

[Lee 93] I. Lee, D. Tang, R. Iyer, and M. Hsueh. "Measurement-based evaluation of operating system fault tolerance". *IEEE Transactions on Reliability*, Vol. 42, No. 2, pp. 238–249, 1993.

[Leve 86] N. Leveson. "Software safety: Why, what, and how". *ACM Computing Surveys (CSUR)*, Vol. 18, No. 2, p. 163, 1986.

[Li 97] C. Li and C. Lee. "Scheduling with agreeable release times and due dates on a batch processing machine". *European Journal of Operational Research*, Vol. 96, No. 3, pp. 564–569, 1997.

[Libe 78] V. Liberman and U. Yechiali. "On the Hotel Overbooking Problem-An Inventory System with Stochastic Cancellations". *Management Science*, Vol. 24, No. 11, pp. 1117–1126, 1978.

[Lich 96] S. Lichtenstein. "Factors in the selection of a risk assessment method". *Information Management & Computer Security*, Vol. 4, No. 4, pp. 20–25, 1996.

[Litz 02] M. Litzkow, M. Livny, and M. Mutka. "Condor-a hunter of idle workstations". In: *Proceedings on the 8th International Conference on Distributed Computing Systems*, pp. 104–111, IEEE, 2002.

[Lo 97] V. Lo, K. Windisch, W. Liu, and B. Nitzberg. "Noncontiguous processor allocation algorithms for mesh-connected multicomputers". *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 7, pp. 712–726, 1997.

[Lodi 02a] A. Lodi, S. Martello, and M. Monaci. "Two-dimensional packing problems: A survey". *European Journal of Operational Research*, Vol. 141, No. 2, pp. 241–252, 2002.

[Lodi 02b] A. Lodi, S. Martello, and D. Vigo. "Recent advances on two-dimensional bin packing problems". *Discrete Applied Mathematics*, Vol. 123, No. 1-3, pp. 379–396, 2002.

[Los 11] Los Alamos National Laboratory. "The Los Alamos National Laboratory Failure Data Archive". Project Website: `http://institutes.lanl.gov/data/fdata/`, 2011.

[Lowe 09] S. Lowe. *Mastering VMware VSphere 4. For Dummies*, John Wiley & Sons, 2009.

[Ludw 03] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck. "Web service level agreement (WSLA) language specification". Tech. Rep., IBM Corporation, 2003.

[Majl 06] P. Majlender, C. Carlsson, F. Tétard, M. Heikkilä, I. Gourlay, K. Voss, and G. Birkenheuer. "Risk Management Evaluation". Tech. Rep., 2006.

[McKe 07] P. McKee, S. Taylor, M. Surridge, R. Lowe, and C. Ragusa. "Strategies for the Service Market Place". *Grid Economics and Business Models*, p. 58, 2007.

[Mell 09] P. Mell and T. Grance. "The NIST definition of cloud computing". *National Institute of Standards and Technology*, Vol. 53, No. 6, p. 50, 2009.

[Mitc 05] B. Mitchell and P. Mckee. "SLAs a key commercial tool". *Innovation and the Knowledge Economy: Issues, Applications, Case Studies*, Vol. 4, pp. 2–2, 2005.

[Moba 06] D. Mobach, B. Overeinder, and F. Brazier. "A WS-Agreement based resource negotiation framework for mobile agents". *Scalable Computing: Practice and Experience*, Vol. 7, No. 1, pp. 23–36, 2006.

[Moha 93] P. Mohapatra, C. Yu, C. Das, and J. Kim. "A lazy scheduling scheme for improving hypercube performance". In: *International Conference on Parallel Processing (ICPP)*, pp. 110–117, IEEE, 1993.

[Mont 06] R. Montero, E. Huedo, and I. Llorente. "Grid Scheduling Infrastructures based on the GridWay Meta-scheduler". Tech. Rep. Newslet-

ter 2, 2006.

[Mual 01] A. Mu'alem and D. Feitelson. "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling". *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 6, pp. 529–543, 2001.

[Mutk 87] M. Mutka and M. Livny. *Profiling workstations' available capacity for remote execution*. University of Wisconsin, Computer Sciences Department, 1987.

[Nah 04] F. Nah. "A study on tolerable waiting time: how long are Web users willing to wait?". *Behaviour & Information Technology*, Vol. 23, No. 3, pp. 153–163, 2004.

