KTH Information and
Communication Technology

Statistical Methods for Computational Markets

Proportional Share Market Prediction and Admission Control

THOMAS SANDHOLM

Doctoral Thesis
Stockholm, Sweden 2008

DSV Report series No. 08-006
ISSN 1101-8526
ISRN SU-KTH/DSV/R--08/6--SE
ISBN 978-91-7178-924-2

KTH School of Information and Communication Technology
Forum 100, SE-164 40 Kista
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av Filosofie doktorsexamen i Data- och Systemvetenskap måndagen den 26 maj 2008 klockan 13.00 i Sal C, Forum, Kungl Tekniska högskolan, Kista.

Tryck: Universitetsservice US AB

Abstract

We design, implement and evaluate statistical methods for managing uncertainty when consuming and provisioning resources in a federated computational market. To enable efficient allocation of resources in this environment, providers need to know consumers' risk preferences, and the expected future demand. The guarantee levels to offer thus depend on techniques to forecast future usage and to accurately capture and model uncertainties. Our main contribution in this thesis is threefold; first, we evaluate a set of techniques to forecast demand in computational markets; second, we design a scalable method which captures a succinct summary of usage statistics and allows consumers to express risk preferences; and finally we propose a method for providers to set resource prices and determine guarantee levels to offer. The methods employed are based on fundamental concepts in probability theory, and are thus easy to implement, as well as to analyze and evaluate. The key component of our solution is a predictor that dynamically constructs approximations of the price probability density and quantile functions for arbitrary resources in a computational market. Because highly fluctuating and skewed demand is common in these markets, it is difficult to accurately and automatically construct representations of arbitrary demand distributions. We discovered that a technique based on the Chebyshev inequality and empirical prediction bounds, which estimates worst case bounds on deviations from the mean given a variance, provided the most reliable forecasts for a set of representative high performance and shared cluster workload traces. We further show how these forecasts can help the consumers determine how much to spend given a risk preference and how providers can offer admission control services with different guarantee levels given a recent history of resource prices.

Keywords: Distributed Systems, Grid Computing, Performance Analysis, Workload Modeling, Middleware, Quality of Service, Prediction, Admission Control, Grid Economics, High Performance Computing, Time Series Analysis

Sammanfattning

Vi utformar, implementerar och utvärderar statistiska metoder för att hantera osäkerhet vid användning och tillhandahållande av datorresurser på en marknad. För att kunna möjliggöra en effektiv allokering av resurser i en sådan miljö måste konsumenters riskpreferenser och framtida efterfrågan tas med i beräkningen. De garantinivåer som erbjuds beror således på tekniker för att prognostisera framtida användning och på precis modellering av osäkerhet i prognoserna.

Denna avhandling har tre huvudsakliga bidrag: en utvärdering av tekniker för att prognostisera efterfrågan på datorresursmarknader, en skalbar metod för att summera användningsstatistik som tillåter konsumenter att uttrycka riskpreferenser, samt en metod för att sätta resurspriser och bestämma vilka garantinivåer som kan erbjudas. Metoderna som använts bygger på grundläggande sannolikhetsteori och är därför enkla att implementera, analysera och utvärdera.

Den viktigaste komponenten i vår lösning är en prognosgenerator som dynamiskt konstruerar approximeringar av prisfördelningsfunktioner för godtyckliga resurser på en datorresursmarknad. Eftersom kraftigt fluktuerande och sneda pristäthetsfunktioner är vanliga är det svårt att approximera efterfrågans godtyckliga fördelningsfunktioner automatiskt och precist. Vi fann att en teknik som bygger på Chebyshevs olikhet och empiriska prognosintervaller, vilken uppskattar avvikelser från ett medelvärde givet en varians, producerade de mest tillförlitliga prognoserna i tester med olika användningsloggar från superdatorcentra. Vi visar också hur dessa prognoser kan hjälpa konsumenter att bestämma hur mycket som skall spenderas givet riskpreferenser och hur tillträdesregler för resurser kan bestämmas med olika servicegarantier givet en resurs prishistorik.

To Gisell and Errol

Acknowledgments

Thanks to Professor Magnus Boman, my advisor, for taking me on as a graduate student and being instrumental in helping me through a difficult transition between research departments as well as research topics midway through my graduate program. I am also thankful for all the general thesis and scientific advice and prompt and detailed feedback on my manuscripts, which helped shape this thesis.

I am very grateful to Lars Rasmusson, my secondary advisor, and also a former colleague and collaborator at Hewlett-Packard Labs, who with his insightful advice allowed me to refocus my research and who constantly challenged me to improve my scientific reasoning and presentation.

Most of the research presented in this thesis was done in a collaboration with the Information Dynamics Lab at Hewlett-Packard Labs, in Palo Alto. I therefore received most of the day-to-day technical advice and feedback from my mentor there, Kevin Lai. He taught me the economics and resource allocation science underlying computational markets, and gave me a new perspective on how software systems should be developed to be economically efficient. As a co-author of many of the papers in this thesis he also contributed a great deal in terms of focussing and positioning my work to address interesting research problems, guiding experimental setups, relating the work to the grander research challenges and refining the presentation of the motivation and the results. Needless to say, his contributions were invaluable to me. My work at HP Labs would not have been possible without the constant support from Bernardo Huberman, who has been a great inspiration both as a scientist and as a research manager. I had the pleasure of collaborating with many great scientists at HP Labs who helped improve the work in this thesis through numerous discussions and direct feedback on regular presentations I gave. I particularly would like to mention Scott Clearwater, who introduced me to the statistics of high performance computing, and Li Zhang who mentored me on most of the mathematical theory covered in this thesis. I would also like to thank Fang Wu, Mike Brzozowski, Ali Ghodsi, John Wilkes, Tad Hogg and all the members of the IDL group.

Since this thesis is the end product of a long education and previous work in other research labs, I would also like to take the opportunity to thank people who inspired me in my research career and helped me choose this path. Professor Stefan Tai, my master's thesis advisor, was the person who first got me interested

in research as a career and who introduced me to the scientific method, which I have benefited greatly from both academically and professionally ever since. While working as a consultant at IONA Technologies, I learned a great deal about large-scale distributed computing and professional programming through my mentors Eamon Walshe, and Tom Watson. During my 2-year visit as a research programmer at Argonne National Laboratories, I was mentored by Steven Tuecke and Ian Foster who helped shape my research focus on Grid computing. When I first started out as a graduate student at the Royal Institute of Technology in Stockholm I was able to keep my focus on Grid research and expand my knowledge about Grid security thanks to the expertise of fellow researcher Olle Mulmo. I would also like to thank Professor Lennart Johnsson who gave me the opportunity to embark on my PhD studies, and Peter Gardfjell and Erik Elmroth, who I collaborated with on my first research project as a graduate student, Grid accounting.

On a more personal note, I would like to thank my wife Gisell and my son Errol, whom I dedicate this thesis to, for always being there for me and helping me through the hard work of pushing the envelope and coming up with new research results, and for their understanding of daddy spending many weekends at work instead of being around.

Finally, I would like to thank my family in Sweden, my parents Hete and Lennart, and my brother Peo, for making it possible through their constant and tireless help with tasks that had to be done locally, which allowed me to be a graduate student in Stockholm while living in Palo Alto.

Chronological List of Contributions

[1] Thomas Sandholm, Kevin Lai, and Scott Clearwater. Admission Control in a Computational Market. In *CCGrid '08: Proceedings of the 8th International Symposium on Cluster Computing and the Grid*, 2008. To appear.

[2] Thomas Sandholm. Autoregressive Time Series Forecasting of Computational Demand. Technical Report 0711.2062v1 [cs.DC], arXiv, 2007.

[3] Thomas Sandholm and Kevin Lai. Prediction-Based Enforcement of Performance Contracts. In *GECON '07: Proceedings of the 4th International Workshop on Grid Economics and Business Models*, 2007.

[4] Thomas Sandholm. *Managing Service Levels in Grid Computing Systems*. Licentitate Thesis ISRN KTH/CSC/A–07/06–SE. Royal Institute of Technology, Stockholm, 2007.

[5] Thomas Sandholm and Kevin Lai. A Statistical Approach to Risk Mitigation in Computational Markets. In *HPDC '07: Proceedings of the 16th ACM International Symposium on High Performance Distributed Computing*, 2007.

[6] Peter Gardfjell, Erik Elmroth, Lennart Johnsson, Olle Mulmo, and Thomas Sandholm. Scalable Grid-wide Capacity Allocation with the SweGrid Accounting System (SGAS). *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[7] Thomas Sandholm and Kevin Lai. Evaluating Demand Prediction Techniques for Computational Markets. In *GECON '06: Proceedings of the 3rd International Workshop on Grid Economics and Business Models*, May 2006.

[8] Thomas Sandholm, Kevin Lai, Jorge Andrade, and Jacob Odeberg. Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications. In *HPDC '06: Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, June 2006.

[9] Thomas Sandholm, Peter Gardfjell, Erik Elmroth, Lennart Johnsson, and Olle Mulmo. A Service-Oriented Approach to Enforce Grid Resource Al-

locations. *International Journal of Cooperative Information Systems*, 15(3): 439–459, 2006.

[10] Kevin Lai and Thomas Sandholm. The Design, Implementation, and Evaluation of a Market-Based Resource Allocation System. Technical Report Manuscript for Publication, Royal Institute of Technology and Hewlett-Packard Labs, Stockholm, Sweden, May 2006.

[11] Thomas Sandholm. Service Level Agreement Requirements of an Accounting-Driven Computational Grid. Technical Report TRITA-NA-0533, Royal Institute of Technology, Stockholm, Sweden, September 2005.

[12] Thomas Sandholm. The Philosophy of the Grid: Ontology Theory - From Aristotle to Self-Managed IT Resources. Technical Report TRITA-NA-0532, Royal Institute of Technology, Stockholm, Sweden, September 2005.

[13] Ludwig Seitz, Erik Rissanen, Thomas Sandholm, Babak Sadighi Firozabadi, and Olle Mulmo. Policy Administration Control and Delegation using XACML and Delegent. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, November 2005.

[14] Frank Siebenlist, Takuya Mori, Rachana Ananthakrishnan, Liang Fang, Tim Freeman, Kate Keahey, Sam Meder, Olle Mulmo, and Thomas Sandholm. The Globus Authorization Processing Framework. In *Workshop on New Challenges for Access Control*, Ottawa, Canada, April 2005.

[15] Erik Elmroth, Peter Gardfjell, Olle Mulmo, and Thomas Sandholm. An OGSA-Based Bank Service for Grid Accounting Systems. In Jerzy Wasniewski, editor, *Lecture Notes in Computer Science: Applied Parallel Computing. State-of-the-art in Scientific Computing*. Springer Verlag, 2004.

[16] Thomas Sandholm, Peter Gardfjell, Erik Elmroth, Lennart Johnsson, and Olle Mulmo. An OGSA-based Accounting System for Allocation Enforcement across HPC Centers. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 279–288, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-871-7.

[17] Steven Tuecke, Karl Czajkowski, Ian Foster, Jeff Frey, Steven Graham, Carl Kesselman, Tom Maquire, Thomas Sandholm, David Snelling, and Peter Vanderbilt. Open Grid Services Infrastructure (OGSI) Version 1.0. Technical report, Global Grid Forum, 2003.

Contents

Chapter 1

Introduction

Computational resources are becoming easier to share as a result of technological advances in network, operating system, and middleware infrastructure. This new infrastructure has enabled a trend towards openness, accessibility, and community contribution where resources, services and applications are hosted and consumed on the Internet. The scientific computing community has led the advancement of highly-scalable wide-area network distributed compute farms (a.k.a. compute clouds), capable of solving complex, academic, grand-challenge problems by means of large pools of commodity resources. Operating system virtualization technologies have made it possible to both consume and provide these commodity resources with a high level of control over the Quality-of-Service (QoS) being delivered. This increased control allows more accurate scheduling decisions to be made to shape the supply to fit the demand, in addition to the more traditional approach of fitting the demand to the supply. However, when allowing individual consumers to control their own allocations, some mechanism is required to ensure fairness and efficiency in the system as a whole.

Traditional compute-farm job schedulers have focussed on optimizing utilization, throughput and response time and have only given users minimal control over service levels for individual jobs. Priorities among users and jobs are typically configured *a priori* by system administrators, and cannot easily adapt to changes in demand and supply.

Computational markets have been proposed as an economic solution to this problem of large-scale resource sharing. The general idea is to allow users to express their preferences both in terms of resource requirements and value or priority of tasks to be computed. Priorities and service levels are automatically set based on how much money is spent. Users can thus budget their jobs by minimizing cost while maximizing performance. Conversely, providers can multiplex users' jobs by applying some algorithm that maximizes their profit. The advantage of this approach is that the self-optimizing behavior of users and providers lead to globally fairer and more efficient allocations without centralized control. Providers

can control the demand by setting prices dynamically, and users can signal their satisfaction with a service level delivered by their willingness to pay a certain price for it.

Just like in traditional markets such as the stock market, being able to predict future demand is important to make sensible and economically sound resource allocation decisions both for the seller (resource provider) and the buyer (resource consumer). For example, if a user has a low priority job and the price is predicted to drop, the execution of the job could be deferred. On the other hand, a provider who sees a trend of increasing prices may want to be careful with admitting low-priority tasks that cannot be preempted. The advantage of a computational market is that prices can effectively summarize a large set of system states, which simplifies the monitoring and accounting as well as the forecasting processes. Price and time series forecasting are well studied fields in economics and physics, and a large number of techniques have been developed to summarize trends and make claims of future outcomes.

1.1 Problem Statement

The main question addressed in this thesis is how statistical demand forecasting methods can be integrated into a large-scale compute farm infrastructure to allow both resource consumers and resource providers to make economically and computationally efficient allocation decisions.

1.2 Main Contribution

The main contribution of this thesis is a set of methods to predict demand in computational markets based on the proportional share resource allocation principle. We call the model encompassing these methods the *Proportional Share Market Prediction and Admission Control* (PS-MP/AC) model. PS-MP/AC comprises methods to efficiently collect and summarize resource prices; algorithms to approximate bounds of future demand with statistical claims; a risk probing interface for resource consumers; and finally an access control mechanism for resource providers. In addition to developing and analyzing these methods we also contribute:

- a Grid market broker which schedules Grid applications in an economic compute farm,
- an evaluation of a large set of computational demand prediction techniques,
- an analysis and characterization of demand measured in existing compute farms,
- a time-series regression (ARIMA) modeling evaluation of computational demand,

- and finally experimental and simulation based evaluations of access control in computational markets.

1.3 Other Contributions

This thesis also summarizes some contributions leading up to the development of the PS-MP/AC model including:

- contributions to a standard Grid service protocol,
- development of a Grid accounting system,
- development of a Grid security policy framework,
- experimental evaluation of economic efficiency and fairness bounds in a proportional share market,
- and an analysis of the accounting requirements of High Performance Computing (HPC) applications.

The contributions in this category were described in the Licentiate thesis: *Managing Service Levels in Grid Computing Systems: Quota Policy and Computational Market Approaches* [101], and will therefore only be briefly summarized in this thesis.

1.4 Methodology

The results in this thesis were obtained mainly by five means:

- **Trace Analysis.** High Performance Computing (HPC) and super computing center job traces as well as load traces from a large-scale shared computational network were pre-processed to represent time series of global demand. The time series were then analyzed to find patterns and statistical properties. Traces were also used to evaluate predictor models, and to drive simulations and experiments. Exploratory data-analysis with a focus on distribution and correlation visualizations was extensively employed. Distribution fitting techniques, such as moment fitting, maximum entropy and maximum-likelihood optimization, and expectation maximization (for mixed distributions) were used to create quantitative models of the data.
- **Mathematical Modeling.** Probabilistic models were designed based on the trace analyses and the distributional properties discovered. Alternative models were designed and compared through simulations and experiments. Standard models used in related studies were also compared to the new models proposed. This typically included one or many benchmark models that lacked one or more properties that were considered important based on the data

analysis. A preference was given to models that were a) easy to implement and b) easy to analyze. Occam's razor was one of the leading principles of this step.

- **Prototype Implementation.** Models are simplifications, and some behavior can only be uncovered in actual implementations, for example performance overhead. All models were therefore implemented in both prototype simulations and more robust implementations in full-scale systems. Many of the implementations were also tested with real users and under real workloads. Feedback was received from users and this information was used to refine the models and implementations. The feedback in some cases also helped refine the questions that we focussed on in our research. Implementation languages used included: Python, R, Matlab, Java and Bash-shell.

- **Simulation Evaluation.** Simulations were designed to test initial implementations and to narrow down the problem and parameter space that was most interesting to test in a live system. Simulations could also more easily be compared to previous work and standard as well as purely theoretical models. For instance optimal off-line clairvoyance scenarios could be compared to the on-line algorithms proposed. Statistical significance tests are used where appropriate, including Monte Carlo bootstrap driven tests, Kolmogorov-Smirnov tests, Student t-tests, various distributional boundary tests and standard time-series regression tests, such as Dickey-Fuller and Box-Ljung tests.

- **Experiment Evaluation.** Simulations in closed, controlled virtual systems are easier to evaluate but the end-goal is to build a distributed system that works in practice. To this end the focus of the work in this thesis has been on running experiments under as realistic conditions as possible, allocating real resources in live systems with real users and applications. Ideally we would have preferred to have a heavily used (competitive) computational pilot market to verify the models on, but since no such pilot was available to us, we had to go with the second best option of running simulated user loads with real applications in the real system, and measuring sporadic usage from real users. We finally note that both the main predictor contribution as well as the accounting system contribution (under other contributions) implementations have been tested in production mode with external users, and are also available to the community as open source. All experiments were conducted in the HP Labs Tycoon cluster of 80 nodes in Palo Alto. The core part of the Tycoon system had already been developed before the work with this thesis began, so most of the development work involved extending Tycoon in the areas of Grid integration, predictability, and admission control and to write simulation and experiment test suites.

1.5 Related Work

Related work in this area has focussed on the problem of predicting wait times in batch schedulers and predicting economic equilibrium prices based on centralized optimization algorithms [77, 124, 127, 13, 128]. We found these techniques to be insufficient either in terms of economic viability or computational efficiency. The fact that these methods do not leverage the inherent predictability of proportional share resource allocations also made them suboptimal for our purpose. Many robust statistical techniques exist if stationarity, independent and identically-distributed (iid) samples, and symmetry of distributions are assumed. However, our compute farm workload trace analyses confirmed what others also have found when analyzing parallel compute center traces: that these assumptions do not hold in most real systems [80, 40, 60, 26]. Our solution is fully decentralized and thus scales better than the centralized optimization-based techniques, such as [24, 62, 93, 5]. Furthermore, the centralized optimization solutions typically depend on home-grown heuristics, because the general problem of scheduling parallel non-preemptive tasks has been shown to be NP-hard [53]. What sets our solution apart from other work is also the careful attention to scalability issues and the minimal collection of statistical summary data points while imposing as few restrictions as possible on the types of predictions the data point consumers can make. A more elaborate description and treatment of related work can be found in Chapter 6.

1.6 Disposition

The thesis is organized as follows. In Part I we summarize the foundations and results of our work. Chapter 2 presents the problem domain and the underlying technology and theory. Our solution is described in Chapter 3. The software that was developed as part of the thesis research is described in some more detail in Chapter 4, and then the contributions are summarized in Chapter 5. In Chapter 6 we discuss related work, and finally Chapter 7 provides concluding remarks.

In Part II we include six papers that represent the main contribution of this thesis. These papers have all been previously published in conference proceedings or as technical reports.

Part I
Overview

Chapter 2

Foundation

In this chapter, we discuss the foundational concepts and theory of the work presented in this thesis. First, we describe the new paradigm of computing emerging in *Computational Grid* systems. Then, we review the fundamental microeconomic theory of games and strategic behavior; discuss the concepts and different flavors of computational markets; present the underlying theory of the prediction models used in this thesis; and outline some common approaches to distributed resource admission control. Finally, we describe the purpose of various statistical tests employed.

2.1 Grid Computing

In the context of this thesis the Grid refers to a collection of computational resources shared across organizational boundaries to deliver non-trivial Qualities of Service (QoS) [49, 48, 7]. Non-trivial here means that services beyond pure information sharing, as typical in the World Wide Web, are offered. What is in common for these more advanced services offered by a Grid is that they typically involve large-scale resource consumption within a dynamic community of users and providers spread across a large geographic area. This community is known as a *Virtual Organization* (VO) [47]. An example VO architecture is shown in Figure 2.1. One of the first super computing projects to span multiple organizations while utilizing a cross-Atlantic Grid was the I-WAY project [31], which paved the way for Grid computing as a scientific field.

Security

Many of the trust, privacy and general security issues appearing in the Grid revolves around management of rights within a VO. The idea is that a VO is a web of trust where information exchange and resource sharing can take place just like in a corporate Intranet. The difference is that Virtual Organizations may be created,

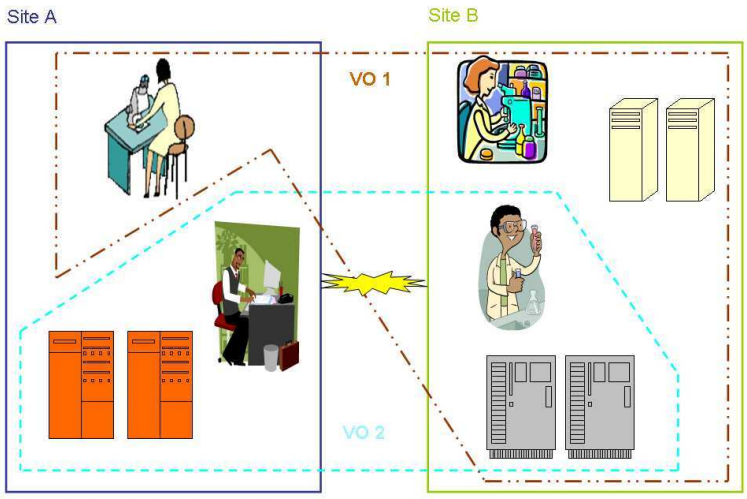


Figure 2.1: Virtual Organization Example.

managed and destroyed in a more dynamic manner. Examples include ATLAS [1], a particle physics experiment utilizing the computational Grid of the Large Hydron Collider at CERN; and HapGrid, a bioinformatics project performing haplotype reconstruction and frequency estimation using the SweGrid computational Grid resources [2].

The trust verification mechanism in Grid systems is based on the Public Key Infrastructure (PKI) [59], with extensions to allow delegation of rights and single sign-on using self-signed *proxy certificates* [114, 123]. A user will have a secret key on her local machine and then distribute a public key to all communication partners. A message can then be signed or encrypted with the private key by the sender to allow the recipient to verify the authenticity of the message including non-repudiation, and protection against denial-of-service (DoS) and replay attacks. The PKI handshake protocol where authenticity is verified has two main advantages compared to more traditional username and password based authentication protocols. First, no personal secret such as a password or private key needs to be sent across the communication link exposing it to eavesdropping. Second, mutual authentication of senders and receivers is seamless, making it a good fit for peer-to-peer like systems, such as the Grid. Another fundamental concept is the *Certificate Authority* (CA), which is a trust anchor asserting the identity of its users by signing their credentials (public keys). CA's may be established for individual VO's, a

[1] http://atlas.web.cern.ch/Atlas/

collection of VO's using a particular Grid environment, a country for its citizens, etc. Certificate Authorities may also be organized hierarchically, where the parent nodes assert the identity of their child nodes.

The use of proxy certificates allows brokers or agents to act on behalf of users to complete a task. The broker will not simply receive the private key of the user, as it would violate the rule of *strong authentication*, which states that no long-lived personal secrets should be distributed as part of the identity verification process. Instead the user creates a temporary key-pair, signs it, encrypts it, and sends it to the broker. Proxy certificates thus enables single sign-on across a network of brokers.

Resource Allocation

Service level and QoS enforcement was addressed in a Grid context in the Grid Advanced Reservation and Allocation (GARA) [44, 45] project allowing CPU, bandwidth and OS process resource capacity enforcement at different levels of service. Here resources were configured using resource specific control mechanisms, such as DiffServ and RSVP router management [9, 12], and DSRT CPU scheduling control [83]. This work evolved into the SNAP protocol [28] and then eventually was standardized in the WS-Agreement specification [3], by GGF, which also borrows many concepts from IBM's WSLA (SLA for Web services) solution [29] and SLAng [72].

Complimentary to protocol standardization, heterogeneity can also be addressed by resource virtualization. For example, virtualization of a host operating system [34] gives fine-grained control over the service levels offered. CPU, disk, memory, and other resource shares can be allocated to user specific virtual machines. This technique has been explored in the context of Grid job execution management in [64].

As the Grid deployments extend beyond academic projects, such as EDG [2] [10], EGEE [3] , TeraGrid [4], NEESit [5], ESG [6], and OSG [7] to self-sufficient commercial Grid environments, the need to charge for compute resource usage like any other commodity arises. This business model is in-line with many IT companies' utility (or cloud) computing strategy [54, 16, 58]. Economic models from the field of utility computing could also solve the growing problem in academic Grid projects of a small number of strategic users hogging the system. We will elaborate on how this could be approached in the next section.