[Naro 02] E. Naroska and U. Schwiegelshohn. "On an on-line scheduling problem for parallel jobs". *Information Processing Letters*, Vol. 81, No. 6, pp. 297–304, 2002.

[Ngai 05] E. Ngai and F. Wat. "Fuzzy decision support system for risk analysis in e-commerce development". *Decision support systems*, Vol. 40, No. 2, pp. 235–255, 2005.

[Nieh 09] O. Niehörster, G. Birkenheuer, A. Brinkmann, D. Blunk, B. Elsässer, S. Herres-Pawlis, J. Krüger, J. Niehörster, L. Packschies, and G. Fels. "Providing Scientific Software as a Service in Consideration of Service Level Agreements". In: *Proceedings of the Cracow Grid Workshop (CGW)*, pp. 55 – 63, 2009.

[Nisa 98] N. Nisan, S. London, O. Regev, and N. Camiel. "Globally distributed computation over the internet: The POPCORN project". In: *Proceedings of the International Conference Distributed Computing Systems (ICDCS), Amsterdam, The Netherlands,May 26-29*, 1998.

[Niss 07] A. Nissimov and D. Feitelson. "Probabilistic Backfilling". In: *Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP)*, Jan 2007.

[Nitz 03] B. Nitzberg, J. Schopf, and J. Jones. "PBS Pro: Grid computing and scheduling attributes". *International Series in Operations Research and Management Science*, pp. 183–192, 2003.

[Nten 07] N. Ntene. "An Algorithmic Approach to the 2D Oriented Strip Packing Problem". *disseration*, 2007.

[Nurm 05] D. Nurmi, J. Brevik, and R. Wolski. "Modeling Machine Availability in Enterprise and Wide-Area Distributed Computing Environments". In: *Proceedings of the European Conference on Parallel Processing (Euro-Par)*, pp. 612–612, Springer Berlin / Heidelberg, 2005.

[Nurm 09] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. "The eucalyptus open-source

cloud-computing system". In: *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pp. 124–131, IEEE, 2009.

[Oinn 06] T. Oinn, M. Greenwood, M. Addis, M. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, *et al.* "Taverna: Lessons in creating a workflow environment for the life sciences". *Concurrency and Computation: Practice and Experience*, Vol. 18, No. 10, pp. 1067–1100, 2006.

[Open 11] Openstack Team. "Openstack, open source software to build private and public clouds.". Project Website: `www.openstack.org/`, 2011.

[Pada 03] P. Padala, C. Harrison, N. Pelfort, E. Jansen, M. Frank, and C. Chokkareddy. "OCEAN: The open computation exchange and arbitration network, a market approach to meta computing". In: *Proceedings of the Second International Symposium on Parallel and Distributed Computing*, Published by the IEEE Computer Society, 2003.

[Park 08] M. Parkin, R. Badia, and J. Martrat. "A comparison of SLA use in six of the european commissions FP6 projects". *Institute on Resource Management and Scheduling, CoreGRID-Network of Excellence, Tech. Rep. TR-0129*, 2008.

[Petr 07] A. Petry. *Design and Implementation of a Xen-Based Execution Environment*. Master's thesis, University of Kaiserslautern, Apr. 2007.

[PHOS 10] PHOSPHORUS Team. "PHOSPHORUS – Lambda User Controlled Infrastructure For European Research". Project Website: `www.ist-phosphorus.eu`, 2010.

[Pruh 03] K. Pruhs, J. Sgall, and E. Torng. "Online Scheduling". In: *Handbook of Scheduling: Algorithms, Models and Performance Evaluation*, pp. 115–124, CRC Press, 2003.

[Rain 91] R. Rainer Jr, C. Snyder, and H. Carr. "Risk analysis for information technology". *Journal of Management Information Systems*, Vol. 8, No. 1, pp. 129–147, 1991.

[Reed 99] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. "Xenoservers: Accounted execution of untrusted code". In: *Proceedings of the 7th Workshop Hot Topics in Operating Systems (HotOS-VII), Rio Rico, AZ, Mar. 28-30*, 1999.

[Ress 11] C. Ressources. "TORQUE Resource Manager". Project Website: `http://www.clusterresources.com/pages/products/torque-resource-manager.php`, 2011.

[Rima 09] B. Rimal, E. Choi, and I. Lumb. "A taxonomy and survey of cloud computing systems". In: *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pp. 44–51, Ieee, 2009.

[Roth 85] M. Rothstein. "OR and the Airline Overbooking Problem". *Operations Research*, Vol. 33, No. 2, pp. 237–248, 1985.

[Saha 03] A. Sahai, S. Graupner, V. Machiraju, and A. van Moorsel. "Specifying and monitoring guarantees in commercial grids through SLA". In: *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid CCGrid 2003.*, pp. 292–299, IEEE, 2003.

[Saho 04] R. Sahoo, M. Squillante, A. Sivasubramaniam, and Y. Zhang. "Failure data analysis of a large-scale heterogeneous server environment". In: *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 772–781, 2004.

[Sand 07] T. Sandholm and K. Lai. "Prediction-based enforcement of performance contracts". In: *Proceedings of the 4th International Workshop on Grid Economics and Business Models, GECON 07*, p. 71, 2007.

[Schi 94] I. Schiermeyer. "Reverse-fit: A 2-optimal algorithm for packing rectangles". *Algorithms ESA'94*, pp. 290–299, 1994.

[Schr 06] B. Schroeder and G. Gibson. "A large-scale study of failures in high-performance computing systems". In: *Proceedings of the 2006 international Conference on Dependable Systems and Networks (DSN 06)*, 2006.

[Schu 07] B. Schuller, B. Demuth, H. Mix, K. Rasch, M. Romberg, S. Sild, U. Maran, P. Bała, E. Del Grosso, M. Casalegno, *et al.* "Chemomentum-UNICORE 6 based infrastructure for complex applications in science and technology". In: *Proceedings of the 2007 conference on Parallel processing*, pp. 82–93, Springer-Verlag, 2007.

[Sidd 06] M. Siddiqui, A. Villazón, and T. Fahringer. "Grid allocation and reservation - Grid capacity planning with negotiation-based advance reservation for optimized QoS". In: *Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing*, p. 103, 2006.

[Simc 94] D. Simchi-Levi. "New worst-case results for the bin-packing problem". *Naval Research Logistics*, Vol. 41, No. 4, pp. 579–585, 1994.

[SLA4 11] SLA4D-Grid Team. "SLA4D-Grid". Project Website: `http://www.sla4d-grid.de/`, 2011.

[SLAS 11] SLA@SOI Team. "SLA@SOI". Project Website: `http://www.sla-at-soi.org`, 2011.

[Slea 80] D. Sleator. "A 2.5 times optimal algorithm for packing in two dimensions". *Information Processing Letters*, Vol. 10, No. 1, pp. 37–40, 1980.

[Sloa 03] T. Sloan, R. Abrol, G. Cawood, T. Seed, and F. Ferstl. "Sun data and compute grids". *TOG*, Vol. 500, p. 2, 2003.

[smartlm 10] "SmartLM – Grid-friendly software licensing for location independent application execution". 2010. Project Website: http://www.smartlm.eu.

[Smit 03] C. Smith. "Open source metascheduling for virtual organizations with the community scheduler framework (CSF)". *Technical whitepaper, Platform Computing*, 2003.

[Smit 09] M. Smith, M. Schmidt, N. Fallenbeck, T. Dörnemann, C. Schridde, and B. Freisleben. "Secure on-demand grid computing". *Future Generation Computer Systems (FGCS)*, Vol. 25, No. 3, pp. 315–325, 2009.

[Smit 98] W. Smith, I. Foster, and V. Taylor. "Predicting application run times using historical information". In: *Proceedings of the Job Scheduling Strategies for Parallel Processing (JSSPP)*, Jan 1998.

[Solt 07] S. Soltesz, H. Pötzl, M. Fiuczynski, A. Bavier, and L. Peterson. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors". In: *Proceedings of the 2nd European Conference on Computer Systems ACM SIGOPS/EuroSys*, pp. 275–287, ACM, 2007.

[sorma 10] "SORMA - Self-Organizing ICT Resource Management". 2010. Project Website: http://www.im.uni-karlsruhe.de/sorma.

[Soto 08a] B. Sotomayor, K. Keahey, and I. Foster. "Combining batch execution and leasing using virtual machines". In: *Proceedings of the 17th International ACM Symposium on High Performance Parallel and Distributed Computing (HPDC)*, pp. 87–96, ACM, 2008.