[2] European Data Grid, http://www.edg.org
[3] Enabling Grids for EScienc E, http://egee-intranet.web.cern.ch/egee-intranet/gateway.html
[4] http://www.teragrid.org
[5] http://it.nees.org
[6] Earth System Grid, http://www.earthsystemgrid.org
[7] Open Science Grid, http://www.opensciencegrid.org

2.2 Microeconomic Theory

When managing service levels, we would like to make sure that the system cannot be abused by strategic users, who could starve out competing resource consumers. We therefore turn to microeconomic theory to study how mechanisms can be developed to ensure an overall healthy system even with strategic users.

Tragedy of the Commons

Consider the problem often referred to as the *Tragedy of the Commons* [57]. Farmers let their sheep eat grass on a common. A farmer can sell one of his sheep when it has been well fed and earn a profit compared to the original purchase price of the sheep. Let's further assume that the profit that an individual farmer gains from selling a sheep is higher than the relative cost of having one more sheep share the grass of the common, and thus leaving less grass available for other sheep. A strategic farmer who is trying to optimize his own profits would under such circumstances always choose to purchase another sheep. The main issue with this situation is that the overall health of the community of farmers declines as individuals optimize their profits, and eventually it will collapse when there are too many sheep on the common for any single one of them to get fed well enough to be sold. It is not hard to see that such situations could easily arise if computational power is offered as a common good without providing some incentive for users to constrain their usage.

Game Theory

In *Game Theory* [88, 87] a number of *players* and their possible *actions* with associated individual *preferences* model a *game*. Other players' actions affect the *utility* or *payoff* a player receives from a game. However, the other players' actions may not be known before a player chooses an action. In order to choose an action each player hence needs to make a guess of other players' likely actions given past experience, which is referred to as forming a *belief*.

Let

$$a^* = \{a_1 ... a_k\} \tag{2.1}$$

be the set of actions taken by the k players in a game, where a_i is the action taken by player i. This set is called the *action profile* of the game.

We can now make statements about the *steady states* of a game, when no player has an incentive to change her action.

Nash Equilibrium

A *Nash equilibrium* is defined as an action profile a^* where no player i can get a higher utility by changing her action a_i^*, given that every other player j performs the action a_j^*. More concisely expressed

$$u_i(a^*) \geq u_i(a_i, a_{-i}^*) \tag{2.2}$$

for every action a_i of player i, where u_i is the utility function that represents player i's preferences and (a_i, a^*_{-i}) is the action profile where player i performs action a_i and all other players j perform action a^*_j.

It is important to note that a Nash equilibrium does not make any statements about uniqueness of the solution, and many games can indeed have multiple Nash equilibria.

To simplify the decision making process for a player given prior beliefs a *best response function* is typically defined. It yields the set of best actions to take for a player given an action profile of the other players, or more precisely

$$B_i(a_{-i}) = \{a_i \in A_i : u_i(a_i, a_{-i}) \geq u_i(a'_i, a_{-i})|_{\forall a'_i \in A_i}\} \tag{2.3}$$

where A_i is the set of all possible actions player i can take, a_{-i} the action profile including all players except player i, and B_i is the set of best response actions.

Resource Allocation Game

In our case a game can be defined as the process of allocating available Grid resources, or shares of a resource, to the applications that users are requesting to run on those resources. The users can form their prior beliefs of other users' demand of the resources by studying the current resource prices on the market. In order to analyze the efficiency and fairness of a resource allocation algorithm we need some additional definitions.

The *efficiency* or *price of anarchy* [89] is calculated as the sum of all users' utilities of a certain allocation outcome compared to the optimal utility in the system. The sum of all users' utilities is typically referred to as the *social welfare*, and it is an indication of the global health of the system.

The social welfare for an allocation scheme ω is defined as

$$U(\omega) = \sum_{i=1}^{k} u_i(r_i) \tag{2.4}$$

where k is the number of users, r_i is the resource share allocated to user i, and u_i is the utility function of user i.

The fairness of a resource allocation scheme can be defined in terms of *envy-freeness* [116] which can be calculated as

$$\rho(\omega) = \min(\min_{i,j} \frac{u_i(r_i)}{u_i(r_j)}, 1) \tag{2.5}$$

where $u_i(r_i)$ is the utility that user i received from being allocated share r_i, whereas $u_i(r_j)$ is the utility user i would have received had she been allocated the resource share r_j of user j instead. In an envy-free system (optimally fair) $\rho(\omega)$ equals 1. The closer the value is to 0 the more envy there is, and the more unfair the allocation scheme is.

The task of an economically healthy resource allocation scheme is to enforce both high efficiency and high fairness in the Nash equilibrium states of the game.

When constructing a mechanism to allocate resources in a computational market, it is therefore important to force users towards taking actions that yield one of these equilibrium state. In a system where a *Tragedy of Commons* behavior is possible no equilibrium states will ever be reached. In other words, it should not be possible to game (trick) the allocator for individual benefit at the cost of the overall health of the system in terms of fairness and efficiency. A mechanism that yields an equilibrium state in the presence of strategic users is said to be *strategy proof*. Likewise a software system architecture implementing a computational economy is *truth-telling* if users have an incentive to restrict their signaled and actual usage of a resource to their true needs. Further, it is *incentive-compatible* if users who have an incentive to perform a task either perform it themselves or transfer the incentive to a broker to perform the task on behalf of them. Incentive-compatibility is key to any system to avoid the Tragedy of Commons problem occurring, and it necessitates the deployment of transposable (tradable) and commensurable (comparable) entities, e.g. a currency.

Best Response Agent

A game theoretical analysis tries to model the behavior of players and make statements about optimal strategies and mechanisms enforcing certain global behavior based on local rules. Strategies can be implemented on behalf of a player by an *agent*. One example of an agent that implements an optimal strategy to solve the resource allocation game just described is the *best response agent* presented in [41, 132]. Given a fixed budget and a pool of *divisible* resources allocated according to a proportional share mechanism that allows users to bid on shares (described in more detail in Section 3.1 and 4.1), the best response agent finds the distribution of bids across resources that yields the highest utility for an individual player. The prior beliefs of the demand used by the agent to make its decision is the sum of all bids in the previous bidding cycle for all the available resources. Zhang [132] shows that there always exists a Nash equilibrium when the players' utility functions are strongly competitive, i.e. when there are at least two users competing for each resource. Furthermore, a tight, lower efficiency bound of $\Theta(1/\sqrt{m})$ and a lower envy-freeness bound of $2\sqrt{2}-2$ or approximately 0.828 in Nash equilibria with m players are theoretically deduced.

2.3 Computational Markets

The concept of computational markets was defined in [81] as a software system using market mechanisms. Other terms used are *agoric system* and *computational ecology* to emphasize the meeting aspect or the decentralized, evolutionary, and self-organizing properties of markets respectively.

Why use markets?

Markets can effectively aggregate information from multiple sources in large, dynamic and complex systems where there is not a single source with complete information. In a computational market, resource allocation can be made locally, where detailed performance information is available, to achieve better scalability and reliability. Prices summarize demand succinctly and uniformly and can be communicated globally to enable economic load balancing between resources as well as between different resource types. The vision is that selfishly utility-optimizing resource consumers and selfishly profit-optimizing resource providers will move the system into an equilibrium state where resources are allocated efficiently.

There are a couple of fundamental goals of computational markets to meet this vision. First, to provide the right incentive to users to specify truthful cross-user priorities of their resource needs. Second, to dynamically set resource prices to find the level where demand equals supply, a.k.a. the market equilibrium price.

Incentives

Most computational markets use some kind of currency to meet the first goal, which in essence combats the tragedy of commons effect, and the main differences here are whether real or fake money is used and how users receive funding. A stream of income [120, 36], automatic refunding after resources have been consumed [113], and closed loop funding [69, 92] are all examples of funding policies that have been implemented. Closed loop funding refers to the policy where resource providers receive currency spent during local usage and this income can later on be used to consume resources elsewhere. Whether fake or real money is used as a currency can change the user behavior drastically. Risk attitudes of users tend to change, for example. Lessons learned from experimental economics [19] tell us that the most truthful behavior is obtained when real money is at stake. Fake money, on the other hand has the advantage of allowing full control over inflation, which is why it is preferred in most experimental settings. A compromise is to provide exchange rates between fake and real money.

Price-Setting

The most distinguishing feature of a computational market is how prices are set. The simplest approach is to assume that the providers manually set prices based on observed demand [63, 6, 99]. One critique of this approach is that price levels adapt very slowly and some price fluctuations may be missed completely. The result is lost profit for the provider caused by truncated demand during peak-demand periods and utilization loss during low-demand periods.

The most popular approach in computational markets is to use some form of auction to extract the market price directly from users' bids. Examples of auctions that have been used include open-bid English [113] and Dutch auctions [43]

(auctioneer increases or decreases prices until a single bidder is found); sealed-bid second price auctions a.k.a. Vickrey auctions [120]; and continuous double auctions [71, 20], typical for the stock market where both buy and sell requests are posted concurrently and dynamically matched. More advanced combinatorial auctions, where preferences across multiple resource can be expressed in bundles, have also been used [30, 18]. They typically rely on centralized evaluation, and are known to be NP-complete and thus require heuristics to solve. There are four main critiques against auctions. First, they rely on high market liquidity to be successful, in particular continuous double auctions are known to fail for this reason [91]. Second, bidders have to wait for auction clearing (resolution) times before they get their allocations. Third, there can be extensive computational overhead when bidding for resources under contention, leading to a slow adjustment to market prices. Fourth, auctions may cause unpredictable and heavily fluctuating price levels.

To achieve price stability more rapidly, market equilibrium prices may also be found directly by the providers using mathematical and economic optimization models such as Walras's tatonnement process, Smale's equilibrium method or gradient descent methods [128, 66, 125]. The main drawback of these methods is that they are complex and hard to implement efficiently in large-scale systems. Simpler methods of finding stable prices include proportional share auctions [119, 69, 23], where allocations are proportional to the bid a consumer places and inversely proportional to the sum of all other bids for the same resource; and prices based on exponential smoothing of observed demand in the past [36].

2.4 Forecasting

The most efficient resource allocations can be inferred *a posteriori* by analyzing the complete set of usage interactions in the system. Hence, models that can predict future usage accurately can make more efficient allocation decisions. For example, a long job of little importance with a large footprint of resource usage may be rejected or delayed if short and important jobs are expected to arrive. Demand forecasts can, furthermore, be exploited both by users to determine the optimal time to submit their jobs, and by providers to price capacity reservations. In that sense, a forecasting tool can have a self-optimizing effect on the system. More specifically, forecasts provide the foundation when making *a priori* estimates of the risk of missing deadlines, or losing profit. To make good predictions the dynamics of the demand needs to be effectively captured and reproduced. A good predictor is recognized both by its accuracy, most estimates are close to the true value, and its reliability, even the worst estimates are close to the true value. Below a number of techniques, known from the fields of economics and econometrics, to forecast demand and to estimate risk are discussed.

Central Limit Theorem

The *central limit theorem* (CLT) due to de Moivre and Laplace [82] postulates that any sum of independent and identically-distributed (iid) random variables with finite variance tend to a normal distribution. The CLT implies that the mean obtained from a large enough sample is the best predictor for future point estimates. This property is also known as the law of large numbers due to Bernoulli and Poisson [55]. It also implies that the risk of outliers can be easily obtained from the sample variance, and the known Gaussian distribution function. A typical usage of the CLT is to create $(1-\alpha)100$ per cent confidence bounds of a prediction as

$$\mu \pm \sigma\Phi^{-1}(1-\frac{\alpha}{2}) \tag{2.6}$$

where μ is the sample mean, σ the sample standard deviation (square root of sample variance), and Φ^{-1} is the inverse of the standard Gaussian (zero mean and variance 1) cumulative distribution function (CDF). For example based on the CLT assumptions and measured values of μ and σ of a large enough sample (typically at least 30 data points) we can claim that 95 per cent of future values will lie within the interval $[\mu - 1.96\sigma, \mu + 1.96\sigma]$. Many known processes such as the Poisson process and Brownian motion or Wiener processes adhere to the assumptions of the CLT, and can thus be accurately estimated with this technique. The CLT assumptions are however violated for processes that can be described by so called scale-free or power law distributions with infinite higher moments, such as the Pareto distribution. Furthermore, it does not hold if the samples are correlated, which is typical in time-dependent series. Demand for computational resources has been observed to be both highly time-correlated and exhibit extreme peaks as a result of highly skewed, heavy tailed power law distributions [40, 60].

Time Series Analysis

The observation that samples often exhibit correlations in time, is the cornerstone of the theory of *time series analysis* [122, 108]. Time series analysis is heavily based on the theory of regression analysis where maximum likelihood algorithms are used to fit data trends. The time series models, however, introduce additional restrictions on the model parameters to ensure *stationarity*. Stationarity here means that the mean and the variance are stable over time. If a series is not stationary it is transformed or detrended using techniques such as differencing (studying the difference or the increments of subsequent values), and power (Box-Cox) transforms. Time series analysis methods fall into two broad categories, the time-domain approach and the frequency-domain approach. We will focus on the former as it uses more easily tractable and simple mathematical models. The advantage of the time series approach is that it provides very generic mathematical formulations encompassing a wide range of widely used statistical techniques such as moving average, exponential smoothing, and random walk or Wiener process modeling. The general

idea is to measure the correlations in time, and then to reconstruct models that reproduce these correlations as accurately as possible. The models are built up from two parts; a regressive part representing the correlation between the current and previous values; and a moving average part representing the correlation between a random error in the current value and previous random errors. These models are in their general form referred to as $ARIMA(p, d, q)$ or autoregressive integrated moving average models with regression order p, differencing order d and moving average order q. The orders determine how far back in time correlations should be represented. The mathematical representation for a time dependent series Z at time t is:

$$\phi_p(B)(1-B)^d Z_t = \theta_0 + \theta_q(B)a_t. \tag{2.7}$$

where $\phi_p(B) = (1 - \phi_1 B - \cdots - \phi_p B^p)$, $\theta_q(B) = (1 - \theta_1 B - \cdots - \theta_q B^q)$ and B is the backshift operator where $(1 - B)Z_t = Z_t - Z_{t-1}$ The constants ϕ and θ are determined by studying the autocorrelations (correlations between time lags in the same series) of the time series typically using autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. The plots allow the estimation of p, d, and q, and variations of maximum likelihood or other regression and correlation fitting techniques can then be used to estimate the most likely values of the constants. Once the model dimensions and the parameters are found the model can be used to make forecasts. One weakness of these models is that predictions typically quickly converge to a mean value or diverge to infinity. But short term (e.g. 1,2,3 steps ahead) forecasts can be very accurate, and it is straightforward to construct confidence intervals analytically. Other weaknesses include: the stationary assumption may be violated beyond transformational repair, and the model dimensions and parameters themselves may not be stationary. Various extensions of the ARIMA model try to address these shortcomings, such as the popular GARCH model by Engle [39] which models heteroskedasticity or volatility changes over time. In general, though, time series models have been more successful in describing and understanding time-correlated processes than predicting them. For example a random walk process is described as an ARIMA(0,1,0) process, an exponential smoothing process can be described by an ARIMA(0,1,1) process, and standard q-order regressive models reduce to ARIMA(q,0,0).

Density Estimation

To make predictions that focus on confidence bounds and general distributional properties of a time series as opposed to point estimates, a density estimation [106] predictor may be used. The approach generalizes the predictor in Equation 2.6 and allows for arbitrary underlying distributions, instead of assuming normal or Gaussian distributions. The general idea is that summary statistics are collected from samples and then an approximation of the density function, CDF, or its inverse, the percent point function, PPF, is constructed. To obtain a confidence interval the appropriate percent points serve as input to the PPF. The simplest approach is

to build a histogram with bins recording the portion of the samples falling within fixed intervals within the full range of values. The number of bins used will affect the results, and thus needs to be chosen with great care. One popular approach is the Freedman-Diaconis binning [50] algorithm which looks at the distance between quartiles to determine the number of bins to use. This approach still produces unreliable results if the range of values is very large, which is the case in power law or heavy tailed distributions. Moreover the optimal bin size to use may change over time, e.g. when there are regime shifts in the series. To resolve these shortcomings smoother functional representations are used. These functions may be constructed by explicitly smoothing empirical distributions (shaving off outliers and shrinking the distance between local samples of values), by fitting the moments to generic density functions, or by optimizing the entropy given a set of restrictions such as moment estimates. Explicit smoothing has the problem of knowing when to stop smoothing the distribution, the more you smooth the further away from the sample distribution you get and the closer you get to a straight line through the mean. Capturing the empirical distribution is also expensive computationally, and not practical for on-line *ad hoc* predictions. Generic distributional models include, generalized extreme value theory models (generalization of Weibull which in turn generalizes normal and heavy tailed distributions) and deviation theory bounds distributions (e.g. Chebyshev and the 3σ-rule [42]). Finally, the entropy model is based on information theory results obtained by Shannon [27]. The approach is known as MaxEnt [129], since it maximizes the entropy defined as

$$H(X) = -\sum_i p(x_i) log(p(x_i)) \tag{2.8}$$

where p is $Pr(X = x_i)$ or the probability of a given outcome, x_i, and the summation is done over all possible outcomes of the random variable X. The entropy intuitively represents the level of randomness (or information contained in a model). So the idea is to maximize the randomness allowed by the model (encompass as many extreme deviations as possible) given a set of restrictions in the form of moments (typically the first three or four). If only the first two moments are used the maximum entropy algorithm produces the normal distribution model. The algorithm can be implemented using a standard optimization algorithm such as gradient descent. It has the nice property that not only the mean and variation can be mimicked in the model but also the skewness and the kurtosis behavior. However, a high skewness or high kurtosis will cause problems for the model to converge, so they are sometimes not fit directly but estimated through transformations. In any case, the risk of not converging imposes both computational overhead and unreliable result during highly unstable periods (e.g. regime shifts).

2.5 Admission Control

Purely statistical models assume that there is a correlation between the current value and the past history of values observed, and that some structural pattern

or trend can be extracted. They also assume that there is an opportunity for statistical multiplexing, i.e. some users can preempt others while still meeting the statistical guarantees for everyone. In cases where these assumptions are not true, a model where the provider can reject admission is needed. A system without the capability of making explicit rejections is known as a *best-effort system*, whereas a system which makes explicit decisions about rejections is known as a *reservation system*. Admission controllers can be designed to optimize the utility for users or the profit for providers. The general goal is to reject as few users as possible to avoid underutilization, while violating (preempting) as few existing commitments or contract reservations as possible. A good admission controller ensures that high priority tasks receive a high level of service even when the system is overloaded. A provider deciding on whether to use a best-effort or a reservation model must make tradeoffs between optimizing scalability, reliability, and utilization (best-effort) or offering higher guarantees to users (reservations). Higher guarantees thus come at a cost for the provider and would therefore naturally result in a premium price for users. The key issue to solve is hence to set the price for reservations to compensate for the incurred costs while still meeting the price and guarantee preferences of the users in the system. Since the price depends on supply and demand, the expected load (demand) as well as the capacity to offer (supply) on the reservation versus best-effort (spot) markets must be taken into account.

Traffic Engineering

The Internet was designed as a best-effort service in terms of bandwidth at its core in order to be reliable and scalable. But some applications need more stringent flow performance guarantees to operate, and thus various QoS models were proposed on top of the core infrastructure. DiffServ [8] uses a decentralized approach where packets are marked with *per-hop-forwarding behavior* (PHB) priorities, at network boundary nodes, which are enforced at the core nodes. Services built on top of the DiffServ architecture can give guaranteed boundaries of throughput, delay, jitter and loss at differentiated price levels, to accommodate a heterogeneous set of applications with different risk and guarantee preferences. The IntServ [11] architecture on the other hand allows end-to-end network paths to be reserved by applications with real-time requirements, such as on-demand media streaming. The fundamental change to the original best-effort model of the Internet is that flow specific state is maintained in the routers at the core of the network. This state allows the routers to enforce absolute end-to-end reservations of bandwidth for specific flows using admission control. DiffServ is in general seen as the more successful attempt of the two at ensuring QoS in IP networks as it is more scalable and robust. The usefulness of the models depends heavily on whether there is a need to explicitly resolve resource contention conflicts between different applications and whether these applications are willing to pay the additional cost incurred by the providers. However, due to the massive improvement of bandwidth performance and availability in recent years these models have only had limited success and very

marginal impact on the Internet infrastructure as a whole.

Queuing Network Theory

Most computational systems can be represented as a set of connected resources with different service times, queue sizes and arrival rates. Markov chain models (states with memory-less transition probabilities) can then be applied to provide throughput and response time guarantees to users as well as provide capacity planning recommendations to the providers based on measured service times and arrival rates. The fundamental result often used is Little's theorem [74]:

$$N = \lambda T \tag{2.9}$$

which states that the average number of users, N, in a stable system equals the inter-arrival rate, λ, of users times the average time, T, users spend in the system. Queuing network models have been particularly popular in Web server admission control and capacity planning applications, as both service times and arrival time models are simple and predictable. However, as Web servers provide more advanced services in addition to simple file retrieval, the queueing models become harder to apply.

2.6 Statistical Tests

Statistical tests are used to confirm or reject hypotheses (conjectures) formally and quantitatively in order to make claims about experimental results with different levels of confidence. Typically 5 per cent confidence levels are used, which means that there is less than 5 per cent probability that the test results could have been obtained by virtue of pure randomness and not because the hypothesis is true. More formally, if the significance level (confidence) of a test is a, it means that the probability of rejecting a hypothesis given that it is true, $Pr(Reject|True) = a$. To determine whether to reject or not to reject (supporting but not accepting) a hypothesis a test statistic T, is used. If the test statistic of an experiment falls within a range $(lower, upper)$, the significance level a, mapped to that range by some appropriate distribution is said to be the significance level of the result.

Monte Carlo Bootstrapping

One issue with these tests is that they depend on the size of the sample generated by the experiment. Furthermore, most standard statistical tests have been designed with the assumption that a very small sample is taken from a very large population. In a computational setting with very large samples of job traces, one might in some cases rather like to test a theoretical model on many small subsamples of the entire trace, and then make statements about how well the model fits these subsamples. The technique of generating a large number of samples and evaluating

their individual fits to models is known as Monte Carlo bootstrapping [35, 32] and it was conceived as a way to leverage the power of automated computational verification to test large parameter spaces. A typical use-case is to run risk scenario evaluations of different portfolios of securities in finance.

Kolmogorov-Smirnov

One of the most popular tests is the Kolmogorov-Smirnov (K-S) test, which takes two distributions, typically one generated by a standard model and one generated by an experiment, and then determines whether they are the same. In essence the K-S test measures the maximum difference between the CDFs of the distributions and is thus both intuitive and easy to calculate. Formally, the K-S statistic is defined as:

$$D = \sup_x |F_n(x) - F(x)| \tag{2.10}$$

where $F_n(x)$ is the empirical distribution function obtained from the experiment, and $F(x)$ is the distribution function of the standard model.

Distributional Confidence Bounds

The Central Limit Theorem is commonly applied to simplify the test procedure. If one can assume that the experimental results have a normal distribution, e.g. because of a large number of independently drawn samples, then the z-test may be applied. The z-test uses the normal (Gaussian) distribution to calculate the lower $a/2$ and upper $1 - a/2$ confidence bounds of an experimental measurement with significance level a, i.e. the probability is $1 - a$ that the measured variable is within the confidence bounds given the mean and variance of its distribution. However, when the sample size is small (typically less than 30) the z-test may not be applied, because the sample variance can then no longer be assumed to be the population variance. In this case the t-student test may be applied instead. It determines whether the mean of two samples are the same given their sample means and variances. Instead of constructing the confidence bounds using the Gaussian distribution they are taken from the t-student distribution. If the population distribution model is known, the confidence bounds can be calculated in the same way as for the z and the t-student tests using the percent point (quantile) function of the known distribution. Alternatively if the model is unknown but the sample is large enough the empirical CDF/PPF may be used as basis for the bounds.

Time-Series Regression Tests

When constructing models of time-series, a large number of statistical tests are applied at different stages of the analysis. Here we just summarize some of the most important ones used in this thesis. The Box-Ljung test [75] is used to determine whether the autocorrelations (lagged correlations) of the residuals of a model are

significant, i.e. if the model represents the data well. The Dickey-Fuller test [33] is used for unit-root testing to determine whether the series needs to be differentiated and to determine if the process is well represented by a memory-less Brownian motion or Wiener process where the correlation between the current and last value is 1, that is the movement of the process is fully determined by white noise (Gaussian randomness). The t-test, and ANOVA (Analysis of Variance) tests [82] such as the R^2-test are also very common in time series regression analysis to determine how many lags to include in the ARIMA model, and how well the models explain the variance of the data respectively. The R^2 statistic for n data points is generally computed as:

$$R^2 = 1 - \frac{\sum_i^n (x_i - r_i)^2}{\sum_i^n (x_i - \mu)^2} \tag{2.11}$$

where x_i is data point i, r_i is the regression model point i, and μ is the mean of all data points. As the model fit improves, the R^2-statistic approaches 1.