[Soto 08b] B. Sotomayor, R. Montero, I. Llorente, and I. Foster. "Capacity leasing in cloud systems using the opennebula engine". *Cloud Computing and Applications*, Vol. 2008, pp. 1–5, 2008.

[Soto 09a] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. "Virtual Infrastructure Management in Private and Hybrid Clouds". *IEEE Internet Computing*, Vol. 13, No. 5, pp. 14 – 22, 2009.

[Soto 09b] B. Sotomayor, R. Montero, I. Llorente, and I. Foster. "Resource leasing and the art of suspending virtual machines". In: *Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications (HPCC)*, pp. 59–68, IEEE, 2009.

[Stei 97] A. Steinberg. "A strip-packing algorithm with absolute performance bound 2". *SIAM Journal on Computing*, Vol. 26, p. 401, 1997.

[Stew 04] A. Stewart. "On risk: perception and direction". *Computers & Security*, Vol. 23, No. 5, pp. 362–370, 2004.

[Ston 94] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. S. A. Pfeffer, and C. Staelin. "An economic paradigm for query processing and data migration in mariposa". In: *Proceedings of the 3rd International Conference Parallel and Distributed Information Systems, Austin, TX, Sep., 28-30*, 1994.

[Stre 03] A. Streit. *Self-tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids*. PhD thesis, University of Paderborn, 2003.

[Subr 02] V. Subramani, R. Kettimuthu, S. Srinivasan, J. Johnston, and P. Sadayappan. "Selective buddy allocation for scheduling parallel jobs on clusters". In: *Proceedings of the IEEE International Conference on Cluster Computing*, pp. 107–116, IEEE, 2002.

[Subr 99] J. Subramanian, S. Stidham Jr, and C. Lautenbacher. "Airline Yield Management with Overbooking, Cancellations, and No-Shows". *Transportation Science*, Vol. 33, No. 2, pp. 147–167, 1999.

[Suli 08] A. Sulistio, K. H. Kim, and R. Buyya. "Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue". In: *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, pp. 267–276, 2008.

[Take 01] A. Takefusa, H. Casanova, S. Matsuoka, and F. Berman. "A study of deadline scheduling for client-server systems on the computational grid". In: *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 406–415, IEEE, 2001.

[Tayl 07] I. Taylor. *Workflows for e-science: scientific workflows for grids, chapter Petri Nets*. Springer-Verlag New York Inc, 2007.

[Tsaf 05] D. Tsafrir, Y. Etsion, and D. Feitelson. "Modeling user runtime estimates". In: *Proceedings of the Job Scheduling Strategies for Parallel Processing (JSSPP)*, 2005.

[Tsaf 06] D. Tsafrir and D. Feitelson. "The dynamics of backfilling: solving the mystery of why increased inaccuracy may help". In: *Proceedings of the IEEE International Symposium on Workload Characterization*, 2006.

[Tsaf 07] D. Tsafrir, Y. Etsion, and D. Feitelson. "Backfilling Using System-Generated Predictions Rather than User Runtime Estimates". *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, pp. 789–803, 2007.

[Ture 94] J. Turek, U. Schwiegelshohn, J. Wolf, and P. Yu. "Scheduling parallel tasks to minimize average response time". In: *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 112–121, Society for Industrial and Applied Mathematics, 1994.

[Urga 02] B. Urgaonkar, P. Shenoy, and T. Roscoe. "Resource overbooking and application profiling in shared hosting platforms". *ACM SIGOPS Operating Systems Review*, Vol. 36, No. si, p. 239, 2002.

[Van 05] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. Mark, and H. Berendsen. "GROMACS: fast, flexible, and free". *Journal of computational chemistry*, Vol. 26, No. 16, pp. 1701–1718, 2005.

[Vaqu 08] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. "A break in the clouds: towards a cloud definition". *ACM SIGCOMM Computer Communication Review*, Vol. 39, No. 1, pp. 50–55, 2008.

[Vega 81] W. Fernandez de la Vega and G. Lueker. "Bin packing can be solved within 1+ ε in linear time". *Combinatorica*, Vol. 1, No. 4, pp. 349–355, 1981.

[Venu 04] S. Venugopal, R. Buyya, and L. Winton. "A grid service broker for scheduling distributed data-oriented applications on global grids". In: *Proceedings of the 2nd workshop on Middleware for grid computing*, pp. 75–80, ACM, 2004.

[Verb 10] S. Verboven, K. Vanmechelen, and J. Broeckhove. "Multiplexing Low and High QoS workloads in Virtual Environments". In: *Proceedings of the 15th International Workshop of Job Scheduling Strategies for Parallel Processing JSSPP Apr. 23, Atlanta, GA, USA*, 2010.

[VMwa 09] VMware. "Whitepaper: VMware vSphere 4 - The best platform for building cloud infrastructures". Project Website: `http://www.vmware.com/products/vsphere/mid-size-and-enterprise-business/features.html`, 2009.

[Voss 06] K. Voss. "Risk-aware Migrations For Prepossessing SLAs". In: *Proceedings of the International conference on Networking and Services*, p. 68, IEEE Computer Society, 2006.

[Voss 08] K. Voss, I. Gourlay, J. Padgett, D. Battré, I. Rosenberg, and G. Birkenheuer. "Risk-enhanced Broker Service". Tech. Rep., 2008.

[Wael 11] O. Waeldrich. "WSAG4J". Project Website: `http://packcs-e0.scai.fraunhofer.de/wsag4j/`, 2011.

[Wald 02] C. Waldspurger. "Memory Resource Management in VMware ESX Server". In: *Proceedings of the 5th Conference on Operating Systems Design and Implementation (OSDI)*, pp. 181 – 194, Boston, MA, Dec. 2002.

[Wald 92] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta. "Spawn: A distributed computational economy". In: *IEEE Trans. Softw. Eng., vol. 18, no. 2, pp. 103-117*, Feb. 1992.

[Walt 08] J. Walters, B. Bantwal, and V. Chaudhary. "Enabling interactive jobs in virtualized data centers". In: *Proceedings of the 1st Work-*

shop on Cloud Computing and Its Applications (CCA), Chicago, Illinois, USA, 2008.

[Wang 10] P. Wang, W. Huang, and C. Varela. "Impact of virtual machine granularity on cloud computing workloads performance". In: *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pp. 393–400, IEEE, 2010.

[Wats 08] J. Watson. "Virtualbox: bits and bytes masquerading as machines". *Linux Journal*, Vol. 2008, No. 166, p. 1, 2008.

[Whit 95] D. White. "Application of systems thinking to risk management:: a review of the literature". *Management Decision*, Vol. 33, No. 10, pp. 35–45, 1995.

[Wied 08] P. Wieder, J. Seidel, O. Wäldrich, W. Ziegler, and R. Yahyapour. "Using SLA for Resource Management and Scheduling-A Survey". *Grid Middleware and Services*, pp. 335–347, 2008.

[Wils 07] M. Wilson, A. Arenas, D. Chadwick, T. Dimitrakos, J. Doser, P. Giambiagi, D. Golby, C. Geuer-Pollman, J. Haller, S. Ketil, *et al.* "The TrustCoM approach to enforcing agreements between interoperating enterprises". *Enterprise Interoperability*, pp. 365–375, 2007.

[Wols 01] R. Wolski, J. Plank, J. Brevik, and T. Bryan. "Analyzing market-based resource allocation strategies for the computational grid". In: *Proceedings of the International J. High-Performance Computing Applications, vol. 15, no. 3*, 2001.

[Xia 10] B. Xia and Z. Tan. "Tighter bounds of the First Fit algorithm for the bin-packing problem". *Discrete Applied Mathematics*, 2010.

[Xu 99] J. Xu, Z. Kalbarczyk, and R. Iyer. "Networked Windows NT system field failure data analysis". In: *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, pp. 178–185, IEEE, 1999.

[Ye 07] X. Ye and G. Zhang. "Strip Packing vs. Bin Packing". In: *Proceedings of the 3rd international conference on algorithmic aspects in information and management (AAIM)*, 2007.

[Ye 09] D. Ye, X. Han, and G. Zhang. "A note on online strip packing". *Journal of combinatorial optimization*, Vol. 17, No. 4, pp. 417–423, 2009.

[Yoo 03] A. Yoo, M. Jette, and M. Grondona. "SLURM: Simple linux utility for resource management". In: *Job Scheduling Strategies for Parallel Processing (JSSPP)*, pp. 44–60, Springer, 2003.

[Yue 91] M. Yue. "A simple proof of the inequality FFD (L) geq 11/9 OPT (L)+ 1,all L for the FFD bin-packing algorithm". *Acta Mathematicae Applicatae Sinica*, Vol. 7, No. 4, pp. 321–331, 1991.