Chapter 3

Model

In this chapter we present our *Proportional Share Market Prediction and Admission Control* (PS-MP/AC) model. The goal of PS-MP/AC is twofold, first it should allow users to bid for resources according to their risk preferences, and second it should allow providers to price admission control services. There are four layers in the model. The bottom layer is responsible for allocating proportional resource shares, the second layer collects and summarizes price history, the third layer generates density estimation based predictions, and the top layer comprises two components, one that serves as a risk probing interface to users and one that helps providers control admissions (see Figure 3.1).

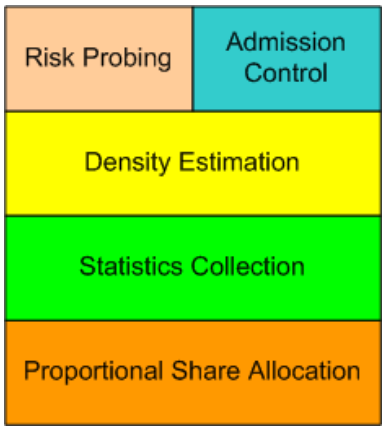


Figure 3.1: Proportional Share Market Prediction and Admission Control Model.

3.1 Proportional Share Allocation

In the proportional share allocation model, each consumer obtains a share of the resource which equals the ratio of the weight of the consumer to the sum of all weights of consumers of the resource,

$$q_j = \frac{w_j}{\sum_{i=1}^{k} w_i} \tag{3.1}$$

where q_j is the resource share obtained by consumer j, w_j the weight of consumer j, and k the number of consumers. The weights can all be equal in which case the allocation reduces to $1/k$ for all consumers. In our market-based resource allocation model the weights are set by bids made by the individual consumers, and the sum of all bids on a resource represents the price, c.

$$c = \sum_{i=1}^{k} b_i \tag{3.2}$$

where b_i is the bid of consumer i.

3.2 Statistics Collection

The next layer in our model is responsible for continuously collecting and summarizing historical prices for resources. Statistics are provided in the form of a series of sample moments over different time intervals. Two alternative models have been implemented to collect the moments, exponential smoothing used in [104, 103], and time horizon slots used in [105].

Exponential Smoothing

The moments for a time window of size n samples are calculated as follows:

$$\begin{aligned} \mu_{i,p} &= \alpha \mu_{i-1,p} + (1-\alpha) c_i^p \\ \mu_{0,p} &= c_0^p \end{aligned} \tag{3.3}$$

where $\mu_{i,p}$ is the p^{th}-order moment in snapshot i, $\alpha = 1 - 1/n$, and c_i is the sample price value in snapshot i. The advantage of this approach is that it is very flexible and easy to implement but it does not handle regime shifts symmetrically. A regime shift from high to low values will linger longer in memory than a regime shift from low to high values. This is an issue particularly for longer time intervals.

Time Horizon Slots

To address the asymmetric memory issue, a second model was designed. In this model running sums of moments are aggregated in time slots at different scales or

time horizons, with one series of time slots per scale. The series at scale dimension d, for moment m, at time t contains

$$S(d) = \left\{ \sum_{i=0}^{d-1} c_{t-i}^{m}, \sum_{i=0}^{d-1} c_{t-d-i}^{m}, \cdots, \sum_{i=0}^{d-1} c_{t-s\times d-i}^{m} \right\} \tag{3.4}$$

where s is the number of slots in the series, which can vary between different dimensions. For example, an hour dimension would have 60 slots, a day dimension 24 slots, a week dimension 7 slots, and a month dimension 30 slots. The m^{th}-order sample moment over the last n-scale(d) time interval can then be calculated as:

$$\mu(m) = \frac{1}{n} \sum_{j=1}^{n} S_j(d)/d \tag{3.5}$$

where $S_j(d)$ is the jth slot in the series of dimension d. As an example, to get the average (first moment) over the last three-day time interval at time t and assuming new values appear every minute, we have $m = 1$,$n = 3$, $d = 60 \times 24 = 1440$, and get

$$\frac{\left(\sum_{i=0}^{1439} c_{t-i}\right) + \left(\sum_{i=0}^{1439} c_{t-1440-i}\right) + \left(\sum_{i=0}^{1439} c_{t-1440\times 2-i}\right)}{3 \times 1440} \tag{3.6}$$

We note that a separate series of slots is needed for each moment and each dimension that is supported, and the moments obtained are moments about the origin (as in the exponential smoothing model). The advantage of this model is that the accuracy of historical moments is much higher. However, it requires some more data to be gathered, and it is thus not feasible if a large number of moments is to be collected.

To calculate the moments about the mean (central moments) which are needed to obtain, e.g. the standard deviation, skewness and kurtosis values the binomial transformation [90] can be applied:

$$mom_m = \sum_{j=0}^{m} \binom{m}{j} (-1)^{m-j} mom'_j \times \mu^{m-j} \tag{3.7}$$

where mom_m is the m^{th}-order central moment, mom'_j is the j^{th}-order moment about the origin, and μ is the sample mean.

3.3 Density Estimation

To make predictions based on the statistics collected, we can first use a time series smoothing technique to predict the T-step ahead moments. Since the most common use case is to predict the 1-step ahead moment and it can be approximated well with the 1-step back moment directly, this smoothing phase may be skipped. We in other words assume an ARIMA(0,1,0) model or a random walk process. For $T > 1$

more advanced ARIMA models may be applied. Using the collected and possibly smoothed moments we create a functional approximation of the CDF and PPF of the price. In our model we apply the Chebyshev inequality [42] and 3-σ rule [118] to extract the CDF from the first two moments as follows. The Chebyshev inequality postulates that

$$Pr(|C - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{3.8}$$

where C is the random variable representing the price c, μ is the sample mean and σ the sample standard deviation. The proof of this bound can be sketched as follows:
Define $I_{(|X| \geq a)}$ to be 1 if $|X| \geq a$ and 0 if $|X| < a$. Thus we have

$$\begin{gathered} aI_{(|X|\geq a)} \leq |X| \\ E[aI_{(|X|\geq a)}] \leq E[|X|] \\ Pr(|X| \geq a) \leq E[|X|]/a \\ \text{which is the Markov bound. Now Setting } X = (C-\mu)^2 \\ \text{and } a = (k\sigma)^2 \text{ we get} \\ Pr(|C - \mu| \geq k\sigma) \leq 1/k^2 \end{gathered} \tag{3.9}$$

This bound can be tightened in two ways. First by considering the one-tailed Chebyshev ineqality called Cantelli's inequality [42]

$$Pr(C - \mu \geq k\sigma) \leq \frac{1}{1 + k^2}. \tag{3.10}$$

and second by applying the 3-σ rule (Vysochanskii-Petunin inequality) which tightens the bound for unimodal distributions for $k > \sqrt{\frac{8}{3}}$

$$Pr(|C - \mu| \geq k\sigma) \leq \frac{4}{9k^2} \tag{3.11}$$

The resulting CDFs are then obtained as follows, using Cantelli's tightened Chebyshev bound as our example:

$$\begin{aligned} Pr(C - \mu \geq k\sigma) &\leq \frac{1}{1 + k^2} \\ Pr(C \geq k\sigma + \mu) &\leq \frac{1}{1 + k^2} \\ 1 - Pr(C \geq k\sigma + \mu) &\geq 1 - \frac{1}{1 + k^2} \\ Pr(C < k\sigma + \mu) &\geq 1 - 1/(1 + k^2) \end{aligned} \tag{3.12}$$

This last inequality can be used as a conservative bound for the price CDF, i.e. $Pr(C \leq c)$, where $c = k\sigma + \mu$. In other words the likelihood is at least $1 - 1/(1 + k^2)$ that the price is less than c but most likely greater. Hence we define

$$CDF(c) = \begin{cases} 1 - \frac{1}{1+k^2} & , k \leq \sqrt{\frac{8}{3}}, \\ 1 - \frac{4}{9k^2} & , k > \sqrt{\frac{8}{3}}. \end{cases} \tag{3.13}$$

where k is $\frac{c-\mu}{\sigma}$, and the PPF (inverse CDF)

$$PPF(p) = \begin{cases} \mu \pm \sigma\sqrt{\frac{1}{1-p} - 1} & , k \leq \sqrt{\frac{8}{3}}, \\ \mu \pm \sigma\sqrt{\frac{4}{9(1-p)}} & , k > \sqrt{\frac{8}{3}}. \end{cases} \tag{3.14}$$

3.4 Risk Probing

With approximations of the CDF and PPF of the price in our market, we can now return to the proportional share definitions in Equation (3.1) and (3.2) to make predictions regarding performance (expected resource shares), bids, and guarantees. The model can answer three questions posed by resource consumers to probe the risk involved in bidding for resources.

1. **What is the performance q that can be obtained with a g100 per cent likelihood if b monetary units (e.g. dollars) are bid?** Our model gives:

$$q = \frac{b}{b + PPF(g)} \tag{3.15}$$

2. **What should be bid, b, to obtain a g100 per cent likelihood that the performance will not drop below q?** Our model gives:

$$b = \frac{PPF(g)q}{1 - q} \tag{3.16}$$

3. **What is the g100 per cent likelihood that the performance will not drop below q when bidding b?** Our model gives:

$$g = CDF\left(\frac{(1-q)b}{q}\right) \tag{3.17}$$

To provide information about the uncertainty of the predictions and the model itself, we also calculate empirical prediction bounds [56, 126]. The PPF and CDF will then be applied to historical values of the price, and upper percentiles of the results will be used as opposed to the current point estimates. This gives a sense about how stable the price predictions are over time and how far ahead in time it makes sense to predict before the prediction bounds become too wide. The three questions above can thus be extended with a fourth parameter describing which upper prediction bound percentile to use. This percentile can be interpreted as the

likelihood that the prediction made is true. We also note that the PPF function can be used instead of the current price in a game theoretical best response agent algorithm to determine bids and performance risks of bids across a set of hosts when competing for resources with other consumers. This approach is described in more detail in [104].

3.5 Admission Control

The prediction model (CDF and PPF approximations) can also be used by the provider to determine which resource contract requests to admit and which to reject. In our model the admission decision is made based on two admission tests that both have to evaluate to true to grant admission. The first test is whether the bid associated with the request is greater than a certain percentile of the price distribution of the resource requested, i.e.

$$\frac{(1-q)b}{q} \geq PPF(g) \tag{3.18}$$

where q is the resource share (performance) requested by the user, b is the user's bid, and g is a guarantee level set by the provider to account for non-preemption loss. The basic idea is to force consumers to pay more for the additional service of not allowing preemption of admitted requests. The additional price paid can be seen as a compensation for the provider's loss of rejecting future higher paying requests. The second test is whether the contract request would violate any previously admitted contracts, determined as

$$\forall s \in S : \sum_{h}^{n} \frac{b_h(s)}{b_h(s) + b_h(r) + c} \geq q_s \tag{3.19}$$

where S is the set of all existing contracts including the requested contract, n is the number of resource hosts, $b_h(s)$ is the bid on resource host h in contract s, $b_h(r)$ is the bid on resource host h in the requested contract r, and q_s is the minimum performance share promised in contract s.

Chapter 4

Software

In this chapter we present the two key software contributions related to the main theme of this thesis, statistical methods for computational markets. The first contribution is an economic broker developed for academic High Performance Computing Grid applications, and the second contribution is the proportional share market prediction and access control implementation. Both of these contributions are based on a marked-based resource allocation system, called Tycoon. Although the core part of Tycoon was developed by others we describe it here to set the context for the software contributions described in Section 4.2 and Section 4.3. For more details on which parts of Tycoon were contributed to, see Chapter 5.

4.1 Tycoon

Tycoon [68, 69, 67] is a market-based resource allocation system allowing resource shares to be auctioned out proportionally to users' bids. In short it implements the resource allocation game and the best response agent as described in Section 2.2. Furthermore, Tycoon implements resource virtualization as described in Section 2.1. A user i bids on a resource by specifying a total bid size b_i and a bidding interval t_i. The bid is then calculated as $\frac{b_i}{t_i}$. If the total size of a resource is R, then r_i, the total amount of resource allocated to user i over a period P, is

$$r_i = \frac{\frac{b_i}{t_i}}{\sum_{j=0}^{n-1} \frac{b_j}{t_j}} R \tag{4.1}$$

If q_i is the amount of the resource consumed by user i in period P, then i pays at a rate of:

$$s_i = \min(\frac{q_i}{r_i}, 1)\frac{b_i}{t_i} \tag{4.2}$$

Note that payments are made, as common for a utility, per time unit on a continuous basis. A resource exposes its price y as the sum of all the currently paid rates on

a resource, $\sum s_i$. A worst case bound for this price is thus the sum of all bids normalized by time, $\sum b_i/t_i$, which can be used as a conservative assumption in predictions.

To determine the best response function yielding a distribution of bids across a set of resources given a total budget and the resource prices, Tycoon implements the best response algorithm [41] that solves the following optimization problem for a user: from a set of n resources pick the set $\{x_{ij}...x_{in}\}$ that

$$\text{maximizes } U_i = \sum_{j=1}^{n} w_{ij} \frac{x_{ij}}{x_{ij}+y_j} \text{ subject to } \sum_{j=1}^{n} x_{ij} = X_i, \text{ and } x_{ij} \geq 0 \quad (4.3)$$

where U_i is the utility of user i across a set of resources, w_{ij} is the preference of machine j as perceived by user i (for example the CPU capacity of the machine), x_{ij} is the bid user i should put on host j, y_j the total of all current bids or the price of host j, and finally X_i is the total budget of user i.

The prior beliefs of the demand used as input to the algorithm are represented by the y_j values, which are reported by all resource auctioneers after each completed bidding and accounting cycle, typically once a minute. However, users are allocated their appropriate shares instantaneously after bidding. Furthermore, a bid may be placed at any time.

4.2 Market-Based Grid Resource Broker

As part of our investigation of service-level management in Grid systems we developed a Grid broker on top of Tycoon (see Figure 4.1), which allows Grid HPC users to prioritize their jobs in an incentive-compatible way by transferring Tycoon credits to the broker. The broker receives credits from the user and automatically creates local virtual host accounts to execute the job on the resources picked by the best response algorithm described in Equation (4.3). The jobs run on each host at a service level determined by the Tycoon allocator proportional to the bid determined by the best response algorithm. The actual enforcement of the service level is done by the virtualization engine in Tycoon, which is Xen [34]. An important addition to Tycoon that we also developed was a tool for Grid users to predict future prices of resources in order to make better decisions on how much money should be spent on a resource to get a certain performance level (see Section 4.3 for more details).

The user interface of the broker uses the Nordugrid ARC *meta-scheduler* [110] which in turn is based on the Globus Toolkit [46], both extensively deployed in production Grid systems worldwide.

Our Grid market broker also performs a number of job related tasks on behalf of the user, and it is important to note that these tasks are all performed as a result of the user transferring additional money to the broker to maintain the incentive-compatible properties of Tycoon in the Grid market. Some of the tasks we added to the broker are enumerated here:

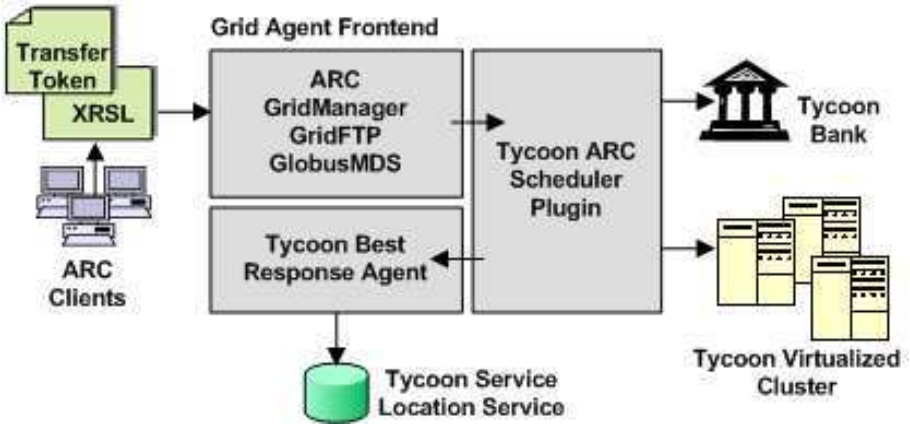


Figure 4.1: Tycoon Grid Market Architecture. Grid clients attach a transfer token to their ARC metascheduler XRSL job description before submitting the job with the regular ARC client. The job is received at the back-end Tycoon cluster and is first parsed, authenticated and then pre-staged (input transferred to the back-end access node) by the ARC/Globus middleware. The Tycoon node scheduler then picks the nodes to run on based on the funding provided and charges the bank account associated with the job (transfer token). The individual bids on nodes are determined using the best response algorithm, and the current node prices obtained from the Tycoon SLS service. Finally a virtual machine is created to host each subjob and the job script is run.

- **Job Payments.** A Grid user can pay for her jobs by attaching a transfer token to the job submission. The transfer token is a receipt of a credit transfer from the user account to the Grid broker account. The token maps the Grid identity to a Tycoon bank account user identity. The token can also be issued by a third party to clients who do not have any Tycoon components installed, and thereby use the token as a gift certificate. More commonly, though, the token will be created as part of the job submission process on the client side. This design allows the broker to also utilize the full VO-authorization management support provided by the Grid job manager, a.k.a. the gatekeeper. It could be seen as a combination of identity-based authentication, policy-based VO authorization and then finally capability-based authorization in the Tycoon layer.

- **Price Prediction.** Future prices, performance estimates, at certain guarantee levels are communicated to the user in order to give guidance as to how much a job may cost (see Section 4.3).

- **Job Boosting.** A job that is running slower than first anticipated and that is not likely to meet its deadline can be boosted with initial funds without

resubmitting the job.

- **Job Snapshots.** It is hard to tell from a generic infrastructure perspective, how close the job is to completing and whether it is therefore likely to meet its deadline. We therefore added an interface that allows users to get snapshots of their output files while the job is still running.

- **Job Stage-In, Stage-Out**. Input files are seamlessly transferred from the user to the compute node that was selected to run the job, and output files are gathered and transferred back to the user when a job has completed.

- **Multijob Support**. If multiple jobs are to be run at the same time it is preferrable to submit them all at once and let the best response algorithm take care of the optimal distribution and funding of them on each host. We therefore provide support for submitting one Grid job with different inputs for each individual compute node subjob.

- **Runtime Setup**. We use the YUM[1] installer to automatically provide a wide range of installation packages that may optionally be installed on demand before the job is run to customize the compute node configuration easily for the specific application needs and dependencies.

- **Bank Account Isolation and Refunds**. Each Grid user using our broker gets a separate local bank account used to fund end refund jobs. This improves accounting and isolation of individual user jobs, and allows the Grid broker to maintain the Tycoon property that users only pay for what they use.

- **Virtual Machine Recycling**. A user can create at most one virtual machine per compute node at any point in time to avoid the user competing with herself, and creating a higher price of the resource than necessary. It further helps in terms of avoiding starvation problems on a machine, since there are physical memory limitations in the virtualization engine of maximum number of virtual machines that can be served. In general the more slices a machine can handle the better effect does the market approach have. However, there is also substantial overhead incurred when creating and starting up a new virtual machine and installing the runtimes, so we allow the user to reuse virtual machine runtimes between job submissions (but not scratch space), but only if the idle virtual machine was not *outcompeted* by other users in the meantime. The reason why we do not support scratch space reuse is that the VM reuse should be transparent and only be detectable by means of a perceived performance improvement.

- **Seamless Backend Integration**. In order to allow seamless backend deployment of the Tycoon Grid scheduler into any Grid middleware job submis-

[1] Yellow dog Updater, Modified. http://linux.duke.edu/projects/yum/

sion infrastructure we provide the same command line interface as OpenPBS [2], one of the most common cluster job submission toolkits.

4.3 Market Prediction and Admission Control

The Proportional Share Market Prediction and Admission Control Model (PS-MP/AC), presented in Section 3, was implemented as a set of cross-platform Python classes. The overall design of PS-MP/AC is depicted in Figure 4.2. The modular design increases generality and component reuse. The *Statistics Collection* classes can handle any stream of time series data, such as prices from any of the resources in our computational market. The *Density Estimation* classes can be reused by both the *Risk Probing* class supporting the user and the *Admission Control* class serving the provider. Furthermore, only the Risk Probing and Admission Control classes are aware of the underlying proportional share allocation mechanism.

Below we discuss the implementation of the five main components of the system.

- **Proportional Share Allocation.** This part comprises all the components of the core Tycoon resource allocation system. Tycoon implements a proportional share spot market with best effort guarantees enforced by decentralized auctioneers deployed at every physical machine in a cluster hosting virtual machines. The auctioneers publish information about their status and price to a centralized service location service. The service location service is then queried by best response agents which implement the game theory based best response optimization described in Section 4.1.

- **Statistics Collection.** Each auctioneer has a set of classes collecting historical price values for all the resources supported. The historical values are summarized into a parsimonious set of statistics that are published to the service location service. The statistics include a list of statistical moments in different time intervals and a history of previous moments. A typical configuration is to collect the current mean and variance for the last 5 minutes, last hour, last day, last week, and last month; and additionally three historical values of these moments that get refreshed periodically. For example the average price of CPU for the last 7 days, as well as the averages for the last three weeks updated every 7 days are collected. The current average calculated is thus only the same as the most recent historical value right after the historical update has occurred. This semantic difference between current and historical summary statistics becomes more important the longer the time interval is. The statistics collector must be aware of the combinatorial explosion of statistics maintained. Each host in the cluster has five distinct resources, each resource supports four time intervals, with three historical data points each, and each data point is a list of moments collected. A service location service serving a cluster of 1000 machines, would thus need to maintain

[2] Open Portable Batch System. http://www.openpbs.org

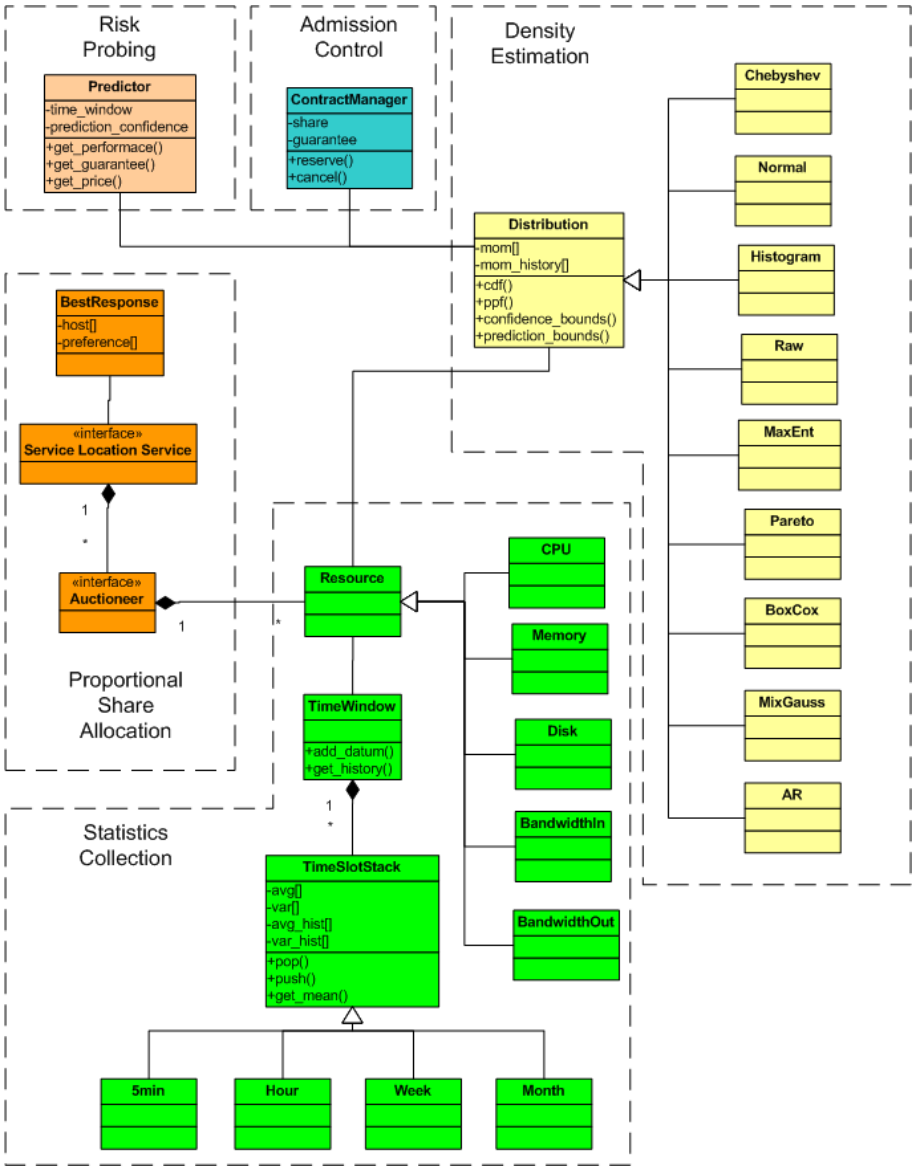


Figure 4.2: Market Prediction and Admission Control Design.