[Zhou 92] S. Zhou. "LSF: Load sharing in large-scale heterogeneous distributed systems". In: *Proceedings of the Workshop on Cluster*

Computing, pp. 1995–1996, 1992.

[Zieg 08] W. Ziegler, P. Wieder, and D. Battré. “Extending WS-Agreement for dynamic negotiation of service level agreements”. Tech. Rep., vTR-0172, 2008.

[Zotk 99] D. Zotkin and P. Keleher. “Job-length estimation and performance in backfilling schedulers”. In: *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC)*, Jan 1999.

Glossary

$P_{\textbf{available}}$ The probability that the resources of a job are available at start time. 43, 49

$P_{\textbf{executable}}$ The probability that the job is successful within its given time slot. 44, 50

$P_{\textbf{success}}$ The probability that the resources are valuable during the whole runtime. 45, 50

PoF(j) The probability of failure calculated for a new job j. 43

PoS(j) The probability of success calculated for a new job j. 43, 49

acceptance test Decision whether or not to accept a new job. 77

additional gain The additional income of a DCI provider by applying an overbooking strategy compared to the not overbooking one. 100

application analysis Statistical analysis of the user estimation quality based on classes of jobs build on the requested application of worker. 65

Arminius Compute cluster at the PC^2 in Paderborn that produces input traces for the simulation. 58

AssessGrid Advanced Risk Assessment and Management for Trustable Grids. 6

Auctions A market mechanism. 18, 53

backfilling Scheduling strategy that is able to fill gaps in case a job fits in. 26

Bartering Models A market mechanism for resource exchange. 19, 54

BF Best Fit packing algorithm. 24, 82

BFDH Best Fit Decreasing Height. 25

Bidding A reverse auction. 19, 54

BL Bottom Left packing algorithm. 23

BPEL WS-Business Process Execution Language. 29

C3Grid Grid Scheduler of the Climate Community. 28

CAVE Cave Automatic Virtual Environment. 8

CCS Computing Center Software. 28

CDF Cummulative Density Function. 12, 38

charge The charge or fees are earned for fulfilling an SLA. 75

CIM Common Information Model. 29

CloudStack Management of server and compute nodes with virtualization technologies. 30

combined PDF resulting PDF of the PoF calculation for the comprehensive overbooking. 41

Commodity Market A market where the prices are given by a formula. 19, 54

comprehensive overbooking Overbooking strategy that works on the same basis as the conservative backfilling. It calculated the PoF for overbooking based on specific resource stability information and the plan of jobs. It is part of the simulation environment. 42, 55

Condor Workload management system for compute-intensive jobs. 28

conservative backfilling Backfilling strategy that does not delay other jobs, part of the simulation environment. 27, 55

CPU Central Processing Unit. 3

CSF Community Scheduler Framework. 29

CTC Cornell Theory Center. 86

DCI Distributed Compute Infrastructure. 6, 8–11

deadline The SLA defined time until a job has to be finished. 34

DGSI D-Grid Scheduler Interoperability. 8

EASY Extensible Argonne Scheduling sYstem. 27

EASY++ EASY with system-generated execution time predictions. 27

EC European Commission. 6

Eucalyptus Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems. 30

failure rate The failure rate λ. 34, 49

FBS Finite Best-Strip. 25

FCFS First Come First Serve. 26

fee The charge or fees are earned for fulfilling an SLA. 75, 100

FF First Fit packing algorithm. 23, 24, 81

FFDH First Fit Decreasing Height. 25

FFS First Fit Shelf. 25

FMEA Failure Mode and Effects Analysis. 21

Freefluo The workflow enactment engine of Taverna. 29

FST Fuzzy Set Theory. 21

FT Fault Tolerance. 22

FTA Fault Tree Analysis. 21

gain The gain is the income of a DCI provider. 88

Grid Service Broker A broker developed to handle data-oriented applications. 29

GridEngine successor of the Sun Grid Engine. 28

GridWay meta scheduler of Globus Toolkit. 28

GT Guarantee Term. 16

GWES Grid Workflow Execution Service. 28

Heizea OpenNebula scheduler. 30

heuristic overbooking Overbooking strategy that works on the basis of the heuristic planning. It is part of the simulation environment. 48, 55

heuristic planning Counts resources and checks whether or not a job can be executed with its requested resources and full estimated runtime. Resources are not assigned to jobs at planning time, a job can start anytime after the release on any free resource. It is part of the simulation environment. 55

HFF Hybrid First Fit. 25

HPC High Performance Computing. 11, 31

HPC2n High-Performance Computing Center North, Sweden. 103

Hyper-Exponential Failure Model Statistical model to describe a continuous probability distribution. 46, 50

IaaS Infrastructure as a Service. 11

ITIL IT Infrastructure Library. 16

JAWS Extension to batch schedulers to allow the use of virtual machines. 30

jBPM The workflow engine from JBoss. 29

joint PDF PDF build on all the jobs before a new job. 40

KVM Kernel Virtual Machine. 30

LANL Los Alamos National Lab. 109

libvirt A virtualization API. 11

LoadLeveler IBM scheduler for serial and parallel jobs. 28

LRMS Local Resource Management System. 9, 28

LSF Load Sharing Facility. 28

Maestro-VC Xen based cloud solution. 29

Maui Scheduler that can be used by many different RMSs. 28

MSS Meta Scheduling Service. 28

MTTF Mean Time To Failure. 44

MTTR Mean Time To Repair. 44

NF Next Fit packing algorithm. 23

NFDH Next Fit Decreasing Height. 25

NFS Next Fit Shelf. 25

nimrod/G A meta scheduler based on Globus Toolkit. 29

OGF Open Grid Forum. 2

OpenNebula Project to study scheduling of virtual machines. 30

openPBS open Portable Batch System. 28

OpenStack Nasa project that provides components for compute and object storage with the support of CloudStack. 30

opportunity The probability of success multiplied with its worth. 36

P-GRADE P-GRADE or WS-PGRADE is a grid portal technology based on the grid User Support Environment. 29

PaaS Platform as a Service. 11

PBS Portable Batch System. 28

PDF Probability Density Function. 12, 38, 44

penalty The penalty has to be paid by the provider for violating an SLA. 75

PoF Probability of Failure. 4, 21

Poisson Process Statistical model to describe a continuous probability distribution. 46, 51

PoS Probability of Success. 5, 21

profit The profit is the income of a DCI provider. 35, 75

PWA Parallel Workload Archive. 73, 85

QoS Quality of Service. 16

RAM Random-Access Memory. 3

repair rate The repair rate μ. 34, 49

resource analysis Statistical analysis of the user estimation quality based on classes of jobs build on the requested amount of resources. 63

risk The probability of occurrence of an dangerous event multiplied with its the impact. 20, 21, 36

RMS Resource Management System. 24, 27

SaaS Software as a Service. 11

SDSC San-Diego Supercomputer Center. 114

SDSC-BLUE San-Diego Supercomputer Center Blue Horizon. 115

SDSC-DataStar San-Diego Supercomputer Center DataStar. 121

SDSC-SP2 San-Diego Supercomputer Center SP2. 127

SDT Service Description Term. 16

SGE Sun Grid Engine. 28

SLA Service Level Agreement. 2

SLO Service Level Objective. 16

SLURM a highly scalable resource manager designed for Linux clusters of all sizes. 28

SME Short and Medium Enterprise. 2

Taverna A workflow language for the semantic grid. 29

Tendering A reverse auction. 19, 54

time analysis Statistical analysis of the user estimation quality based on classes of jobs build on the length of the estimated runtime. 60

Torque RMS based on openPBS. 28

UBIS Extension to batch schedulers to allow the use of virtual machines. 30

UNICORE Uniform Interface to Computing Resources. 28

user analysis Statistical analysis of the user estimation quality based on classes of jobs build for specific users. 67

VC Virtual Coin, the corresponding virtual currency unit for a booked CPU hour. 76

VM Virtual Machine. 11, 29

VMware VMware VM Hypervisor. 30

VO Virtual Organization. 10

Weibull Failure Model Statistical model to describe a continuous probability distribution. 45, 50

WS-Agreement Web Service Agreement. 2, 16

WS-Negotiation Web Service Agreement Negotiation. 16

WS-PGRADE P-GRADE or WS-PGRADE is a grid portal technology based on the grid user support environment (gUSE). 29

WSAG4J Web Service Agreement for Java. 16

WSLA Web Service Level Agreement. 16

WSS Workflow Scheduling Service. 28

XBEE Xen Based Execution Environment. 29

Xen Xen VM Hypervisor. 30

XGE Xen Grid Engine. 29