$1000 \cdot 5 \cdot 4 \cdot 3 \cdot 2 = 120000$ price points to support the first two moments or 180000 price points to support the first three moments. Assuming that all hosts report their values once every minute to the service location service, the service needs to be capable of handling 2000 or 3000 price points per second when collecting two or three moments respectively. Density estimators which only rely on the first two moments are thus preferred for scalability reasons. We have deployed this collection method in a Tycoon cluster with up to 400 hosts without observing any scalability issues, and simulations confirm that it will scale well beyond that. An excerpt from the time horizon slots implementation in Python, which adds a new datum to the historical values and calculates the average across a slot series (cf. Equation (3.4) and Equation (3.5)) can be seen in Listing 4.1. Note that the implementation applies linear smoothing of the first and last element in a slot series (which has an additional element) to calculate the moment continuously.

Listing 4.1: Functions to add a new datum and calculate moments.

```
# S - slot series
# c - price snapshot (datum)
# m - moment
# S['i'] - number of items in current slot
# S['d'] - dimension of series
def add_datum(self, S, c, m):
        S['i'] += 1
        S['sum'] += c**m
        S['avg'][-1] = S['sum']/S['i']
        if S['i'] == S['d']:
            S['i'] = 0
            S['sum'] = 0
            S['avg'].pop(0)
            S['avg'].append(0.0)

def mu(self, S):
        n = len(S['avg'])
        weight = S['i']/float(S['d'])
        return ((1 - weight) * S['avg'][0] + \
                sum(S['avg'][1:-1]) + \
                weight * S['avg'][n - 1])/(n - 1)
```

- **Density Estimation.** The purpose of our probability density estimator is to continuously construct approximations of cumulative density functions and their inverses. Empirical representations as well as histogram representations suffer a high performance penalty if the number of evaluations per model constructed is high. As we would like to encourage user risk probing, faster functional (arithmetic as opposed to algorithmic) representations of the density

function are preferred. A large number of distributional models were implemented as seen in Figure 4.2. The Chebyshev model described in Section 3 rendered the most reliable predictions, verified by means of HPC workload trace experiments and analyses, while meeting our statistics collection scalability requirements. The Normal estimator fits the first two moments to the standard normal distribution, The Histogram (value binning) and Raw (empirical distribution) estimators are benchmark estimators making use of the entire time series sample (which is not distributed in the live system for the scalability reasons mentioned above). The MaxEnt estimator uses three moments to optimize the model entropy through gradient descent, BoxCox performs a BoxCox transformation to remove skewness and non-stationarity in the second moment, and then applies the normal distribution estimator. MixGauss fits a Gaussian mixture model using an expectation maximization algorithm for a predefined number of mixtures, and the AR estimator implements an autoregressive model predicting a set of point values into the future and then taking the empirical distribution of these point values. The Chebyshev PPF is calculated as seen in Listing 4.2 (cf. Equation (3.14)).

Listing 4.2: Chebyshev percent point function.

```
# p - percentile
# mom non-central moments
# mu - first non-central moment (mean)
# mu2 - second non-central moment
# sigma - standard deviation
def chebyshev_ppf(p, mom):
    mu = mom[0]
    mu2 = mom[1]
    sigma = sqrt(mu2 - mu**2)
    cantelli = sqrt(1/(1 - p) - 1)
    petunin = mu + sigma * sqrt(4/(9 * (1 - p)))
    k = (petunin - mu)/sigma
    if k > sqrt(8./3):
        return petunin
    return mu + sigma * cantelli
```

- **Risk Probing.** A resource consumer uses the density estimator in four different ways: to calculate the probability that the price drops below a certain value (CDF), to get the price corresponding to a certain likelihood of a price minimum (PPF), to get an interval within which a certain percentage of the values will fall (confidence bounds), and finally to estimate the uncertainty of the prediction model itself (prediction bounds). Risk probing is done by means of three operations mapping performance, q, guarantees, g, and price c. The operations with their corresponding mappings are *get_performance()*:

$(g \times c) \rightarrow q$, *get_guarantee()*: $(q \times c) \rightarrow g$, and finally *get_price()*: $(q \times g) \rightarrow c$. The implementations of these three operations follow intuitively from the formulas given in Equations (3.15), (3.16) and (3.17). The typical reason for probing is to determine the diminishing returns of spending more money on a computation. Capturing risk explicitly with different percentile points and prediction confidence levels, as described above, allows a provider to efficiently multiplex between users with different resource requirements **and** risk attitudes. The prediction intervals are calculated empirically as seen in Listing 4.3. An upper bound is created based on the distribution of past predictions. This distribution is then fit to a CDF/PPF model to estimate a percentile which is the same as the confidence of the prediction (the probability of the probability or model uncertainty). Note that the error distribution is fit separately to allow for e.g. a perfect model and white noise error assumption. By default the prices are assumed to be distributed using our Chebyshev model and the errors in predictions are assumed to be Gaussian. Further note that the implementation allows both an arbitrary number of historical values and an arbitrary number of moments to be used when calculating the bound.

Listing 4.3: Empirical prediction bound function.

```
# moms - historical series of non-central moments
# conf - confidence level of prediction bound
# p - percentile
# order - max order of moments collected
# m - moment
def prediction_bound(moms, conf=.9, \
                     p=.95, \
                     ppf=chebyshev_ppf, \
                     err_ppf=normal_ppf):
    weight = 1./len(moms)
    order = len(moms[0])
    mom_avg = []
    for mom in moms:
        ppf_val = ppf(p, mom)
        for m in range(1, order + 1):
            if len(mom_avg) < m:
                mom_avg.append(weight * ppf_val**m)
            else:
                mom_avg[m - 1] += weight * ppf_val**m
    return err_ppf(conf, mom_avg)
```

The input to the function is a list of historical statistical moments that are used to run individual predictions and then capture the spread of errors ob-

tained. Note that the actual outcome corresponding to a predicted moment appears in the subsequent historical value, and we can thus focus on calculating the spread of predicted values as a means to determine the stability of the predictions and as a proxy for model uncertainty.

- **Admission Control.** A resource provider uses the same density estimation primitives as a resource user. The main question the provider needs to answer is however, how much profit will be lost if a resource request that cannot be preempted is accepted. The likelihood that competing consumers will come along and increase the price (and thereby increase the provider's profit) can be directly obtained from the cumulative distribution function of the current price. Conversely if the provider wants to set a price that guarantees with a certain likelihood that no losses will be made if the consumer is admitted the inverse or the percent point function can be used. Refusing all requests from users with a budget lower than a certain percent point guarantee is an effective way of filtering out low priority users during high demand periods. However, one might conceive periods when there simply is not enough capacity to serve all users' needs even with statistical guarantees. This kind of saturation can lead to price inflation and reduced predictability as was shown in [102]. Explicit capacity management will then have to complement the statistical guarantees. Capacity management in a proportional share allocation model is very straightforward. A new request for a share is simply rejected if its bid will cause the promise of a share to an already existing user to be violated. A capacity manager component in our access control service enforces this restriction across a set of hosts and also allows users to set performance limits across hosts. However, if consumers are given this access control service without increasing the price of the service beyond the current spot market price, the provider will make a certain profit loss. In our experiments we saw that a 60 or 70 percentile of the price could be used instead of the spot market price and still yield high system efficiency.

Chapter 5

Contributions

In this chapter, the published contributions related to this thesis are presented. First we discuss the six publications that form the main contribution of this thesis[1], and then additional contributions are summarized. The contribution level of the author of this thesis is given within parenthesis in each paper headline.

5.1 Main Contribution Papers

The six main publications represent the evolution of approaches leading to the design and implementation of the Proportional Share Market Prediction and Access Control model proposed in this thesis. They are presented in chronological order below.

Paper 1: Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications (90% Main Author)

In conference paper[2] [104], we present the design, implementation and evaluation of a Grid resource market for HPC users. This market is further supported by a suite of prediction models and tools to allow users to spend their money more efficiently in the market to meet their requirements.

Our solution is to integrate a Grid meta-scheduler and resource manager with Tycoon. We thus maintain the cross organizational VO-supported PKI security model and the support for high-volume data transfers to stage in and out jobs to compute nodes seamlessly. At the same time we leverage the economically efficient and fair Tycoon model including the best response scheduler and the proportional share allocator. The integration is achieved by two means, a) a transfer token used

[1] Attached to the end of this thesis in full.

[2] Published in the proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing, Paris, France, June 2006.

as a lightweight contract simulating a *gift certificate* to purchase resource shares, and b) a broker that receives the transfer token attached to the jobs to be submitted, and funds and executes the jobs according to the best response bidding algorithm.

The experimental results were obtained by running a Bioinformatics application, from SweGrid, in a cluster managed by the Tycoon Grid Market. We showed that a continuous service level (as opposed to the binary one in SweGrid) proportional to the funding of the job could be offered. Account management is also simplified in our Grid Market, as a result of the local accounts being created on demand and then dynamically configured to match the service level purchased. Finally, rights delegation is seamless as it only involves transferring Tycoon credits between user accounts, and resources get credits when users run jobs. These credits can then in turn be used to submit jobs. Therefore, our Grid Market has the desirable property of offering closed-loop (self-sustained) sharing of resources among peers, true to the foundational idea of the Grid.

Paper 2: Evaluating Demand Prediction Techniques for Computational Markets (90% Main Author)

In invited workshop paper[3] [104], a series of prediction techniques are evaluated using workload demand traces from the PlanetLab network. The techniques studied included probability density estimation based on histograms, maximum entropy, normal distribution approximation, and autoregressive forecasting. Paper 1 assumed a normal distribution of demand. This paper, on the other hand, addresses and highlights the problem of non-symmetric demand distributions, common in computational workload traces. Additionally, this paper introduces the general formulation of long tail percentile predictions as a means to mitigate risk of performance degradation of market-based resource allocations.

A PlanetLab trace containing 5-min snapshots of overall demand (Unix uptime load) across all machines in the network during a 2-month period was used as basis for the evaluation, and demand densities (cumulative distribution functions) in time windows from 2 hours to 3 days were approximated and predicted.

The conclusion of this paper was that the maximum entropy technique captured the distribution asymmetry better than the other techniques, when predicting long tail percentiles for demand. However, the algorithm had convergence problems when predicting long time windows.

[3]Published in the proceedings of the 3rd International Workshop on Grid Economics and Business Models, Singapore, May 2006.

Paper 3: A Statistical Approach to Risk Mitigation in a Computation Market (90% Main Author)

In conference paper[4] [103], we propose a general prediction model for statistics collection and risk mitigation probing, based on the long tail prediction formulation presented in Paper 2. The Chebyshev inequality predictor is introduced, and evaluated against a normal distribution predictor and a histogram based predictor. The maximum entropy convergence problem previously mentioned and the high overhead of the inherent optimization rendered the MaxEnt technique presented in Paper 2 inferior to the other predictors in terms of scalability in the more realistic on-line predictions performed in this study.

The study contains two parts. In the first part distributional characteristics of demand from four computational clusters, PlanetLab, San Diego Super Computer Center, Ohio Super Computing Center and KTH Parallel Computer Center are analyzed. We show that all traces (based on demand from job submissions during a one year period) show signs of distribution asymmetry; time correlations and long term dependence; and heteroskedasticity[5] and long distribution tails. These properties are then in turn addressed in the prediction model presented using distribution-free percentile bound estimations, moving prediction time windows, and empirical prediction bounds.

The main result from the predictor evaluation is that the Chebyshev model produced the most reliable predictions in terms of accurate percentile bounds and high success rates of prediction bounds. We also performed a simple prediction experiment showing that the overall performance of a batch of parallel jobs can increase by 20 per cent if demand prediction is applied when scheduling jobs as opposed to basing the scheduling decision on the current demand. An experiment measuring system overhead was also conducted, which showed that the implemented predictor scaled very well.

Paper 4: Prediction-Based Enforcement of Performance Contracts (90% Main Author)

In workshop paper[6] [102], the notion of a hybrid reservation and spot market realized by proportional share, and statistical admission control is introduced. The main point of the paper is to show that statistical guarantees are not sufficient when the market is saturated by users demanding high guarantees, without the presence of best-effort users who do not have any guarantee requirements. Furthermore pure statistical guarantees can lead to price inflation, and more skewed and thus less predictable demand.

[4] Published in the proceedings of the 16th ACM International Symposium on High Performance Distributed Computing, Monterey, USA, June 2007.

[5] Variance changes over time.

[6] Published in the proceedings of the 4th International Workshop on Grid Economics and Business Models, Rennes, France, August 2007.

To address these issues we implement a mechanism where the provider offers absolute guarantees in limited time windows at prices set by the statistical models presented in Paper 3 (Chebyshev bound predictions). The idea is that the users need to pay a premium price for getting their jobs admission controlled with more reliable guarantees. This premium price is based on expected demand so that higher guarantees cost more when demand fluctuates more.

Simulations show that the admission controller can provide guarantees more accurately during high contention (guarantee saturation) periods while avoiding inflation.

Paper 5: Autoregressive Time Series Forecasting of Computational Demand (100% Single Author)

In technical report[7] [97], we evaluate techniques to predict demand using Autoregressive Integrated Moving Average (ARIMA) models. The motivation for this evaluation was that we saw problems with the exponential smoothing of statistical moments, introduced in Paper 3, in the live market during regime shifts, when the demand level changed. The standard procedure for constructing and evaluating ARIMA models is used to study different moment smoothing techniques for the statistics collected in our model before it is sent to the risk probing predictor. Traces both from PlanetLab and our proportional share market (Tycoon) are used in the evaluations.

First the general model structure including regression, integration and moving average orders, is evaluated using a sample part of the data. This model structure is then instantiated with parameter values estimated continuously from a separate evaluation part of the data, and used for predictions in different time windows. The predictions performed are designed to mimic the on-line predictions performed in our computational market.

The results of this study show that a random walk model where no moment smoothing is done performs best for one-step-ahead predictions, whereas ARIMA(1,1,0) and exponential smoothing models perform better for two- and three-step-ahead forecasts. As a result we perform no smoothing of values by default in our PS-MP/AC model but use the average values calculated from the time slot tables directly. An interesting secondary result from this study was that the statistical properties of the Tycoon trace and the PlanetLab trace were very similar, which puts more confidence in previous studies where we used PlanetLab traces as a proxy for demand in our computational market, e.g. Paper 2 and Paper 3.

[7] Published in the arxiv.org archive in November 2007.

Paper 6: Admission Control in a Computational Market (90% Main Author)

In conference paper[8][105], we extend the study in Paper 4, on the evaluation of a hybrid reservation and spot market, in four ways. First, experiments are run in our live computational market. Second, the workload is generated based on a careful analysis and construction of a workload model from an extensive HPC trace. Third the economic efficiency and fairness of the system as a whole are evaluated. Finally, experimental and analytical results are used to give quantitative guidelines on how to partition the reservation and spot markets dynamically based on demand.

The experimental setting included a set of clients with the same initial budget and the same number of jobs. Each job was assigned various characteristics such as run time, priority, inter-arrival time, size in CPUs, based on our workload model. The clients then split their budget appropriately based on the expected value of each job. After the run of all jobs, the price paid and the performance obtained of all jobs were recorded, and used to calculate the utility. We also recorded the price fluctuations during all experiment runs to study the correlation between the price dynamics of different admission models. The idea here was that a fair admission model sets prices that correlates well with the demand in a completely unrestricted (best-effort) market.

The results show that the admission controller achieves high efficiency and acceptable fairness during high contention periods in particular with jobs that are inelastic in their performance requirements. Simple functional formulations were deduced to aid both consumers when determining whether to submit their jobs on the spot or reservation market, and providers when deciding how to partition the pool of resources on the different markets based on demand.

5.2 Other Contributions

Other contributions comprise contributions made to a Grid accounting system, an authorization framework and a standard Grid service protocol. These contributions were summarized in the licentiate thesis described in Contribution 11 below, and are not directly related to the prediction and access control problem which is the main theme of this thesis. However, the work on the accounting system, in particular, motivated and identified many of the research problems addressed in the main contribution papers.

[8] Admitted to the 8th International Symposium on Cluster Computing and the Grid, Lyon, France, May 2008.

Contribution 1: A Service-Oriented Approach to Enforce Grid Resource Allocations (90% Main Author)

In journal article[9] [100], we discuss the initial approach of enforcing global resource quotas on a project basis across the SweGrid machines. SweGrid is the Swedish national Grid resource comprising 600 compute nodes distributed across six High Performance Computing (HPC) Centers and interconnected with a 10Gbit/s WAN. Various research projects are allocated CPU quota by the Swedish National Allocation Committe (SNAC), after a peer review of the scientific value of the project and its computational needs. Allocations are administered and renewed on a six-month basis. The problem we are addressing in this work is how the allocations can be enforced in real-time on all of the SweGrid machines in a coherent manner.

The problem is to a large extent a systems integration problem, in that all HPC centers already use their own resource management system and their own accounting and access control policies and tools. We therefore introduced an integration platform based on a service-oriented XML Web services architecture entirely written in Java. The architecture comprises a Bank service, responsible for enforcing the global resource quota and managing project accounts; a Logging and Usage Tracking service, for off-line usage analysis and post-accounting; and finally a Job Account Reservation Manager, which integrates the local site resource manager into the global accounting system.

The most important research contribution from this work is the policy-based access control system, which, at real-time, lets user, resource, and allocation authority policies determine whether a Grid job should be allowed to run on a resource and at what level of service. We call this solution soft real-time allocation enforcement, because resources may not want to strictly refuse access if the quota has been exceeded, but instead downgrade the priority of the job. This model extends the state-of-the art in that a binary service-level is provisioned based on usage history and centrally allocated grants. A higher level of fairness is thus achieved, and problems like denial-of-service attacks and job starvation can be mitigated.

Contribution 2: Service Level Agreement Requirements of an Accounting-Driven Computational Grid (100% Single Author)

In technical report[10] [96], we discuss the requirements obtained after studying the first production deployment of the accounting system presented in Contribution 1 [100]. We more specifically focus on how electronic contracts, a.k.a. Service Level Agreements, can be used to address some of the shortcomings of the existing system.

[9] First edition published in the proceedings of the 2nd ACM International Conference on Service-Oriented Computing, New York City, USA, November 2004. Second edition published in the World Scientific International Journal on Cooperative Information Systems, September 2006.

[10] Published in the NADA TRITA technical report series at the Royal Institute of Technology, Stockholm, Sweden, September 2005.

An enhanced, agent and policy-driven architecture is proposed, where the service levels are determined and enforced in a continuous and automatic way based on mutually signed contracts. The contracts represent a user capability as well as a resource provider obligation, and can thus be used as the basis for access control and service-level configuration.

The main contribution of this paper is the mapping of typical Grid user requirements to an agent-based, contract-driven architecture. The first insight gained from the SweGrid accounting system [100], was that it was very flexible to customize policies of all components, but determining what those policies should be quickly became a non-trivial task for a human actor. Agents could thus use contracts embodying user and provider preferences to optimize user utility, or provider profit and utilization by automatically setting these policies.

Contribution 3: The Design, Implementation, and Evaluation of a Market-Based Resource Allocation System (50% Co-Author)

In manuscript[11] [70], we introduce Tycoon, a market-based resource allocation system for large-scale networks like PlanetLab and the Grid. Tycoon allocates virtualized slices on hosts proportional to user bids. The main focus of this paper is to evaluate and benchmark the economic properties of the Tycoon resource allocation algorithms in a real cluster environment through a set of experiments. We study efficiency, based on the sum of the utilities across all users, a.k.a. as social welfare; and fairness, defined as the level of envy-freeness. Envy in turn is defined as the ratio between the maximum utility a user would get from another user's allocation and the utility of the allocation obtained. An optimally fair system would thus have an envy-freeness value of 1.

It is shown in our experiments that the Tycoon proportional share allocation is more efficient than an equal-share allocation algorithm like the one used in PlanetLab when slicing individual resources in shares. It is further shown that the best response algorithm implemented in Tycoon to distribute bids optimally across hosts yields a higher efficiency than other load balancing algorithms. In terms of fairness our experiments were not able to show clear trends due to noise in the live cluster contributing to increased envy.

The results in this paper confirms previous simulation results and also shows how Tycoon can be used to dynamically trade off winner-takes-it-all and equal-share allocation algorithm properties. In essence, the higher the statistical variance on the bids is, the closer the Tycoon algorithm is to the winner-takes-it-all scheme. If the variance is 0 it is equivalent to an equal-share algorithm.

[11] Manuscript prepared for publication at Hewlett-Packard Laboratories, Palo Alto, USA, May 2006.

Contribution 4: Open Grid Services Infrastructure (OGSI) Version 1.0 (10% Co-Author)

The Global Grid Forum Open Grid Services Infrastructure (OGSI) specification [115] introduces many of the fundamental integration concepts that the Swegrid Account System (SGAS) work is based on. The thesis author contributed the XML rendering and a reference implementation of that specification.

Contribution 5: An OGSA-based Accounting System for Allocation Enforcement across HPC Centers (90% Main Author)

A conference version of Contribution 1 was presented in [99]. It contains some additional Fuzzy Logic experiments and it is based on an earlier Web service integration platform. Contribution 1 also contains some lessons learned from deploying the solution presented in [99] in SweGrid.

Contribution 6: An OGSA-Based Bank Service for Grid Accounting Systems (50% Co-Author)

The Bank service of SGAS is presented in some more detail in the conference paper [37]. The Bank was implemented by a collaborator, but the core Web services infrastructure, and the access control and policy framework were the thesis author's contributions. The overall design of the Bank was also a collaborative effort.

Contribution 7: The Globus Authorization Processing Framework (10% Co-Author)

The SGAS authorization framework was contributed to the Globus Toolkit, and it is the foundation for extended work presented in the workshop publication [109]. Our authorization framework, in turn, borrows many concepts from the XACML architecture [1] and the GGF Authorization Working Group model [76].

Contribution 8: Policy Administration Control and Delegation using XACML and Delegent (10% Co-Author)

SGAS provides a testbed for authorization management rights delegation, in the conference paper [107]. This work is also based on the authorization policy framework developed as part of SGAS, and extends it by integrating a third-party authorization engine as a policy administration and decision point.

Contribution 9: The Philosophy of the Grid – Ontology Theory from Aristotle to Self-managed IT Resources (100% Single Author)

In technical report [95], a philosophical view of the Grid is presented. The main contribution is to relate the concept of Ontologies in the Philosophy of Science community to the use of Ontologies in Computer Science in general and in Service Level Agreement protocols in particular. Ontologies play an important role in policy definition and embodies the universe of discourse used by agents to optimize the users' utility based on their preferences. The discussion in this report shows that work as early as Aristotle had striking similarities to the use of Ontologies today.

Contribution 10: Scalable Grid-wide Capacity Allocation with the Swegrid Accounting System (SGAS) (10% Co-Author)

In journal article [52], the SGAS system is extended with a more scalable and robust Bank based on a novel Naming service authorization scheme. The Globus GRAM scheduler was integrated with the SGAS resource allocator and simulations showed that the SGAS component incurred negligible overhead. The thesis author's contribution to this work was restricted to a consulting role and some text comparing the solution to market-based systems.

Contribution 11: Managing Service Levels in Grid Computing Systems (100% Single Author)

In thesis[12] [98], two approaches to manage service levels in Grid systems are compared. First a policy based quota enforcement approach implemented in the SweGrid accounting system, and second, a market based resource allocation approach implemented in Tycoon. This contribution summarizes the research in Paper 1, and Contribution 1, 2, and 3. The conclusion was that the policy approach was very effective when enforcing a dual service level model, with high and low priority tasks, but made it hard to enforce and offer finer-grained differentiated services. Furthermore, the inherently static pricing model in the policy management approach did not allow load balancing based on demand. These shortcoming were all addressed in the market-based resource allocation approach, but at the cost of non-deterministic run-time performance levels due to fluctuating demand. This conclusion motivated the research on predicting demand in market-based compute farms, which was initially addressed in Paper 1 and subsequently served as the main theme of Paper 2, 3, 4, 5, and 6 presented in Section 5.1.

[12]Published as a Licentiate Thesis at the School of Computer Science and Communication, KTH, May 2007.

Chapter 6

Related Work

In this chapter we summarize related work in five general categories: first, general purpose computational economies; second, Grid market systems; third, computational demand prediction; fourth, economic parallel job scheduling; and finally resource admission control.

6.1 Computational Economies

Computational economies have been used as a mechanism to allocate scarce resources more efficiently as far back as in 1968. Seminal work by Sutherland [113], Nielsen [85], and Ellison [36] share many of the same design concerns (funding policy, closed- or open-loop economies, dynamic or static pricing, etc.) as computational markets being developed today including our work. However, their work was only considering access to centralized resources, such as monolithic supercomputers. They therefore did not face problems which we have to address such as packing parallel jobs efficiently across machines or distributing bids optimally across multiple hosts. Distributed computational markets were popularized by Ferguson *et al.* [43], Miller and Drexler [81], and Waldspurger *et al.* [120].

Spawn [120] was one of the first successful implementations of a distributed computational market, and Tycoon, on which we base our work, is an incarnation and evolution of many ideas presented in that work. Tycoon, in essence, extends Spawn by providing a best response agent for optimal and incentive-compatible bid distribution and host selection, and by virtualizing resources to give more fine-grained control over QoS enforcement. Spawn does not have the demand prediction capabilities which are central to our work. However, the general, continuous bid and proportional share auction architecture is largely the same.

Bellagio [84] uses a centralized allocator called SHARE. SHARE uses a centralized combinatorial auction allowing users to express preferences with complementarities. Solving the NP-complete combinatorial auction problem results in an optimally efficient allocation. The price-anticipating scheme in Tycoon is decen-

tralized, i.e. runs an auction at every single host, and does not explicitly operate on complementarities. The efficiency in our system may thus not be as high but all the overhead and computational complexities of combinatorial auctions, as well as the issues with strategic users gaming the mechanism are avoided [70].

REXEC [23] is a proportional share market system that is very similar to our system architecturally. However, contrary to our system, it only allows bidding on the CPU resource, it is not work conserving, and it does not utilize operating system virtualization. The accounting mechanism is also less flexible than in Tycoon in terms of allocation distribution, and fund deduction.

In [119] Waldspurger proposed lottery and stride scheduling algorithms based on proportional share resource allocations, similar to the mechanisms we use. The abstractions are different and there is no notion of markets or price setting mechanisms in Waldspurger's model. However, our statistical methods are general enough to fit both systems based on REXEC and those based on lottery and stride scheduling, thanks to their proportional share semantics.

Other related approaches using computational economies are described in [94, 78, 51, 22, 112, 17].

In summary, our system (Tycoon) provides a novel virtualized, proportional share market-based allocator for different localized resources, (such as CPU, disk, and memory), across distributed hosts.

6.2 Grid Market Systems

Grid market designs were pioneered by Buyya *et al.* [14] in the Nimrod/G project. Nimrod/G schedules bag-of-task Grid jobs based on resource prices set by the providers and uses dedicated machines for all running jobs. Our price-setting mechanism, on the other hand, allows the price to dynamically follow the demand, and resources are sliced in virtual machines automatically configured for each job based on users' resource requirements, spending preferences and risk attitudes.

Faucets [63] is another framework for providing market-driven selection of compute servers. Compute servers compete for jobs by bidding out their resources. The bids are then matched with the requirements of the users by the Faucets schedulers. Adaptive jobs can shrink and grow depending on utilization and prioritization. QoS contracts decide how much a user is willing to pay for a job. The main difference to our work is that Faucets does not provide any mechanism for price setting. Further, it has no banking service, use central server based username-password mechanisms, and does not virtualize resources.

G-commerce [128] is a Grid resource allocation system based on the commodity market model where providers decide the selling price after considering long-term profit and past performance. It is argued and shown in simulations that this model achieves better price predictability than auctions. However, the auctions used in the simulations are quite different from the ones we use in our work. The simulated auctions are winner-takes-it-all auctions and not proportional share, leading

to reduced fairness. Furthermore, the auctions are only performed locally and separately on all hosts leading to poor efficiency across a set of host. In our work the best response algorithm ensures fair and efficient allocations across resources. An interesting concept in G-commerce is that users are allocated budgets that may expire, which could be useful for controlling periodic resource allocations and to avoid price inflation. The price-setting and allocation model differs from our work in that resources are divided into static slots that are sold with a price based on expected revenue. However, the preemption and agile reallocation properties inherit in the bid-based proportional share allocation mechanism employed in our system to ensure work conservation and prevent starvation are missing in the G-commerce model.

A large number of European research projects have investigated Grid markets recently. The UK eScience Grid Markets project[1] was one of the first projects in this area. It aimed at extending existing standard Grid protocols such as OGSA/OGSI with capabilities to account and charge for services. A Grid economic services protocol (GESA) was proposed to the Global Grid Forum (GGF) as a result of this work. The Grid Markets work was focussed on charging for services, whereas we charge for resource usage. It was akin to our previous work on Grid Accounting in that it had more of a standards-based systems integration and middleware focus. Our contribution, on the other hand, is more focused on the modeling, design, implementation and evaluation of economic and statistical mechanisms for computational markets.

The EU GridEcon project[2] is very similar in spirit to our work in that it designs economic models for Grid software systems. Our work is more experimentally driven and it is also more focussed on demand prediction models.

The EU SORMA project[3] aims to develop methods and tools for market-based allocation of resources in open self-organizing Grid environments. The mechanisms proposed in our work could be used to enhance predictability and to offer differentiated service level guarantees in such environments.

The EPSRC GRAIL[4] and Grid Market[5] projects in the UK work on applying stochastic techniques to set verifiable performance constraints, and applying Peer-to-Peer protocols to set prices in Grid environments respectively. Our approach is to merely approximate upper bounds of performance using simple probabilistic laws not to formally verify constraints. Furthermore, our system is not strictly Peer-to-Peer because we use central bank and service location services to simplify resource search and accounting.

Additional Grid Market models are described in [25, 130, 15]. A comprehensive survey of Grid market systems is provided in [131].

[1] http://www.lesc.ic.ac.uk/markets/.

[2] Grid Economics and Business Models. http://www.gridecon.eu/.

[3] Self-Organizing ICT Resource Management. http://www.iw.uni-karslruhe.de/sorma

[4] Grid Enabled Performance Analysis using Stochastic Logics. http://aesop.doc.ic.ac.uk/projects/grail/.

[5] Market Models for Grid Computing. http://aesop.doc.ic.ac.uk/projects/grid-market/

To summarize, our Grid market is unique in employing a user-exposed best response agent on top of a proportional share allocation mechanism and thereby allowing independent decentralized schedulers.

6.3 Computational Demand Prediction

Many prediction models targeted at Grid environments, including ours, have been inspired by the techniques used in the widely deployed Network Weather Service (NWS). Wolski *et al.* describe how they designed and provide an evaluation of the Network Weather Service in [127]. NWS is mainly designed for monitoring compute jobs in large-scale Grid deployments. Our work differs from NWS in both how statistics are collected and stored and how predictions are computed. NWS uses a multi-service infrastructure to track, store and distribute entire time-series feeds from providers to consumers via sensors and memory components (feed history databases). Our solution only maintains summary statistics and therefore is more light-weight. A separate persistent storage or searching infrastructure is therefore not needed in our system. For predictions, NWS uses combinations of simple moving average models with static parameters, selected based on mixture-of-experts heuristics. We use more general predictors that can handle any dynamics and adapt their parameters automatically. In addition, the focus in [127] is on predicting queue wait times, whereas we focus on predicting actual demand or future prices.

Brevik *et al.* [13] present a Binomial Method Batch Predictor (BMBP) complementing NWS [127]. The approach is to assume that each observed value in the time-series can be seen as an independent draw from a Bernoulli trial. The problem is that this does not account for time correlations, which we have found to be substantial in our analysis. Brevik addresses the correlation issue by first detecting structural changes in the feed when BMBP generates a sequence of bad predictions and thereafter truncating the history which the predictor model is fit against. Our approach is to leverage the correlations by using biased samples of the most recent time intervals, which results in dynamic adaptation of structural changes in the feed. The problem of monitoring and fixing prediction problems *a posteriori* as in BMBP is that the detection mechanism is somewhat arbitrary and a structural failure of the model could result in great losses, which could defeat the purpose of providing risk mitigating predictions [79].

MacKie-Mason *et al.* [77] investigate how price predictors can improve users' bidding strategies in a market-based resource scheduling scenario. They conclude that simple predictors, such as taking the average of the previous round of auctions, improve expected bidder performance. Although the goal of this work is similar to ours, they investigate a different combinatorial allocation scenario where first price winner-takes-it-all auctions are employed, as opposed to the proportional share allocation in our work.

Another use of economic predictions is described by Wellman *et al.* [124], where

bidding agents use the expected market clearing price in a competitive or Walrasian equilibrium. They employ *tatonnement* which involves determining users' inclination to bid a certain value given a price-level. Wellman *et al.* compare their competitive analysis predictor to simple historical averaging and machine learning models. They conclude that strategies that consider both historical data and instance-specific data have a competitive advantage. The conditional probability of price dynamics given a price-level could serve as additional useful information in our model. However, this is probably impractical in large-scale systems with users entering and leaving the market at will, and with large real-valued price ranges, so we assume this behavior is incorporated in the price history itself.

Catallactics is an economic theory for reaching market prices, which is very similar to the tatonnement process but with more focus on decentralization and individual self-interested actors. It was applied by Ardaiz *et al.* [4] to coordinate service provisioning in peer-to-peer networks. Simulations showed that the Catallactic approach made service provisioning less sensitive to node dynamics. However, this approach exhibits the same challenges as the work by Wellman *et al.* discussed above; it is complex to implement in a scalable way in a real system. Client and server implementations of a catallactic system are also more involved and require both broadcasting protocols and heuristics combined with machine learning techniques to be employed in order to support price negotiations. Contrast this procedure to the proportional share mechanism we employ, where prices are set instantaneously and deterministically given a locally measured level of demand, using a single network roundtrip.

Oppenheimer *et al.* [86], like us, analyze PlanetLab resource usage and further evaluate usage predictors and conclude that mean reverting processes such as exponential smoothing, median, adaptive median, sliding window average, adaptive average and running average all perform worse than simple random walk predictors and, what they call, *tendency predictors* which assume that the trend in the recent past continues into the near future. They further notice no seasonal correlations over time due to PlanetLab's global deployment. We do see some seasonal correlations in our initial time series analysis but they are not significant enough to take advantage of in predictions. Further, our evaluation approach follows the traditional ARIMA model evaluation method, and we provide a statistical test to verify and compare prediction efficiency. One major difference between our studies and thus also the conclusions is that Oppenheimer et al. only considered one-step ahead predictions whereas we also consider two, and three-step ahead predictors to do justice to the models considering correlations beyond the last observed step.

Our prediction interval calculation was inspired by Haan and Meeker [56] but they also assume that random independent samples are drawn and that a large number of sample data points are used to yield tight prediction bounds. Neither of these two assumptions are true in our scenario. Our calculation of the prediction interval can be seen as more in the spirit of the simple empirical intervals proposed by Williams and Goodman [126]. Their empirical source is the previous sample point, whereas, we use summary statistics as input to the empirical predictions.

This allows us to cover larger prediction horizons with greater confidence using fewer data points.

Our focus on estimating skewness risk and measuring long range dependence was inspired by the work by Mandelbrot on modeling risk in volatile financial markets [80] and Hurst's seminal work on predicting floods [61]. The fat-tail behavior of the demand volatility we observed in HPC traces fits well with the volatility Mandelbrot has seen in the cotton-price, Deutschmark-Dollar exchange rate, and the stock price market dynamics, which he calls *wild* randomness or chance.

To summarize, our focus on providing both computationally efficient and economically efficient predictions sets our work apart from other predictors.

6.4 Economic Parallel Job Scheduling

End-user centric and utility-based batch job scheduling was introduced in Chun and Culler [24], who present a performance analysis of three different scheduling algorithms, FirstPrice (priorities paid for on centralized auction market), SJF (short jobs have priority), and PrioFIFO (three priority queues with different prices set statically) based on aggregate user utility. The utility function used is designed to decline lineary with completion time. FirstPrice outperforms both SJF and PrioFIFO significantly for highly parallel jobs. PrioFIFO was sensitive to changes in demand and deteriorated in performance if the wait time in the most expensive queue was long. In Chun's and Culler's work job valuations followed a bimodal normal distribution representing high (20 per cent of all jobs) and low (80 percent) valued jobs. Our work differs from this work in that our results are independent of which scheduling algorithm is used, our workload is constructed by carefully modeling real traces, and our underlying allocation mechanism is a continuously cleared decentralized spot market auction. To compare the results between economic and non-economic algorithms, Chun and Culler do not factor the price paid for a resource into the utility function. They further assumed that the IAT and runtime distributions were normal, and that the CPU distribution was uniform. In our workload analysis we found these distributional assumptions to be unrealistic. Furthermore, we run experiments in a real cluster with a real economic market, as opposed to simulations, and we take the price but not the completion time (not significant the way the experiment was set up) into account in our evaluation. We also use both economic and system metrics to evaluate or approach, and we focus on resource allocation (selecting a host to run on), rather than on scheduling (selecting a job to run). Furthermore the underlying auction mechanism we use in Tycoon is quite different, where proportional shares are bid for on a spot market which is continuously cleared and has unrestricted preemption. These differences are also apparent in extensions of Chun's and Culler's work [62, 93, 5] that study more elaborate scheduling algorithms and utility functions that take resource price and provider profit into account.

In summary, the combination of reservation and spot market pricing with sta-

tistical guarantees is novel and sets this work apart from other microeconomic systems that control job performance in shared clusters for parallel jobs, such as [120, 111, 128, 15, 124, 63, 22].

6.5 Resource Admission Control

There is a substantial body of work on Internet Protocol quality-of-service enforcement or traffic engineering, represented by the two IETF specifications IntServ [11], and DiffServ [8]. The IntServ specification takes the approach of reserving paths for individual users, and thus does not scale as well as the DiffServ approach, which is based on marking individual packets with different *per-hop behaviors* in a stateless and decentralized architecture. We are facing the same issues and tradeoffs when allocating computational resources across large distributed systems. However, thanks to new virtualization technology and the fact that many of the resources are localized (e.g. CPU, memory, disk) we found it worthwhile to revisit the reservation concepts.

Wang [121] gives an overview of lessons learned and the pros and cons of the reservation approach which can be implemented with IntServ versus the proportional share approach which can be built on top of DiffServ. The conclusion was that fixed allocations over a point-to-point path incur too much overhead for most of the web traffic, it is difficult to determine the resource requirements *a priori*, inter-ISP relationships make end-to-end reservations complicated, and traffic policing breaks down in the event of partial allocation failures. All of these factors result in many IP reservation providers over-provisioning their network capacity, leading to poor utilization.

Knightly and Shroff [65] provide an evaluation of the different admission control algorithms available for IP traffic shaping. The dilemma of choosing between denying access to flows that might have been served and thereby cause underutilization and serving requests that might break existing QoS contracts makes it hard to use coarse statistical bounds and too simplified assumptions about traffic flow distributions. Put differently, both accuracy maximization and risk minimization are desired. The algorithms that accounted for economies of scale and not simply looked at the statistical properties of individual flows were shown to perform much better on average. Again, our admission control decision differs from the IP flow one, in that we can, through virtualization, more directly enforce that an admitted request stays within its bounds. Our decision is thus more about making sure that the provider does not lose out on utilization or profit by admitting low priority tasks prematurely.

Admission control as a means to avoid service degradation of high priority tasks during overload has also been extensively studied in the context of Web servers, as exemplified in [38, 21, 73]. Priorities of individual requests are either set explicitly in the server configuration or inferred implicitly by the admission algorithm. Our admission controller, on the other hand, gives users an incentive to specify the

priority truthfully themselves. Another key difference is that, in a Web server context, the focus is on optimizing throughput and response time by applying queuing and control theory, and estimating expected service time. Our system does not use centralized queues and the performance levels and thereby service times are not estimated but explicitly requested by the users and enforced by virtualized resource configuration, which both simplifies and improves the accuracy of our implementation.

Yet another common type of application of admission control includes multimedia servers. Both the media itself (e.g. JPEG, MPEG) and the media consumers may have different tolerance of loss during playback, as well as different playback speed requirements. On the other end the disk I/O bandwidth limits the server capacity. Deterministically ensuring that all consumer frame-rate requests are met severely under-utilizes server resources under overload scenarios, which is why statistical approaches which exploit the loss tolerance and playback rate preference variations are popular [117, 133]. Our work differs from these I/O and storage centric algorithms, in that any type of resource share may be admission controlled, and the performance obtained is based on the budget spending rate, not solely on consumer preferences.

In general our focus on provider profit loss and price volatility as a means to control admission in a proportional share market makes our approach novel in this context.

Chapter 7

Conclusions

7.1 Concluding Remarks

We have investigated how statistical methods can be used in computational markets to predict demand. The methods proposed were designed for flexibility, scalability, and simplicity. Using only a few fundamental concepts of probability theory, such as statistical moments, probability density, Chebyshev's inequality and empirical prediction bounds we implemented a forecasting infrastructure that can easily be integrated with compute farm resource allocators to offer service guarantees ranging from best-effort to absolute (hard) guarantees. The main benefit of our solution is that it eases the burden on consumers in a computational market when making budget decisions. Instead of specifying that they are willing to spend a certain amount of currency on a computation they can much more intuitively express their valuation of the job in terms of the preferred guarantee of meeting a particular performance level, or meeting a deadline. A risk averse consumer can thereby spend more *insurance* money to complete a job on time. Moreover, by exposing the risk of performance degradation of resources, more efficient scheduling decisions can be made. A risk averse consumer might choose not to schedule jobs on the currently cheapest resource but on a resource that has the greatest likelihood to be the cheapest over the course of the execution of the job. Conversely we have also applied our prediction model to provide absolute guarantees by compensating providers for loss of profit as a result of rejecting consumers that would violate existing users' contracts.

The greatest challenge when predicting demand is that the statistical structure changes over time. Depending on which time window the forecast is based on very different results may be obtained. Our solution to this problem is twofold; first we provide standard time-based prediction windows, such as 5-min, hourly, daily, weekly, and monthly forecasts; second we calculate prediction bounds that indicate the uncertainty of the prediction model. For example if the prediction bounds are too wide for a 95 percentile guarantee of a service level using the daily forecasting

window, it tells the consumer that only hourly predictions may be made reliably on that resource due to a constantly changing structure of the dynamics of the demand. Other types of predictions may be more reliable the longer time window is used. Furthermore, reasoning about reliability of predictions is the only option in systems where demand fluctuates extensively. This observation mimics lessons learned from stock market predictions, where it is very hard to make point estimates of risk, but risky periods are known to cluster, and an indication of the current stability and predictability of the market might thus be the most valuable information to traders.

The second biggest challenge was to address the asymmetry of demand. Many popular prediction techniques known from, e.g. time series analysis and stock market asset pricing are based on the assumption of a completely random Brownian motion process where the increments of the demand between two subsequent periods are uncorrelated, and results in an overall demand distribution following the Gaussian density model. This assumption makes it easy to analyze dynamics mathematically, but risk estimates may be severely off when true demand follows a more erratic process, which has been observed in most real systems including both stock markets and computational markets. If the demand distribution is right skewed, or long tailed a normal distribution would under-estimate the risk of high demand. Predicting tail behavior is important as it gives users bounds of worst case scenarios. Our solution to this problem is to base our predictions on the theoretical Chebychev bound, which is independent of the asymmetry of the distribution, refined with the 3-σ bound where appropriate. We found in our analysis of real workload traces from high performance computing centers that this method resulted in more reliable forecasts than the Gaussian-based methods.

Finally, we have demonstrated in this thesis how resource providers can trade off spot market offerings with reservation market offerings. This trade-off can be compared to the trade-off airline reservation systems have to make when pricing passenger seats. Well ahead of time, airlines are uncertain about the actual demand on the departure date, and might thus want to trade off this uncertainty with a guaranteed profit and give an incentive to passengers to book early by means of discounts. Closer to departure, demand becomes more predictable as fewer seats are available and more passengers' travel plans are finalized, but on the other hand the demand might also change more rapidly. In this case the airline might want to put seats on the spot market to follow the demand as closely as possible to make sure that all the seats get sold, and to profit from last minute travelers who desperately need a seat and are willing to pay a very high price. Our solution to this problem is a set of simple functional expressions that can be evaluated both by consumers and providers to decide which market to enter to optimize efficiency and value based on utility functions representing the elasticity or flexibility of the resource requirements.

7.2 Future Directions

The main remaining challenge for computational markets is to make resource allocations as simple as possible without losing the fine-grained control of service-levels. Whether this usability goal is met can only be evaluated in pilot projects with real users in large markets with resource contention. A thorough analysis of the benefits of predictions and economic allocations in a live system with real users would expose the psychological factors of decision making which are hard to capture accurately in simulations and experiments with modeled user behavior. One approach we are investigating to make these kinds of systems more accessible for distributed applications is to support standard parallel programming environments such as Map/Reduce. This approach also has the benefit of providing higher-level monitoring metrics that can be used as input to the risk probing functionality in our model to make job budgeting self-managed.

Once more experience is gained with real users, the next challenge is to provide more advanced contractual models. Different types of financial derivatives, such as forwards, options, and futures have been suggested in the literature, but their deployment in real computational markets still remain a challenging research topic. Our admission control implementation can be used as the basis for offering these higher level contracts. Design of insurance agent models is a related research topic, where the idea is that agents can aggregate risk from a large pool of consumers and provide a higher level of risk mitigation than can be achieved individually by the budgets of individual consumers. In general the research on computational markets have only barely scraped the surface of economic mechanisms available in more mature financial markets. As consumption of computational resources become more commercialized, commoditized and globalized, as exemplified by the recent trend of cloud computing initiatives, such as Amazon's Elastic Compute Cloud, Hewlett-Packard's Flexible Compute Service, Google's GFS and MapReduce-based Vdoop, IBM's Blue Cloud, Yahoo's M45, Sun's Sun Grid and Microsoft's Dryad, more sophisticated economic models of resource allocation are needed.

Bibliography

[1] A. Anderson, A. Nadalin, B. Parducci, D. Engavatow, H. Lockhart, M. Kudo, P. Humenn, S. Godik, S. Abderson, S. Crocker, and T. Moses. eXtensible Access Control Markup Language (XACML) Version 1.0. Technical report, OASIS, 2003.

[2] Jorge Andrade and Jacob Odeberg. HapGrid: a Resource for Haplotype Reconstruction and Analysis using the Computational Grid Power in Nordugrid. *HGM2004: New Technologies in Haplotyping and Genotyping*, April 2004.

[3] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). Technical report, Global Grid Forum, 2005.

[4] Oscar Ardaiz, Pau Artgas, Torsten Eymann, Felix Freitag, Roc Messeguer, Leandro Navarro, and Michael Reinicke. Exploring the Catallactic Coordination Approach for Peer-to-Peer Systems. In *Proceedings of the 9th International Euro-Par Conference*, 2003.

[5] Alvin AuYoung, Laura Grit, Janet Wiener, and John Wilkes. Service Contracts and Aggregate Utility Functions. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC)*, June 2006.

[6] A. Barmouta and R. Buyya. GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration. In *Int. Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003. IEEE.

[7] F. Berman, G Fox, and A.J.G. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, 2003.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W Weiss. An Architecture for Differentiated Services. RFC 2475, IETF, December 1998.

[9] S. Blake, D. Black, M. Carlson, E. Davis, W. Zheng, and W. Weiss. RFC 2475: An Architecture for Differentiated Services. Technical report, IETF, 1998.

[10] Diana Bosio, James Casey, Akos Frohner, Leanne Guy, Peter Kunszt, Erwin Laure, Sophie Lemaitre, Levi Lucio, Heinz Stockinger, Kurt Stockinger, William Bell, David Cameron, Gavin McCance, Paul Millar, Joni Hahkala, Niklas Karlsson, Ville Nenonen, Mika Silander, Olle Mulmo, Gian-Luca Volpato, Giuseppe Andronico, Federico DiCarlo, Livio Salconi, Andrea Domenici, Ruben Carvajal-Schiaffino, and Floriano Zini. Next-Generation EU DataGrid Data Management Services. In *Proceedings of Computing in High Energy and Nuclear Physics*, La Jolla, CA, USA, March 2003.

[11] R. Braden, S. Clark, and S. Shenker. Integrated Services in the Internet Architecture. RFC 1633, IETF, June 1994.

[12] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. RFC 2205: ReSerVation Protocol (RSVP) Version 1 Functional Specification. Technical report, IETF, 1997.

[13] John Brevik, Daniel Nurmi, and Rich Wolski. Predicting Bounds on Queuing Delay for Batch-scheduled Parallel Machines. In *PPoPP '06: Proceedings of the 2006 ACM Principles and Practices of Parallel Programming*, New York, NY, USA, 2006. ACM.

[14] Rajkumar Buyya, David Abramson, and Jonathan Giddy. An Architecture of a Resource Management and Scheduling System in a Global Computational Grid. In *HPC Asia*, pages 283–289, 2000.

[15] Rajkumar Buyya, Manzur Murshed, David Abramson, and Srikumar Venugopal. Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. *Software: Practice and Experience (SPE) Journal*, 35(5):491–512, April 2005.

[16] Germano Caronni, Tim Curry, Pete St. Pierre, and Glenn Scott. Supernets and snHubs: A Foundation for Public Utility Computing. Technical Report TR-2004-129, Sun Microsystems, 2004.

[17] Anthony Chavez, Alexandros Moukas, and Pattie Maes. Challenger: a Multiagent System for Distributed Resource Allocation. In *AGENTS '97: Proceedings of the First International Conference on Autonomous Agents*, pages 323–331, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-877-0.

[18] Chunming Chen, Muthucumaru Maheswaran, and Michel Toulouse. Supporting Co-allocation in an Auctioning-based Resource Allocator for Grid Systems. In *Proceedings of the 16th IEEE International Parallel and Distributed Processing Symposium (IPDPS'02)*, 2002.

[19] Kay-Yut Chen, Lesli R. Fine, and Bernardo A. Huberman. Predicting the Future. *Information Systems Frontiers*, 5(1):47–61, 2003.

[20] Ming Chen, Guangwen Yang, and Xuezheng Liu. Gridmarket: A Practical, Efficient Market Balancing Resource for Grid and P2P Computing. *Lecture Notes in Computer Science*, 3033:612–619, 2004.

[21] Xiangping Chen, Prasant Mohapatra, and Huamin Chen. An Admission Control Scheme for Predictable Server Response Time for Web Accesses. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 545–554, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-348-0.

[22] Brent N. Chun, Philip Buonadonna, Alvin AuYoung, Chaki Ng, David C. Parkes, Jeffrey Shneidman, Alex C. Snoeren, and Amin Vahdat. Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, 2005.

[23] Brent N. Chun and David E. Culler. REXEC: A Decentralized, Secure Remote Execution Environment for Clusters. In *Communication, Architecture, and Applications for Network-Based Parallel Computing*, pages 1–14, 2000.

[24] Brent N. Chun and David E. Culler. User-centric Performance Analysis of Market-based Cluster Batch Schedulers. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, 2002.

[25] Li ChunLin and Li Layuan. A two Level Market Model for Resource Allocation Optimization in Computational Grid. In *CF '05: Proceedings of the 2nd conference on Computing frontiers*, pages 66–71, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-019-1.

[26] Scott Clearwater and Stephen D. Kleban. Heavy-tailed Distributions in Supercomputer Jobs. Technical Report SAND2002-2378C, Sandia National Labs, 2002.

[27] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[28] Karl Czajkowski, Ian Foster, Carl Kesselman, Volker Sander, and Steven Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. *Lecture Notes in Computer Science*, 2537:153–183, 2002.

[29] A. Dan, E. Davis, R. Kearney, A. Keller, R.P. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and Y.A. Web Services on Demand: WSLA-driven Automated Management. *IBM Systems Journal*, 43, 2004.

[30] Abnubhav Das and Daniel Grosu. Combinatorial Auction-Based Protocols for Resource Allocation in Grids. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

[31] T. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss. Overview of the I-WAY: Wide Area Visual Supercomputing. *International Journal of Supercomputer Applications*, 10:123–130, 1996.

[32] Persi Diaconis and Brad Efron. Computer-Intensive Methods in Statistics. *Scientific American*, (6):116–130, 1983.

[33] D. A. Dickey and W. A. Fuller. Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root. *Econometrica*, (49):1057–1072, 1981.

[34] Boris Dragovic, Keir Fraser, Steve Hand, Tim Harris, Alex Ho, Ian Pratt, Andrew Warfield, Paul Barham, and Rolf Neugebauer. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[35] Brad Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[36] Carl M. Ellison. The Utah TENEX Scheduler. *Proceedings of the IEEE*, 63 (6):940–945, 1975.

[37] Erik Elmroth, Peter Gardfjell, Olle Mulmo, and Thomas Sandholm. An OGSA-Based Bank Service for Grid Accounting Systems. In Jerzy Wasniewski, editor, *Lecture Notes in Computer Science: Applied Parallel Computing. State-of-the-art in Scientific Computing*. Springer Verlag, 2004.

[38] Sameh Elnikety, Erich Nahum, John Tracey, and Willy Zwaenepoel. A Method for Transparent Admission Control and Request Scheduling in e-Commerce Web Sites. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 276–286, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-844-X.

[39] Robert Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50:987–1007, 1982.

[40] Dror G. Feitelson. Workload Modeling for Performance Evaluation. *Lecture Notes in Computer Science*, (2459):114–141, 2002.

[41] Michal Feldman, Kevin Lai, and Li Zhang. A Price-Anticipating Resource Allocation Mechanism for Distributed Shared Clusters. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[42] William Feller. *An Introduction to Probability Theory and its Applications, volume II*. Wiley Eastern Limited, 1988.

[43] Donald Ferguson, Yechiam Yemimi, and Christos Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In *International Conference on Distributed Computer Systems*, pages 491–499, 1988.

[44] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In *Proceedings of the International Workshop on Quality of Service*, 1999.

[45] I. Foster, A. Roy, V. Sander, and L. Winkler. End-to-End Quality of Service for High-End Applications. Technical report, Argonne National Laboratory, 1999.

[46] Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *IFIP'05: Proceedings of International Conference on Network and Parallel Computing*, volume 3799, pages 2–13. LNCS, Springer-Verlag GmbH, 2005.

[47] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organization. *International Journal of Supercomputing Applications*, 15(3), 2001.

[48] Ian Foster and Carl Kessleman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[49] Ian Foster and Carl Kessleman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.

[50] David Freedman and Persi Diaconis. On the Histogram as a Density Estimator: L2 Theory. *Probability Theory and Related Fields*, 57(4):453–476, 1981.

[51] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. SHARP: An Architecture for Secure Resource Peering. In *ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[52] Peter Gardfjell, Erik Elmroth, Lennart Johnsson, Olle Mulmo, and Thomas Sandholm. Scalable Grid-wide Capacity Allocation with the SweGrid Accounting System (SGAS). *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[53] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A.H.G. Rinnoody Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, (5):287–326, 1979.

[54] Sven Graupner, Jim Pruyne, and Singhal Sherad. Making the Utility Data Center a Power Station for the Enterprise Grid. Technical Report HPL-2003-53, Hewlett-Packard Laboratories, 2003.

[55] Ian Hacking. Nineteenth Century Cracks in the Concept of Determinism. *Journal of the History of Ideas* , 44(3):455–475, September 1983.

[56] Gerald J. Hahn and William Q. Meeker. *Statistical Intervals: A Guide for Practitioners*. John Wiley & Sons, Inc, New York, NY, USA, 1991.

[57] Garrett Hardin. The Tragedy of the Commons. *Science*, 162:1243–1248, 1968.

[58] Joseph Hellerstein, Kaan Katricioglu, and Maheswaran Surendra. An Online, Business-Oriented Optimization of Performance and Availability for Utility Computing . Technical Report RC23325, IBM, December 2003.

[59] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459: Internet X.509 Public Key Infrastructure and CRL Profile. Technical report, IETF, 1999.

[60] Bernardo A. Huberman. *The Laws of the Web, Patterns in the Ecology of Information*. MIT Press, 2001.

[61] H.E. Hurst. Long Term Storage Capacity of Reservoirs. *Proc. American Society of Civil Engineers*, 76(11), 1950.

[62] David Irwin, Jeff Chase, and Laura Grit. Balancing Risk and Reward in Market-Based Task Scheduling. In *International Symposium on High Performance Distributed Computing*, 2004.

[63] Laxmikant V. Kale, Sameer Kumar, Mani Potnuru, Jayant DeSouza, and Sindhura Bandhakavi. Faucets: Efficient Resource Allocation on the Computational Grid. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2197-5.

[64] Katarzyna Keahey, Karl Doering, and Ian Foster. From Sandbox to Playground: Dynamic Virtual Environments in the Grid. In *Grid 2004: Proceedings of the 5th International Workshop in Grid Computing*, Pittsburgh, PA, USA, November 2004.

[65] Edward W. Knightly and Ness Shroff. Admission Control for Statistical QoS: Theory and Practice. *IEEE Network*, 13(2):20–29, March/April 1999.

[66] James F. Kurose and Rahul Simha. A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.

[67] Kevin Lai. Markets are Dead, Long Live Markets. *SIGecom Exchanges*, 5(4): 1–10, July 2005.

[68] Kevin Lai, Bernardo A. Huberman, and Leslie Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical report, arXiv, 2004.

[69] Kevin Lai, Lars Rasmusson, Eytan Adar, Stephen Sorkin, Li Zhang, and Bernardo A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. *Multiagent and Grid Systems*, 1(3):169–182, August 2005.

[70] Kevin Lai and Thomas Sandholm. The Design, Implementation, and Evaluation of a Market-Based Resource Allocation System. Technical Report Manuscript for Publication, Royal Institute of Technology and Hewlett-Packard Labs, Stockholm, Sweden, May 2006.

[71] Spyros Lalis and Alexandros Karipidis. JaWS: An Open Market-Based Framework for Distributed Computing over the Internet. *Lecture Notes in Computer Science*, 1971:87–106, 2000.

[72] D. Lamanna, J. Skene, and W. Emmerich. SLAng: A Language for Defining Service Level Agreements. In *Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS03)*, 2003.

[73] Sam C. M. Lee, John C. S. Lui, and David K. Y. Yau. Admission Control and Dynamic Adaptation for a Proportional-delay Diffserv-enabled Web Server. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 172–182, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-531-9.

[74] John D. C. Little. A Proof of the Queuing Formula: L=λW. *Operations Research*, 9(3):383–387, 1961.

[75] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, December 1998.

[76] M. Lorch and D. Skow. Authorization Glossary. Technical report, Global Grid Forum, 2004.

[77] Jeffrey K. MacKie-Mason, Anna Osepayshvili, Daniel M. Reeves, and Michael P. Wellman. Price Prediction Strategies for Market-Based Scheduling. In *ICAPS*, pages 244–252, 2004.

[78] Thomas W. Malone, Richard E. Fikes, Kenneth R. Grant, and Michael T. Howard. Enterprise: A Market-like Task Scheduler for Distributed Computing Environments. In Bernardo A. Huberman, editor, *The Ecology of Computation*, number 2 in Studies in Computer Science and Artificial Intelligence, pages 177–205. Elsevier Science Publishers B.V., 1988.

[79] Benoit Mandelbrot, Adlai Fisher, and Laurent Calvet. The Multifractal Model of Asset Returns. In *Cowles Foundation Discussion Papers: 1164*. Yale University, 1997.

[80] Benoit Mandelbrot and Richard L. Hudson. *The (Mis)behavior of Markets: A Fractal View of Risk, Ruin, and Reward*. Basic Books, New York, NY, USA, 2004.

[81] Mark S. Miller and K. Eric Drexler. Comparative Ecology: A Computational Perspective. In Bernardo A. Huberman, editor, *The Ecology of Computation*, number 2 in Studies in Computer Science and Artificial Intelligence, pages 51–76. Elsevier Science Publishers B.V., 1988.

[82] J. Susan Milton and Jesse C. Arnold. *Introduction to Probability Theory and Statistics*. McGraw Hill, 2003.

[83] K. Nahrstedt, H. Chu, and S. Narayan. QoS-aware Resource Management for Distributed Multimedia Applications. *Journal on High-Speed Networking* , December 1998.

[84] Chaki Ng, Philip Buonadonna, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. Addressing Strategic Behavior in a Deployed Microeconomic Resource Allocator. In *Proceedings of the 3rd Workshop on Economics of Peer-to-Peer Systems*, 2005.

[85] Norman R. Nielsen. The Allocation of Computer Resources–Is Pricing the Answer? *Communications of the ACM*, 13(8):467–474, 1970.

[86] David Oppenheimer, Brent Chun, David Patterson, Alex C. Snoeren, and Amin Vahdat. Service Placements in a Shared Wide-Area Platform. In *USENIX'06: Annual Technical USENIX Conference*, 2006.

[87] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, July 2002.

[88] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[89] Christos H. Papadimitriou. Algorithms, Games, and the Internet. In *Symposium on Theory of Computing*, 2001.

[90] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1984.

[91] David Pennock. A Dynamic Pari-Mutuel Market for Hedging, Wagering, and Information Aggregation. In *Proceedings of the fifth ACM Conference on Electronic Commerce (EC'04)*, 2004.

[92] Rosario M. Piro, Andrea Guarise, and Albert Werbrouck. An Economy-based Accounting Infrastructure for the DataGrid. In *Proceedings of the Fourth IEEE International Workshop on Grid Compuiting (GRID'03)*, 2003.

[93] Florentina I. Popovici and John Wilkes. Profitable services in an uncertain world. In *SC05: Proceedings of Supercomputing*, 2005.

[94] Ori Regev and Noam Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of 1st International Conference on Information and Computation Economies*, pages 148–157, 1998.

[95] Thomas Sandholm. The Philosophy of the Grid: Ontology Theory - From Aristotle to Self-Managed IT Resources. Technical Report TRITA-NA-0532, Royal Institute of Technology, Stockholm, Sweden, September 2005.

[96] Thomas Sandholm. Service Level Agreement Requirements of an Accounting-Driven Computational Grid. Technical Report TRITA-NA-0533, Royal Institute of Technology, Stockholm, Sweden, September 2005.

[97] Thomas Sandholm. Autoregressive Time Series Forecasting of Computational Demand. Technical Report 0711.2062v1 [cs.DC], arXiv, 2007.

[98] Thomas Sandholm. *Managing Service Levels in Grid Computing Systems*. Licentitate Thesis ISRN KTH/CSC/A–07/06–SE. Royal Institute of Technology, Stockholm, 2007.

[99] Thomas Sandholm, Peter Gardfjell, Erik Elmroth, Lennart Johnsson, and Olle Mulmo. An OGSA-based Accounting System for Allocation Enforcement across HPC Centers. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 279–288, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-871-7.

[100] Thomas Sandholm, Peter Gardfjell, Erik Elmroth, Lennart Johnsson, and Olle Mulmo. A Service-Oriented Approach to Enforce Grid Resource Allocations. *International Journal of Cooperative Information Systems*, 2006.

[101] Thomas Sandholm and Kevin Lai. Evaluating Demand Prediction Techniques for Computational Markets. In *GECON '06: Proceedings of the 3rd International Workshop on Grid Economics and Business Models*, May 2006.

[102] Thomas Sandholm and Kevin Lai. Prediction-Based Enforcement of Performance Contracts. In *GECON '07: Proceedings of the 4th International Workshop on Grid Economics and Business Models*, 2007.

[103] Thomas Sandholm and Kevin Lai. A Statistical Approach to Risk Mitigation in Computational Markets. In *HPDC '07: Proceedings of the 16th ACM International Symposium on High Performance Distributed Computing*, 2007.

[104] Thomas Sandholm, Kevin Lai, Jorge Andrade, and Jacob Odeberg. Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications. In *HPDC '06: Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, June 2006.

[105] Thomas Sandholm, Kevin Lai, and Scott Clearwater. Admission Control in a Computational Market. In *CCGrid '08: Proceedings of the 8th International Symposium on Cluster Computing and the Grid*, 2008. To appear.

[106] Dave Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Addison Wesley, 1992.

[107] Ludwig Seitz, Erik Rissanen, Thomas Sandholm, Babak Sadighi Firozabadi, and Olle Mulmo. Policy Administration Control and Delegation using XACML and Delegent. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, November 2005.

[108] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications*. Springer, 2000.

[109] Frank Siebenlist, Takuya Mori, Rachana Ananthakrishnan, Liang Fang, Tim Freeman, Kate Keahey, Sam Meder, Olle Mulmo, and Thomas Sandholm. The Globus Authorization Processing Framework. In *Workshop on New Challenges for Access Control*, Ottawa, Canada, April 2005.

[110] O. Smirnova, P. Erola, T. Ekelöf, M. Ellert, J.R. Hansen, A. Konsantinov, B. Konya, J.L. Nielsen, F. Ould-Saada, and A. Wäänänen. The NorduGrid Architecture and Middleware for Scientific Applications. *Lecture Notes in Computer Science*, 267:264–273, 2003.

[111] Ion Stoica, Hussein Abdel-Wahab, and Alex Pothen. A Microeconomic Scheduler for Parallel Computers. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–135, April 1995.

[112] Michael Stonebraker, Paul M. Aoki, Witold Litwin, Avi Pfeffer, Adam Sah, Jeff Sidell, Carl Staelin, and Andrew Yu. Mariposa: a wide-area distributed database system. *The VLDB Journal*, 5(1):048–063, 1996. ISSN 1066-8888.

[113] I.E. Sutherland. A Futures Market in Computer Time. *Communications of the ACM*, 11(6):449–451, 1968.

[114] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. IETF RFC 3820. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, 2004.

[115] Steven Tuecke, Karl Czajkowski, Ian Foster, Jeff Frey, Steven Graham, Carl Kesselman, Tom Maquire, Thomas Sandholm, David Snelling, and Peter Vanderbilt. Open Grid Services Infrastructure (OGSI) Version 1.0. Technical report, Global Grid Forum, 2003.

[116] Hal R. Varian. Equity, Envy, and Efficiency. *Journal of Economic Theory*, 9: 63–91, 1974.

[117] H. Vin, P. Goyal, and A. Goyal. A Statistical Admission Control Algorithm for Multimedia Servers. In *MULTIMEDIA '94: Proceedings of the second ACM international conference on Multimedia*, pages 33–40, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-686-7.

[118] D. F. Vysochanskij and Y. I. Petunin. Justification of the 3 sigma Rule for Unimodal Distributions. *Theory of Probability and Mathematical Statistics*, 21:25–36, 1980.

[119] C. A. Waldspurger. Lottery and Stride Scheduling: Flexible Proportional-share Resource Management. Technical Report MIT/LCS/TR-667, 1995.

[120] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A Distributed Computational Economy. *Software Engineering*, 18(2):103–117, 1992.

[121] Zheng Wang. A Case for Proportional Fair Sharing. In *IWQoS '98: Proceedings of the Sixth International Workshop on Quality of Service*, pages 33–35. IEEE, 1998. ISBN 0-7803-4482-0.

[122] William W. S. Wei. *Time Series Analysis, Univariate and Multivariate Methods*. Pearson Addison Wesley, 2006.

[123] Von Welch, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Samuel Meder, and Frank Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*, 2004.

[124] Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, and Yevgeniy Vorobeychik. Price Prediction in a Trading Agent Competition. *J. Artif. Intell. Res. (JAIR)*, 21:19–36, 2004.

[125] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jeffery K. MacKie-Mason. Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[126] W.H. Williams and M.L. Goodman. A Simple Method for the Construction of Empirical Confidence Limits for Economic Forecasts. *Journal of the American Statistical Association*, 66(336):752–754, 1971.

[127] Rich Wolski, Graziano Obertelli, Matthew Allen, Daniel Nurmi, and John Brevik. Predicting Grid Resource Performance On-Line. In *Handbook of Innovative Computing: Models, Enabling Technologies, and Applications*. Springer Verlag, 2005.

[128] Rich Wolski, James S. Plank, Todd Bryan, and John Brevik. G-commerce: Market Formulations Controlling Resource Allocation on the Computational

Grid. In *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, page 10046.2, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-0990-8.

[129] Ximing Wu and Thanasis Stengos. Partially Adaptive Estimation via the Maximum Entropy Densities. *Econometrics Journal*, 8(3):352–366, 2005.

[130] Lijuan Xiao, Yanmin Zhu, Lionel M. Ni, and Zhiwei Xu. GridIS: An Incentive-Based Grid Scheduling. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 65.2, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2312-9.

[131] Chee Shin Yeo and Rajkumar Buyya. A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing.

[132] Li Zhang. The Efficiency and Fairness of a Fixed Budget Resource Allocation Game. *Lecture Notes in Computer Science*, 3580:485–496, 2005. ISSN 0302-9743.

[133] Roger Zimmermann and Kun Fu. Comprehensive Statistical Admission Control for Streaming Media Servers. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 75–85, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-722-2.

Part II
Main Contribution Papers

Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications

Thomas Sandholm
KTH – Royal Institute of Technology
Center for Parallel Computers
SE-100 44 Stockholm, Sweden
sandholm@pdc.kth.se

Kevin Lai
Hewlett-Packard Laboratories
Information Dynamics Laboratory
Palo Alto, CA 94304, USA
kevin.lai@hp.com

Jorge Andrade Ortíz and Jacob Odeberg*
KTH – Royal Institute of Technology
Dept. of Biotechnology
SE-100 44 Stockholm, Sweden
{jacob,andrade}@biotech.kth.se

Abstract

We present the implementation and analysis of a market-based resource allocation system for computational Grids. Although Grids provide a way to share resources and take advantage of statistical multiplexing, a variety of challenges remain. One is the economically efficient allocation of resources to users from disparate organizations who have their own and sometimes conflicting requirements for both the quantity and quality of services. Another is secure and scalable authorization despite rapidly changing allocations.

Our solution to both of these challenges is to use a market-based resource allocation system. This system allows users to express diverse quantity- and quality-of-service requirements, yet prevents them from denying service to other users. It does this by providing tools to the user to predict and tradeoff risk and expected return in the computational market. In addition, the system enables secure and scalable authorization by using signed money-transfer tokens instead of identity-based authorization. This removes the overhead of maintaining and updating access control lists, while restricting usage based on the amount of money transferred. We examine the performance of the system by running a bioinformatics application on a fully operational implementation of an integrated Grid market.

1. Introduction

The combination of decreasing cost for network bandwidth and CPU performance and the availability of open-source distributed computing middleware has led the high-performance computing community away from monolithic supercomputers to low-cost distributed cluster solutions. This model of computing allows users with bursty and computationally demanding tasks to share and use compute resources on demand. This usage model, aka utility computing [24], Grid computing [22], peer-to-peer compute farming [28], or Global Computing [23], has been applied to solve diverse technical computing problems in fields such as, bioinformatics [9], high-energy physics [10], graphics rendering [3], and economic simulation [2].

One common question remains: how to manage the allocation of resources to users? One challenge is that users are from disparate organizations and have their own and sometimes conflicting requirements for both the quantity and quality of services. For example, academic Grid projects [37, 4, 38, 8, 7, 6, 5, 10] typically require resources for throughput sensitive, long-running batch applications, while industrial utility computing offerings [14, 24, 26] typically require response-time sensitive, interactive, and continuous application server provisioning. One common solution is to have separate resource allocation mechanisms for different applications. However, this merely shifts the problem from reconciling the resource requirements of different applications to reconciling the resource requirements of different mechanisms.

Another challenge is that users have a web of relation-

*also affiliated with Karolinska Hospital, Gustav V Research Institute, Dept. of Medicine, Atherosclerosis Research Unit, Stockholm, Sweden

ships with regard to how they wish to share resources with each other. For example, one site may wish to share resources with a remote site, but only when demand from local users is low. Another example is that a site may wish to be *reciprocative*, where it only shares resources with sites that share resources with it. One common solution is to use access control lists (ACLs) to authorize access to resources. The problem is that managing ACLs is difficult because many users could potentially access a site, a site has many different types of resources, each of which may need a separate ACL, and the degree of access that each user has could change with each access. For example, as a user from site A uses host X at site B, site B would want to decrease the ability of other site A users from being able to consume resources at host X and possibly other hosts at the site. The dynamic updating of this amount of data not only increases overhead and development time, but could lead to inconsistencies that allow exploitation of the system.

Instead, we examine a market-based approach to resource allocation in Grids. A number of models and mechanisms for electronic markets, and computational economies have been proposed [40, 30, 36, 35, 33, 12, 27]. In previous work we have presented and analyzed Tycoon [31], a market-based resource allocation system for shared clusters. Here we focus on how market-based techniques address issues in a Grid environment. More specifically, our contributions are as follows:

• **A Tycoon controlled cluster scheduler for Grid execution managers.** The NorduGrid/ARC *meta-scheduler* [38] used by many academic Grid projects in Europe, such as SweGrid [37], to schedule large-scale scientific jobs across a collection of heterogeneous HPC sites using a uniform job submission and monitoring interface, was integrated with Tycoon. The integration allows the large existing user base of academic Grids to bid for and use resources controlled by economic markets seamlessly. We are also working on integrating our scheduler with Globus Toolkit 4 [21].

• **Pilot application experiments.** We use a bioinformatics pilot application currently deployed in an academic Grid to verify our model and implementation. A bioinformatics application scanning and analyzing the complete human proteome using a blast-based similarity search was run on a Tycoon cluster and evaluated.

• **Price prediction models and mechanisms.** We discuss, implement and analyze models and mechanisms to predict cost of resources and give guarantees of QoS (Quality of Service) levels based on job funding. In a spot market as implemented by Tycoon it can be hard to predict the future price of a resource and know how much money should be spent on funding a job with a specific set of requirements. To that end we provide a suite of lightweight prediction capabilities that can give users guidance regarding what performance levels to expect for different levels of funding of a job.

• **A security model combining identity and capability-based access control.** In academic Grid networks it is important to identify all users securely because a user's identity, and membership in virtual organizations, can automatically give access to shared resources. In electronic markets, however, the key question is whether you can present proof of payment for a resource. Our model allows Grid users to make use of electronic money transfer tokens, or checks, that can be used to pay for and gain access to resources to be used by a compute job, as well as to specify its priority and thereby 'buy' a certain level of QoS.

As a result, we believe that the combination of Grid and market mechanisms is a promising and viable approach to sharing resources in an economically sustainable way while ensuring fairness and overall system efficiency.

The rest of the paper is organized as follows. In Section 2, we provide an overview of the design and discuss how it meets our goal. In Section 3, we delve more deeply into implementation specifics. Our price prediction analysis is presented in Section 4. Section 5 contains experimental results from running the bioinformatics application on our integration testbed. We describe related work in Section 6. We conclude by discussing some limits of our model and future work in Section 7.

2. Architecture Overview

2.1. A Case for Grid Markets

High-end compute resources, such as Grid-enabled High Performance Computing (HPC) clusters, are necessary for many scientific computing applications. These applications can use more resources to scan larger input data sets, provide a higher resolution for simulations, or simply complete the same amount of work faster. In other words, they exhibit a continuous utility curve where a larger resource share results in a higher utility.

Traditional queueing and batch scheduling algorithms assume that job priorities can simply be set by administrative means. In large-scale, cross-organizational, and potentially untrusted Grids, this can be expensive and slow, and allocations may not reflect the true relative priorities of jobs. Determining the relative priorities of applications requires a truth revelation mechanism like a market [12, 27, 41, 39]. This provides an incentive for uses to accurately prioritize their applications.

2.2. Tycoon

In this paper, we apply a previously described market-based resource allocation system (Tycoon) [29] in the Grid

context. In this section, we briefly review this system before describing the security and prediction extensions that are the focus of this paper. Please see the previous publication for additional details.

The main characteristics of Tycoon are its decentralized and continuous markets. Each host that contributes resources to a Tycoon network runs its own market. This reduces the dependency on centralized components, making the system more scalable, fault-tolerant, and agile in the allocation of resources. The continuous market allows users to bid and receive resources at a high frequency (10 seconds by default), which allows applications to start quickly and react to quickly changing load. This is more important for service-oriented applications like web servers and databases than the typical Grid applications that runs for days. Sharing the same infrastructure across these different types of applications allows better statistical multiplexing.

Another characteristic of Tycoon is the use of virtualized resources. This allows multiple applications to share the same host, which is useful both for applications that do not require a whole host and applications that have highly variable demand. Tycoon currently uses the Xen paravirtualization system [19], which imposes an overhead of 1%-5% for typical workloads.

The architecture of Tycoon is composed of the Bank, which maintains information on users like their credit balance and public keys, the Service Location Service, which maintains information on available resources, and Auctioneers, which run on each host and manage the market used to allocate resources on that host.

One of the main issues in this kind of distributed market is the efficiency relative to an ideal centralized market. Feldman, et al. [20] show that this market achieves both fairness and economic efficiency in the equilibrium when users use the Best Response optimization algorithm. Briefly, this algorithm solves the following optimization problem for a user:

$$\text{maximize } U_i = \sum_{j=1}^{n} w_{i,j} \frac{x_{ij}}{x_{ij}+y_j} \text{ subject to} \quad (1)$$

$$\sum_{j=1}^{n} x_{ij} = X_i, \text{ and } x_{ij} \geq 0. \quad (2)$$

where U_i is the utility of user i across a set of resources, $w_{i,j}$ is the preference of machine j as perceived by user i, for example the CPU capacity of the machine, $x_{i,j}$ is the bid user i puts on host j, y_j the total of all bids or the price of host j, and finally X_i is the total budget of user i.

One contribution in this work is to expose this Best Response algorithm to Grid HPC users, to allow them to easily use compute resources in a competitive market. Next we describe the architecture of this integration.

2.3. Grid Market Architecture

Our solution is novel in the sense that we maintain the overall Public Key Infrastructure (PKI) security model of the Grid, but introduce supply and demand driven dynamic pricing and resource share negotiation on a spot market within a virtual Grid cluster. Differentiated services are offered in an incentive compatible manner where the Grid user can attach a check-like token to jobs to pay for the resources to be consumed. The user needs to be fully authenticated using a Grid PKI handshake, but the authorization step is based on a capability composed of the funds transferred to the scheduling agent. The scheduling agent uses the Best Response algorithm described in the previous section to distribute and place bids on compute nodes. Virtual machines are created dynamically, with the appropriate QoS levels automatically configured in proportion to the bids placed. Job stage-in, execution, monitoring, performance boosting (by adding funds) and stage-out are all handled by the agent. Figure 1 depicts the overall architecture of the system. The Tycoon infrastructure is integrated with the Grid Job manager as a local scheduling system, fully transparent to the end-users. The next section describes the design and implementation in more detail.

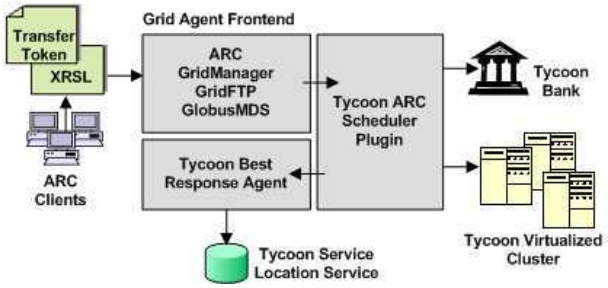


Figure 1. Tycoon Grid Market Architecture.

3. Grid Market Implementation

This section describes our integration of Tycoon with the Grid. Our approach is to integrate Tycoon with the infrastructure currently deployed in the SweGrid project, a national Grid in Sweden spanning six sites and a total of 600 compute nodes interconnected by the 10GB/s GigaSunet network. SweGrid has been in operation for a year and a half. The sites run the Nordugrid ARC middleware, which is based on the Globus toolkit, and currently connects about 50 HPC sites in 12 countries. Both ARC and Globus focus on providing an abstracted local cluster scheduler and execution manager so that the job submission mechanism and interface is the same regardless of whether PBS, LSF, Sun Grid Engine, etc., is used locally at the site.

As a result, to enable Tycoon for Grid users seamlessly, our approach was to write a Tycoon scheduler plugin for ARC to simplify the transition of SweGrid users to this new infrastructure. Integration directly as a Globus (GT4) plugin is work in progress. The Tycoon cluster is exposed as any other ARC cluster both for monitoring and job submission purposes, with the only difference being that the cluster is virtualized and thus reports number of virtual CPUs as opposed to physical compute node CPUs.

In ARC, the compute job requirements are describe in an XRSL (extended Globus Resource Specification Language) file. The CPU or WallTime XRSL attribute is mapped to the Tycoon resource bid deadline. The transfer token, described in more detail in the next section, is mapped to the total budget to be used within this deadline. Finally, the count attribute describes how many concurrent virtual machines or (virtual CPUs) the job requires. Mechanisms are provided to allow a job running in a virtual machine to access its ordinal number within such a batch of sub-jobs to, for example, process a different input data set. The bid distribution is determined by the Best Response algorithm of Tycoon. It calculates the optimal bids to maximize the overall utility gained from running a job on a set of host.

The ARC runtime environment allows users to specify what software needs to be installed in order to run the job. This software is installed automatically in the virtual machine using Yellow dog Updater, Modified (yum)[1]. The Tycoon ARC plugin also handles job stage-in, stage-out and output checkpoint monitoring. Jobs that have been submitted may be boosted with additional funding to complete sooner, and if the money deposited into host accounts has not been used (Tycoon only charges for resources actually used not bid for) the outstanding balance will be refunded to the user. Dynamic pricing, accounting and billing thus all happen automatically by means of the Tycoon infrastructure. Furthermore, the QoS specified in the XRSL description is automatically enforced by configuring custom virtual machines on demand when jobs require them. To limit the overhead of virtual machine creation, a user may reuse the same virtual machine between jobs submitted on the same physical host. However, no application data or scratch space is shared by different jobs.

Figure 2 shows a screenshot of the ARC Grid Monitor monitoring the Tycoon cluster. Note that since the number of CPUs are the number of virtual machines currently created, it can increase dynamically up to a maximum of about 15 times the number of physical nodes. Thus, for the current deployment of 40 hosts, a maximum of 600 virtual single CPU compute nodes can be offered concurrently to users by the Tycoon Grid. We could easily implement a virtual machine purging or hibernation model that could increase this number further if not all 600 machines need to be used at the same time, with the penalty of more overhead to setup a job on a virtual machine.

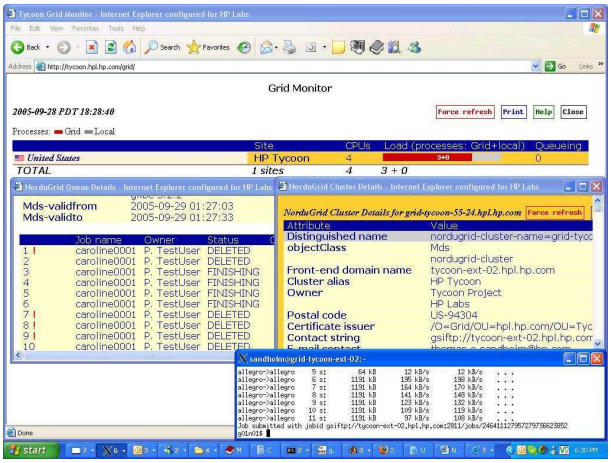


Figure 2. Screenshot of Nordugrid/ARC Tycoon monitor.

In addition to the virtualization of compute nodes, the cluster is also virtualized, in the sense that the Tycoon architecture is flat without hierarchies. This allows the submission agent to pick up compute nodes from any available physical cluster. Most of the current machines are at HP Labs in Palo Alto, California, but others are owned by Intel Research in Oregon, by Singapore, and by the Swedish Institute of Computer Science in Stockholm. The current limitation of 40 physical machines is only an artifact of the current state of the virtual cluster, and can grow dynamically as more clusters and nodes are added to the Tycoon network. Regardless of whether the compute node is local to the submission agent the host funding, job stage-in, job execution, job monitoring and job-stage out will all be done the same way. Finally, the agent itself can be replicated and partitioned to pick up a different set of compute nodes. The ARC meta-scheduler could then be used to load balance and do job to cluster matchmaking between the replicas. We therefore believe that this model will scale well as the number of compute nodes and virtual machines on these compute nodes increase.

3.1. Security Design

Our goal is to make the security integration as seamless as possible to the end-users, which means allowing Grid users to reuse their credentials when accessing a Tycoon cluster. To access a Tycoon cluster one needs a Tycoon bank account. So our task became equivalent to mapping a Grid certificate to a Tycoon bank account. Our approach is to transfer money from the user to the resource broker and map the proof of the transfer to a Grid identity, the Distinguished Name (DN) of the user.

This approach requires no manual access control list setup and it allows the user to keep both the Grid identity private key and Tycoon bank account private key on the local machine only. Furthermore, it does not require any changes to the existing Tycoon services. The user transfers money to the resource broker's bank account and then signs the receipt together with a Grid DN. The mapping can thus be decided independent of the transfer and can therefore also be used by arbitrary Grid users who do not have any Tycoon infrastructure deployed. On the resource side it is verified that the money transfer was indeed made into the broker account and that the transfer token has not been used before. The signature of the DN mapping is also verified to make sure that no middleman has added a fake mapping. Once the transfer token has been verified a new sub-account to the broker account is created and the money verified is transferred into this account by the broker. The new account is then used to fund and create host accounts used to run the job on behalf of the Grid user.

4. Price and Performance Prediction

4.1. Motivation

A challenge in any market-based resource allocation system is providing predictable performance. A variety of solutions exist, including reservations, future markets [11], options, and SLA contracts [42, 18]. However, all of these mechanisms require the ability to predict future demand and supply. Prediction is particularly important in spot-market systems like Tycoon that lack other mechanisms to ensure predictability. The failure to predict accurately either causes users to overspend on resources or prevents them from achieving their required quality of service.

Demand prediction requires a history of resource usage and a set of analysis algorithms. Our goal is to provide both a concise representation of historical prices on the Auctioneer and efficient client-side algorithms to analyze this data.

We represent the demand using the spot-market price on each host. In Tycoon, this reflects both the usage of and demand for a resource. This information is updated every 10 seconds, which is also the reallocation interval. In addition to the instantaneous demand, we also track the average, variation, distribution symmetry, and peak behavior of the price.

To make this information useful to a wide array of applications, periodization of the data is necessary. We implement this by presenting and scoping the statistics in moving, customizable time windows. By looking at smaller time windows we may be able to make simplifying assumptions such as the assumption of a symmetric normal distribution even though such a distribution may not be a good representation of a larger window, and vice versa. To track the price distribution dynamically we implement a self-adjusting slot table recording the proportion of prices that fall into certain ranges.

We now present three high-level prediction techniques to model our price data, (1) normal distribution prediction, chosen because it is simple to implement and rest only on fundamental statistical theory; (2) autoregression prediction of time series, a very common system identification approach; and (3) portfolio selection, a well-studied technique in economic theory to reduce risk. Finally, we present the theory behind our moving window approximation and smoothing implementation.

4.2. Lightweight Single-Host Stateless Price Prediction

In this model we assume a normal probability distribution of the spot market price and calculate the budgets needed to get a certain performance level or higher with a probability guarantee, which could be translated into the probability of a job completing within a certain deadline. Based on this information we would like to recommend a user to spend a certain amount of money given a capacity requirement and a deadline.

More formally, we first assume that the price y is an outcome of the normal random variable Y

$$Y \in N(\mu, \sigma^2), y \in Y \tag{3}$$

p is the probability given by the standard cumulative distribution function Φ, with μ as the measured mean of y and σ^2 as the measured variance. In other words, p is the probability that a resource offers a price less than or equal to y given its variance and mean.

$$p = P(Y \leq \frac{y-\mu}{\sigma}) = \Phi(\frac{y-\mu}{\sigma}) \tag{4}$$

The inverse cumulative distribution function, aka the probit quantile function of the normal distribution gives us the price y to expect with a given probability p.

$$\frac{y-\mu}{\sigma} \leq \Phi^{-1}(p) \Leftrightarrow y \leq \mu + \sigma\Phi^{-1}(p) \tag{5}$$

Combining (1) and (5) gives us the probability p to get the utility U given the budget X.

$$U_i(X_i, p) \geq \sum_{j=1}^{n} w_{i,j} \frac{x_{i,j}}{x_{i,j} + \mu_j + \sigma_j\Phi^{-1}(p)} \tag{6}$$

where $x_{i,j}$ is the bid picked by the best response algorithm in (1) with budget X_i on host j for user i.

If a user knows that the deadline d can be met if a utility greater than U is obtained, we can use (6) to recommend what budget to spend to meet that deadline, and conversely

what completion time to expect given a budget. For example, the budget X required to meet the deadline d with a certainty of p can be used as a recommendation for the extra cushion of funding needed to meet the deadline with a greater probability.

We call this model stateless, since we only need to keep track of running sums to report the mean and standard deviation of the price, and no data points need to be stored.

4.3. Single-Host Price Prediction Analyzing Time Series History Data

An autoregressive, $AR(k)$, [32] model based on a time series of CPU price snapshots was implemented using the following steps:

First, the unbiased autocorrelation with N sample snapshots of x and lag k is calculated as:

$$R(k) = \frac{1}{N-|k|} \sum_{n=0}^{N-|k|-1} x_{n+|k|} x_n$$

Then the following Yule-Walker linear equation system is solved using the Levinson reformulation:

$$L\alpha = r$$

where

$$L_{i,j} = R(i-j)$$

is the Toeplitz matrix with k rows and k columns, α is the column vector of k AR coefficients to be solved, and r is a column vector of size k where

$$r_i = R(i+1)$$

Now, future values of the time series x_i can be predicted using the coefficients in α as:

$$x_{N+1} = \mu + \sum_{j=0}^{k} \alpha_j (x_{i-j} - \mu)$$

where

$$\mu = \frac{1}{N} \sum_{n=0}^{N} x_n$$

Note that we omit the zero mean normal random white noise parameter here for simplicity.

4.4. Risk Management based Performance Prediction across Multiple Hosts

We now look at another prediction model for obtaining guidance in funding resources, portfolio theory. We need to obtain the return and plot that against the risk to calculate the efficient frontier where portfolios yield the most efficient trade-off between the two parameters. The fundamental rule of the frontier is that at a given risk value the return should be maximized and conversely at a given return value the risk should be minimized. We can then apply Morkowitz's mean-variance optimization [34]. As return we select the performance of the resource calculated as number of CPU cycles per second that are delivered per amount of money paid per second (inverse of spot market price).

Given the vectors of return and risk values for the resources, we used the matrix equations from [25] to calculate the risk free portfolio as well as the efficient frontier.

By looking at the efficient frontier we can, based on our degree of risk aversion, select a portfolio with an appropriate return. The advantage of the portfolio model is that we do not have to assume a normal probability distribution of the resource price. However, a symmetric distribution around the mean is assumed and it is also assumed that there is a variance in risk between resources that can be traded off with varying mean returns.

A similar approach focusing on Value-at-Risk analysis is presented in [16]. Their approach inherits the same strength and weaknesses as the general portfolio theory presented here, but extends it to give guarantees like, *within a given time horizon, the minimal performance will be a value V with a probability P* . In contrast, the approach presented here gives guidelines of the form, *given a certain level of risk aversion and expected performance, how should you distribute your budget across a set of hosts?*

4.5. Moving Window Smoothing Theory

We first look at the technique used to calculate moving windows for the price average (mean), variation (standard deviation), asymmetry of distribution (skewness), and peak behavior (kurtosis). A high value of skewness reflects a heavy-tailed (right-skewed) distribution, and a high value of kurtosis indicates that a large portion of the standard deviation is due to a few very high price peaks.

In terms of state information we only need to keep track of the previously calculated sample moments about the mean for the first (mean), second (standard deviation), third (skewness) and forth (kurtosis) moment about the mean. The linear smoothing function is determined by the window size, where a large window size results in the previously calculated moment having a very low impact on the next moment compared to the current snapshot, and vice versa. For window size 1, the previously calculated moments are ignored as expected.

$\mu_{i,p}$ is the pth sample moment about the mean at snapshot i, x_i is the price at snapshot i, n is the number of price samples in a window, σ_i is the standard deviation of price at snapshot i (for window n), $\gamma_{1,i}$ is the price skewness at

snapshot i (for window n), and $\gamma_{2,i}$ is the price kurtosis at snapshot i (for window n)

$$\mu_{0,p} = x_0^p$$

$$\mu_{i,p} = \alpha\mu_{i-1,p} + (1-\alpha)x_i^p$$

$$\alpha = 1 - \frac{1}{n}$$

$$\sum_{j=1}^{n} x_j^p = \mu_{i,p} n \Rightarrow \sigma_i = \sqrt{\mu_{i,2} - \mu_{i,1}^2}$$

equivalently,

$$\gamma_{1,i} = \frac{(\mu_{i,3} - 3\mu_{i,1}\mu_{i,2} + 2\mu_{i,1}^3)}{\sigma_i^3}$$

and,

$$\gamma_{2,i} = \frac{(\mu_{i,4} - 4\mu_{i,3}\mu_{i,1} + 6\mu_{i,2}\mu_{i,1}^2 - 3\mu_{i,1}^4)}{\sigma_i^4} - 3$$

We now look at the price distribution smoothing for moving time windows. The approach taken is to keep track of two price distributions for each window at all times. The distributions will contain twice as many snapshots as is required by the windows and have a time lag of the same size as the window. The merged window distribution to be retrieved at an arbitrary monitoring time is then calculated by using a share of both distributions proportional to how closely they are to the desired window size in terms of number of snapshots collected.

n is the total number of prices in a window, i is the snapshot time, $n_{k,i}$ is the number of prices in distribution array k at time i, $(0..2n)$, $s_{k,j}$ is the proportion of prices in slot j in distribution array k, $r_{i,j}$ is the proportion of prices to report in slot j at snapshot time i, and $w_{i,k}$ is the proportion of distribution array k to use in r at snapshot time i

$$w_{i,k} = 1 - \frac{|n_{1,i} - n|}{n}$$

$$|n_{1,\mathrm{i}} - n_{2,\mathrm{i}}| = n$$

$$\sum_{j=1}^{n_k} s_{k,j} = 1 \Rightarrow r_{i,j} = w_{1,k}\mathrm{s}_{1,\mathrm{j}} + (1 - w_{1,k})\mathrm{s}_{2,\mathrm{j}}$$

5. Grid Application Results

In this section we present some experimental results using a Bioinformatics application targeted for Grid environments, which was developed at the Bioinformatics laboratory at the Royal Institute of Technology in Stockholm [9]. It is a trivially parallelizable bag-of-task application, which is very typical for large-scale Grids. The experiments we present here, do not consider applications with more complicated workflow-like interactions among subtasks. However, none of the experiments depend in any way on the application-specific node processing performed by this application, more than the fact that it is CPU intensive.

5.1. Bioinformatics Application

The goal of the application is to identify protein regions with high or low similarity to the rest of the human proteome. A database of the complete human proteome is analyzed with a blast sequence alignment search tool performing stepwise similarity searches using a sliding window algorithm running in parallel on a distributed compute cluster. The reason for running this application in a compute farm is twofold, the proteome database is continuously evolving and the search is computationally hard. A search on a single machine takes about 8 weeks on a single node, and a run in the SweGrid compute farm utilizing 300 nodes out of 600 takes about 22 hours.

5.2. Experiment Setup

The proteome database is partitioned into chunks that can be analyzed in parallel. One of these chunks takes approximately 212 minutes to analyze on a single node in our cluster with a 100% share of a CPU. With 30 physical machines we can thus achieve a maximum performance of 35 hours/application run to be compared with 22 hours/run in SweGrid with 600 machines. In our experiment we are letting five competing users run the same application with different funding. The application makes use of a maximum of 15 nodes out of a total of 30 physical nodes. To have the users compete against each other but not between their own sub-jobs we restrict the setup to one virtual machine per user per physical machine. Hence, a maximum of 75 virtual machines may be used at any point in time. It should be noted that the physical machines have dual processors and there may thus not be competition for a CPU on a machine even though there are multiple users running there concurrently. The user jobs are launched in sequence with a slight delay to allow the best response selection to take the previous job funding into account. This is why users 1 and 2 tend to get to run on more nodes than the other users, as the price has not gone up as much at that point. Their shares will, however, be recomputed automatically and continuously within every 10s allocation interval.

5.3. Best Response Experiments

In this set of experiments we are interested in finding out whether an economically driven resource allocation mechanism would allow us to offer differentiated QoS levels to

Table 1. Equal Distribution of Funds

Users	Time(h)	Cost($/h)	Latency(min/job)	Nodes
$1-2$	7.16	4.19	28.66	15
$3-5$	6.36	4.28	45.49	8.7

Table 2. Two-Point Distribution of Funds

Users	Time(h)	Cost($/h)	Latency(min/job)	Nodes
$1-2$	7.07	5.10	29.31	14.5
$3-5$	4.16	10.9	23.46	11

Grid application users. We measure the **Time** defined as the wall-clock time as perceived by the user to complete the task of sub-jobs, the **Cost** as the money spent during this time, the **Latency** as the number of minutes it takes for each sub-job to complete (again in wall-clock time), and the number of **Nodes** or parallel sub-jobs used by the task. We start by gauging the environment and running the test with all users having the same funding for their jobs. They should hence expect an equal share of the CPUs. We, however, note from the results summarized in Table 1 that users 3-5 received a much lower quality of service, here defined as number of jobs that can be processed within a time unit, because the best response algorithm found it too expensive to fund more than a very low number of hosts. One possible solution to this issue would be to let the user hold back on submitting if not a threshold of minimum hosts to bid on is met.

The results from a two-point distribution with users funding their jobs with $100, 100, 500, 500, 500$ dollars with a deadline of 5.5 hours is summarized in Table 2.

Here we can see that the jobs with a budget of 500 dollars caused the earlier jobs to decrease their shares to allow the more highly funded jobs to complete within their deadline. We again see that fewer hosts were given to user 3-5 but this time the performance level (latency) is better. We also see that these users pay a higher price for their resource usage, as expected.

5.4. Price Prediction Experiments and Simulations

In this set of experiments we run the same Grid application job load as in the previous experiments with the difference that we let the total budget of the users be random with a normal distribution.

Using the normal distribution analysis presented in Section 4.2, we provide a graph visualization of the price and performance guarantees a user may expect from a host. Depending on what guarantee of average performance the user wants, different curves may be followed to decide on how much to spend. For example, looking at the graph in Figure 3 a user who wants 90% guarantee that the CPU performance will be greater than 1.6GHz should spend $22/day when funding the host. There is a certain point where the curves flatten out, and that point would be the recommended budget to spend on that host to get the best performance per funding unit. For the given example it would not make sense for the user to spend more than roughly $60/day. We can also see that to get any kind of feasible performance out of the machine with at least a 80% guarantee the user needs to spend at least $10/day. In this example, we based our prediction on a time window of one day.

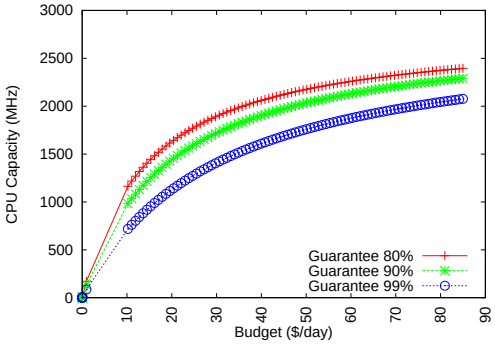


Figure 3. Normal distribution prediction with different guarantee levels

The basic AR model presented in Section 4.3 had problems predicting future prices due to sharp price drops when batch jobs completed. To overcome this issue we applied a smoothing function (cubic smoothing spline) before calculating the AR model. To verify the quality of the prediction we took a data sample of 40 hours of price history from our experimental run of Grid jobs described above. The first 20 hours were used to calculate the model and the last 20 hours were used to verify the model. The prediction error was then calculated as follows:

$$\varepsilon = \frac{1}{\mu_d n} \sum_{i=1}^{n} \sigma_i$$

where μ_d is the mean of the measured prices in the validation interval, n the number of data points in the validation interval, and σ_i the standard deviation of the prediction, measurement pair i. An AR(6) model with one hour forecasting (See Figure 4) yielded an ε of 8.96%, whereas a simple benchmark model always predicting the current price to remain for the next hour resulted in an ε of 9.44%.

Now, turning to portfolio theory (Section 4.4). There are some issues with this model concerning the definition

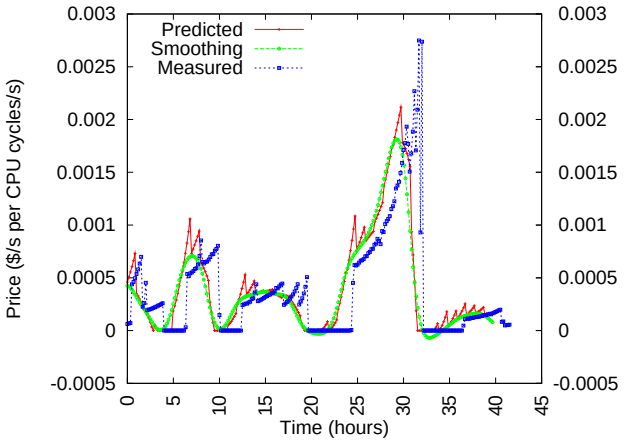


Figure 4. AR(6) prediction with a one hour forecast and smoothing function.

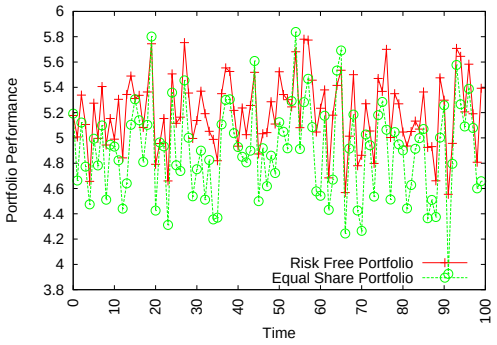


Figure 5. Risk free portfolio performance vs. equal share portfolio.

of risk and asymmetry of distributions as mentioned in Section 4.4, but we also noted in our experiments that a portfolio-based scheduler would not do as well in load balancing batches of user jobs coming in as the best response algorithm which bases its selection on the spot market price, and which could immediately move users away from high-bid machines. Portfolio theory may, however, prove useful for long term investments in hosts, e.g. when hosts should be bought to run a continuous application such as a web server. Another observation is that idle hosts tend to get 100% of the share in the portfolio, to avoid this behavior a larger time window needs to be used when collecting the mean and variance statistics from the hosts.

To test the risk hedging properties of the portfolios returned by our implementation we ran simulations where 10 hosts are picked either using the calculated risk free portfolio or equal shares. The aggregate performance over time is then measured. Individual mean host performance, performance variance, and variance of performance variances were all randomly generated with a normal distribution. The results, depicted in Figure 5 shows that downside risk could be improved by using the risk free portfolio.

Finally, we look at the distribution of prices over three time windows, a week, a day, and an hour. This data can be used to select an appropriate prediction model. For example, if the distribution resembles a normal distribution one could make use of the models described in Section 4.2, if the distribution is symmetric a portfolio analysis may be appropriate. A sample distribution graph is shown in Figure 6. It can be inferred from the graph that the prices exhibit signs of a heavy-tailed distribution (left-skewed) the last hour, mostly fall within the lowest price bracket, but are right-skewed, mostly in the most expensive bracket when considering a week or day-long window.

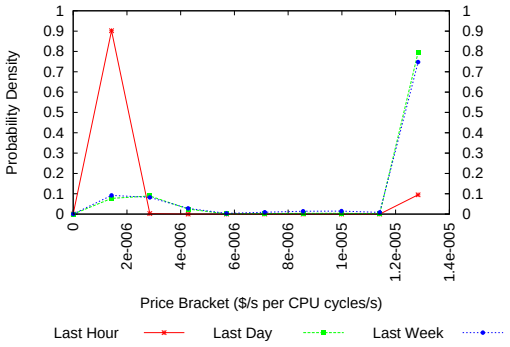


Figure 6. Price distribution within three different time windows.

To measure how accurate our window approximation is we ran a simulation of different distributions. Normal, Exponential and Beta Distributions were given a time lag of half the window size. At this point there is a maximum influence, or noise, from non-window data. The noise was generated using a uniform random distribution. We noted that normal distributions with a small standard deviation ($< 20\%$ of mean) could result in the approximation having its mean shifted slightly compared to the actual distribution. However, in general the approximations followed the actual distributions closely as seen in Figure 7 .

6. Related Work

Faucets [27] is a framework for providing market-driven selection of compute servers. Compute servers compete

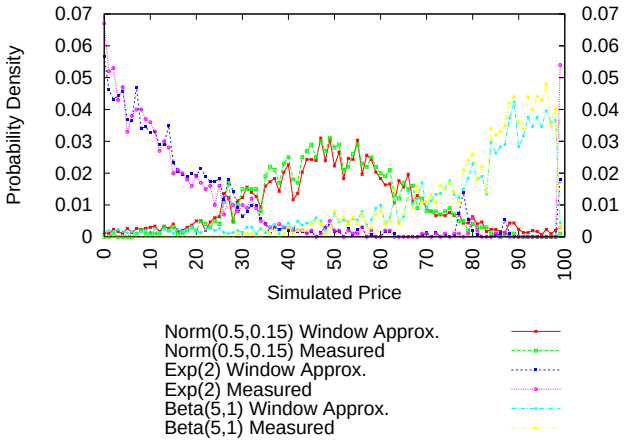


Figure 7. Window approximation of Normal, Exponential and Beta distributions.

for jobs by bidding out their resources. The bids are then matched with the requirements of the users by the Faucets schedulers. Adaptive jobs can shrink and grow depending on utilization and prioritization. QoS contracts decide how much a user is willing to pay for a job. The main difference to our work is that Faucet does not provide any mechanism for price setting. Further, it has no banking service, use central server based user-name password mechanisms, and does not virtualize resources.

Xiao et al. [43] suggest a model where users prioritize their jobs with different budgets and providers schedule jobs based on minimizing penalties from missing promised deadlines. It is argued that a user-initiated auction is more appropriate for lightly loaded system. From our experience with HPC projects like SweGrid, resources are scarce and their is competition for time slots, hence a seller-initiated auction is more appropriate for our work.

Chunlin and Layuan [17] propose a two-layered central market. In the first layer the users negotiate with services to meet deadline and budget constraints, in the second layer services purchase resources to meet the user demand. Service and resource prices are set by iteratively adjusting them up and down based on the measured demand and supply, until a market equilibrium is reached. In simulations they show that this model is more efficient in large Grids than a round-robin approach. Our work is less centralized, and thus more scalable and fail-safe, because all resource providers host their own markets.

G-commerce [41] is a Grid resource allocation system based on the commodity market model where providers decide the selling price after considering long-term profit and past performance. It is argued and shown in simulations that this model achieves better price predictability than auctions. However, the auctions used in the simulations are quite different from the ones we use. The simulated auctions are winner-takes-it-all auctions and not proportional share, leading to reduced fairness. Furthermore, the auctions are only performed locally and separately on all hosts leading to poor efficiency across a set of host. In Tycoon the Best Response algorithm ensures fair and efficient allocations across resources [20]. An interesting concept in G-commerce is that users get periodic budget allocations that may expire, which could be useful for controlling periodic resource allocations (as exemplified by NRAC and SNAC [37]) and to avoid price inflation. The price-setting and allocation model differs from our work in that resources are divided into static slots that are sold with a price based on expected revenue. The preemption and agile reallocation properties inherit in the bid-based proportional share allocation mechanism employed in our system to ensure work conservation and prevent starvation is, however, missing from the G-commerce model.

Buyya et al. [13] implement a completion time minimizing resource allocation algorithm for bag-of-task applications, utilizing an auctioneer infrastructure akin to the one deployed in Tycoon. The difference to the work presented here is that we use fixed budgets and the best response algorithm to place bids, as opposed to allowing bids to vary between a minimum and maximum value to meet deadlines. This allows us to make more precise statements about the fairness and efficiency of our solution in the equilibrium states.

Spawn [40], was one of the first implementations of a computational market, and Tycoon is an incarnation and evolution of many ideas presented in that work. Tycoon, in essence, extends Spawn by providing a Best Response agent for optimal and incentive-compatible bid distribution and host selection, and by virtualizing resources to give more fine-grained control over QoS enforcement. Tycoon also offers a more extensive price prediction infrastructure as presented in this paper. However, the general, continuous-bid and proportional share auction architecture is largely the same.

Other market based resource allocation systems, not focussing on Grid applications, have been presented in [39, 35, 15, 33]

7. Conclusions

We have presented an integrated Grid market of computational resources based on combining a market-based resource allocation system, Tycoon, and a Grid meta-scheduler and job submission framework, Nordugrid ARC.

One of the most challenging integration points was to map the Grid identity to an asserted capability. This prob-

lem was solved by introducing the concept of transfer tokens. This allowed both the private Grid credentials, and the bank account keys to remain local. It also makes it easy for resource users to give out 'gift certificates', to allow users without a Tycoon client installation to submit (and fund) jobs to the Tycoon cluster.

One of the first experiences gained from user feedback of the system was that it was hard to know how much money to use to fund a job. To aid the users in deciding how much funding their jobs would need to complete within a certain deadline, or conversely when a job would be expected to complete given a budget, we developed a suite of prediction models and tools. The accuracy of these predictions depends on the regularity of previous price snapshots and it is therefore crucial, for the results to be good, to pick a time window to study that exhibits these patterns. We therefore also implement a model that allows statistical data within a certain time window to be retrieved, using approximations based on linear smoothing functions.

Finally, our experimental results using a Bioinformatics application developed for the Grid, show that the level of performance delivered when submitting a large batch of jobs, can be customized by the incentive compatible use of transfer tokens. Thus the fairness and economically efficiency properties of Tycoon can be carried over to the Grid Market users.

Future work includes extending the lightweight prediction model presented here to handle arbitrary distributions and studying how higher-level reservation mechanisms, such as Service Level Agreements, Future Markets, Insurance Systems, and Swing Options can be built on top of the prediction infrastructure presented here to provide more user-oriented QoS guarantees.

8. Acknowledgments

We thank Bernardo Huberman, Lars Rasmusson, Fang Wu, Li Zhang, and Kate Keahey for fruitful discussions. The Tycoon Grid Market work would not have been possible without the funding from the HP/Intel Joint Innovation Program (JIP), our JIP liason, Rick McGeer, and our collaborators at Intel, Rob Knauerhase and Jeff Sedayao.

The work on the Bioinformatics application was funded by Wallenberg Consortium North Foundation and Vinnova.

References

[1] Yellow dog Updater, Modified. http://linux.duke.edu/projects/yum/.

[2] SUN, Queen's Univ, First Derivatives to Speed Bank Risk Analysis. *GRID today*, 3(30), July 2004.

[3] The Grid for the Video Industry. *GRIDSTART Business Newsletter*, (2):4, April 2004.

[4] EGEE. Enabling Grids for EsciencE. http://egee-intranet.web.cern.ch/egee-intranet/gateway.html, 2005.

[5] ESG. Earth System Grid. http://www.earthsystemgrid.org, 2005.

[6] NEESit. http://it.nees.org/, 2005.

[7] OSG. Open Science Grid. http://www.opensciencegrid.org, 2005.

[8] TeraGrid. http://www.teragrid.org, 2005.

[9] J. Andrade and J. Odeberg. HapGrid: a resource for haplotype reconstruction and analysis using the computational Grid power in Nordugrid. *HGM2004: New Technologies in Haplotyping and Genotyping*, April 2004.

[10] D. Bosio, J. Casey, A. Frohner, L. Guy, P. Kunszt, E. Laure, S. Lemaitre, L. Lucio, H. Stockinger, K. Stockinger, W. Bell, D. Cameron, G. McCance, P. Millar, J. Hahkala, N. Karlsson, V. Nenonen, M. Silander, O. Mulmo, G.-L. Volpato, G. Andronico, F. DiCarlo, L. Salconi, A. Domenici, R. Carvajal-Schiaffino, and F. Zini. Next-generation eu datagrid data management services. In *Proceedings of Computing in High Energy and Nuclear Physics*, La Jolla, CA, USA, March 2003.

[11] Brent N. Chun and Philip Buonadonna and Chaki Ng. Computational Risk Management for Building Highly Reliable Network Services. In *Proceedings of the 1st Workshop on Hot Topics in System Dependability*, 2005.

[12] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE, Special Issue on Grid Computing*, 93(3):479–484, March 2005.

[13] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. *Software: Practice and Experience (SPE) Journal*, 35(5):491–512, April 2005.

[14] G. Caronni, T. Curry, P. S. Pierre, and G. Scott. Supernets and snHubs: A Foundation for Public Utility Computing. Technical Report TR-2004-129, Sun Microsystems, 2004.

[15] A. Chavez, A. Moukas, and P. Maes. Challenger: a multiagent system for distributed resource allocation. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 323–331, New York, NY, USA, 1997. ACM Press.

[16] B. N. Chun, P. Buonadonna, and C. Ng. Computational Risk Management for Building Highly Reliable Network Services. In *Proceedings of the IEEE First Workshop on Hot Topics in System Dependability*, 2005.

[17] L. ChunLin and L. Layuan. A two level market model for resource allocation optimization in computational grid. In *CF '05: Proceedings of the 2nd conference on Computing frontiers*, pages 66–71, New York, NY, USA, 2005. ACM Press.

[18] S. Clearwater and B. A. Huberman. Swing Options. In *Proceedings of 11th International Conference on Computing in Economics*, June 2005.

[19] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[20] M. Feldman, K. Lai, and L. Zhang. A Price-Anticipating Resource Allocation Mechanism for Distributed Shared Clusters. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[21] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP'05: Proceedings of International Conference on Network and Parallel Computing*, volume 3799, pages 2–13. LNCS, Springer-Verlag GmbH, 2005.

[22] I. Foster and C. Kessleman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.

[23] The Future and Emerging Technologies Global Computing Initiative. Technical report, European Commision, DG Information Society, July 2005.

[24] S. Graupner, J. Pruyne, and S. Sherad. Making the Utility Data Center a Power Station for the Enterprise Grid. Technical Report HPL-2003-53, Hewlett-Packard Laboratories, 2003.

[25] L. J. Halliwell. *Mean-Variance Analysis and the Diversification of Risk*. Casualty Actuarial Society, St. Louis, Missouri, USA, May 1995.

[26] J. Hellerstein, K. Katricioglu, and M. Surendra. An Online, Business-Oriented Optimization of Performance and Availability for Utility Computing . Technical Report RC23325, IBM, December 2003.

[27] L. V. Kale, S. Kumar, M. Potnuru, J. DeSouza, and S. Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society.

[28] G. Kan. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter Gnutella, pages 94–122. O'Reilly & Associates, Inc., 1st edition, March 2001.

[29] K. Lai. Markets are Dead, Long Live Markets. *SIGecom Exchanges*, 5(4):1–10, July 2005.

[30] K. Lai, B. A. Huberman, and L. Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical report, arXiv, 2004. http://arxiv.org/abs/cs.DC/0404013.

[31] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. Technical Report arXiv:cs.DC/0412038, HP Labs, Palo Alto, CA, USA, Dec. 2004.

[32] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, December 1998.

[33] T. W. Malone, R. E. Fikes, K. R. Grant, and M. T. Howard. Enterprise: A Market-like Task Scheduler for Distributed Computing Environments. In B. A. Huberman, editor, *The Ecology of Computation*, number 2 in Studies in Computer Science and Artificial Intelligence, pages 177–205. Elsevier Science Publishers B.V., 1988.

[34] H. M. Markowitz. Portfolio Selection. *Journal of Finance*, 7(1), March 1952.

[35] O. Regev and N. Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of 1st International Conference on Information and Computation Economies*, pages 148–157, 1998.

[36] T. Sandholm. emediator: a next generation electronic commerce server. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*, pages 341–348, New York, NY, USA, 2000. ACM Press.

[37] T. Sandholm, P. Gardfjell, E. Elmroth, L. Johnsson, and O. Mulmo. An ogsa-based accounting system for allocation enforcement across hpc centers. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 279–288, New York, NY, USA, 2004. ACM Press.

[38] O. Smirnova, P. Erola, T. Ekelöf, M. Ellert, J. Hansen, A. Konsantinov, B. Konya, J. Nielsen, F. Ould-Saada, and A. Wäänänen. The NorduGrid Architecture and Middleware for Scientific Applications. *Lecture Notes in Computer Science*, 267:264–273, 2003.

[39] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: a wide-area distributed database system. *The VLDB Journal*, 5(1):048–063, 1996.

[40] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A Distributed Computational Economy. *Software Engineering*, 18(2):103–117, 1992.

[41] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, page 10046.2, Washington, DC, USA, 2001. IEEE Computer Society.

[42] F. Wu, L. Zhang, and B. A. Huberman. Truth-telling Reservations. http://arxiv.org/abs/cs/0508028, 2005.

[43] L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu. Gridis: An incentive-based grid scheduling. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 65.2, Washington, DC, USA, 2005. IEEE Computer Society.

EVALUATING DEMAND PREDICTION TECHNIQUES FOR COMPUTATIONAL MARKETS

THOMAS SANDHOLM
KTH – Royal Institute of Technolog,
SE-100 44 Stockholm, Sweden,
E-mail: sandholm@pdc.kth.se

KEVIN LAI
Informations Dynamics Laboratory,
HP Labs, Palo Alto, CA 94304
E-mail: kevin.lai@hp.com

We evaluate different prediction techniques to estimate future demand of resource usage in a computational market. Usage traces from the PlanetLab network are used to compare the prediction accuracy of models based on histograms, normal distribution approximation, maximum entropy, and autoregression theory. We particularly study the ability to predict the tail of the probability distribution in order to give guarantees of upper bounds of demand. We found that the maximum entropy model was particularly well suited to predict these upper bounds.

1. Introduction

Large scale shared computational Grids allow more efficient usage of resources through statistical multiplexing. Economic allocation of resources in such systems provide a variety of benefits including allocating resources to users who benefit from them the most, encouraging organizations to share resources, and providing accountability [12,6,1,14].

One critical issue for economic allocation systems is predictability. Users require the ability to predict future prices for resources so that they can plan their budgets. Without predictability, users will either over-spend, sacrificing future performance, or over-save, sacrificing current performance. Both lead to dissatisfaction and instability. Moreover, the lack of accurate information precludes rational behavior, which would disrupt the operation of the many allocation mechanisms that depend on rational behavior.

There are three parts to predictability: the predictability provided by the allocation mechanism, the predictability of the users' behavior, and the predictability provided by statistical algorithms used to model the behavior. We examine the latter two. Consequently, these results are not dependent on a specific allocation mechanism and instead apply to many systems.

The goal of this paper is to examine the degree to which future demand can be predicted from previous demand in a shared computing platform. Ideally, we would use the pricing

data from a heavily used economic grid system, but such systems have not yet been widely deployed. Instead, we examine PlanetLab[9], a widely-distributed, shared computing platform with a highly flexible and fluid allocation mechanism. The PlanetLab data set has the advantage of encompassing many users and hosts and having very little friction for allocation. However, PlanetLab does not use economic allocation, so we substitute usage as a proxy for pricing. Since many economic allocation mechanisms (e.g., Spawn[11], Popcorn[10], ICE[4], and Tycoon[7]) adjust prices in response to the demand, we believe that this approximation is appropriate.

We examine this data set using four different statistical prediction algorithms: histogram (Hist) approximation, maximum entropy (MaxEnt) density estimation, an autoregression (AR) time series model, and a normal (Norm) distribution model. We evaluated these algorithms by feeding them samples of usage data over a particular period of time and then measuring the error of the generated model. We then measured the error of using these models to predict future demand.

Our findings are as follows:

• MaxEnt and Norm were able to accurately model the data set over larger time periods. Maximum entropy estimation is approximately twice as accurate as a normal model because of its ability to capture skewness. Both methods are an order of magnitude more accurate than histogram approximation. The MaxEnt model is based on fitting integrals of the distribution function to statistical moments. This fit may not yield satisfactory approximations if the number of data samples in the time window investigated are too few, and we then fall back to the normal distribution approximation.

• All of the techniques produce inaccurate predictions, when trying to predict the cumulative distribution function for future demand. Autoregression has the additional disadvantage of requiring so much compute overhead that it was not able to complete some predictions. Furthermore, the AR model requires more history data to be maintained in order to retrain the prediction model to fit the current load.

• Despite inaccurate predictions of the full cumulative distribution function, MaxEnt and Norm were able to produce accurate *bounds* for demand. This is important because bounds are sufficient for users to budget. For example, if a user knows that the probability of hosts being less than $1 per host within the next week is 99%, and he needs 10 hosts, then he knows he should budget $10.

2. Prediction Algorithms

The goal of the prediction algorithms is to predict the demand for a resource based on historical data. In an economic system, the demand determines the price, which allows users to budget accurately. The general prediction model we use is summarized here.

$$P(Y \leq \frac{y-\mu}{\sigma}) = \Phi(\frac{y-\mu}{\sigma}) \tag{1}$$

$$y \leq \mu + \sigma\Phi^{-1}(p) \tag{2}$$

where y is the demand with mean μ and standard deviation σ, and Φ is the cumulative probability density function (CDF) of a normal distribution. Eq. 1 gives us a way to get a probability of a demand given its mean and standard deviation, and Eq. 2 allows us to find the demand corresponding to level of guarantee or probability.

In this work we want to remove the assumption of a normal distribution, and instead only assume an iid (independent identically distributed) distribution, and then compare the results to those obtained using the normal distribution assumption. More specifically, this means that we want to take the skewness of the distribution into consideration in our predictions. This extension is motivated by previous work on computational markets and usage behavior on the web [3] have shown that heavy-tailed distributions are common.

We evaluate three different approaches to tackle this generalization here, histogram (Hist) approximation, maximum entropy (MaxEnt) density estimation, and an autoregression (AR) time series model. The results are benchmarked against approximations used with the normal (Norm) distribution assumption, and compared to the real outcome.

The Hist approximation is based on placing sample data points in a fixed number of bins with predetermined data ranges. It therefore assumes some a-priori knowledge of the variance of the data. In our benchmarks we used 10 and 100 bins to approximate the distribution of values in a range of about 5000 distinct data values.

The MaxEnt model is based on the concept of choosing a distribution function which maximizes the entropy or randomness (or simply the unknown parameters) of a function given some characteristics such as statistical moments. This idea was first articulated by E.T. Jaynes in [5]. Cover and Thomas [2] then proved that all functions maximizing the entropy of a distribution are of a general form. For example, given the following constraints of the three moments about the origin μ_1, μ_2, μ_3

$$\int_{-\infty}^{\infty} f(x)dx = 1, \int_{-\infty}^{\infty} xf(x)dx = \mu_1, \int_{-\infty}^{\infty} x^2 f(x)dx = \mu_2, \int_{-\infty}^{\infty} x^3 f(x)dx = \mu_3$$

then the distribution function that maximizes the entropy has the form

$$f(x) = e^{\lambda_0 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3}$$

Now the problem of finding the distribution function f reduces to finding the λ parameters. Cover and Thomas suggests starting with the parameters known for a normal distribution and then "wiggle" them to find the best fit. In our implementation we performed this "wiggling" by applying the steepest descent iterative optimization algorithm described in [13]. In summary, we iteratively try to get closer to

$$\theta = \lambda_0, \lambda_1, \lambda_2, \lambda_3$$

by initializing it to the values know for a normal distribution and then assigning it subsequent values according to

$$\theta_{t+1} = \theta_t - \mathrm{H}^{-1}\mathrm{B}$$

where H is the Hessian matrix defined as

$$H_{k,j} = \int x^k x^j f(x, \theta_t) dx, 0 \leq k, j \leq 3$$

and B is the difference vector

$$B_k = \int x^k f(x, \theta_t) dx - \mu_k, 0 \leq k \leq 3$$

Note that we use the first three moments to capture the skewness of the distribution. Using more than three moments introduces irregular fluctuations which could prevent the algorithm from converging, and it also more easily runs into numerical limitations such us number overflows and round off errors.

The AR model [8] is a standard time-series model of the following form

$$X_t = \mu + \sum_{i=1}^{k} \alpha_i (X_{t-1} - \mu)$$

where μ is the measured mean in the training data, and k is the order (we used $k = 2$ in our benchmarks). The model parameters α_i are estimated by first calculating the autocorrelation vector for the training data and then solving the Yule-Walker equations. Note that the white-noise parameter has been omitted for simplicity.

Four different evaluations are performed on time series data using these techniques. First, we look at how well the summary data, such as bin density with Hist, the first three moments about the origin with MaxEnt, and the first two moments with Norm approximate the distribution described in the current period. If we have an iid distribution this should also give an indication of the possible accuracy of future predictions. Second, we look at how well predictions based on approximations of the cumulative density function in previous intervals can predict future distributions, and compare that to AR prediction results. Third, we look at how the actual distribution changes over time in the different intervals studied. Finally we look at how well the 99th percentile of the cumulative distribution function can be estimated in order to see how well guarantees can be given that the price will not exceed a certain value.

We also look at the convergence rate of the MaxEnt estimation. If it does not converge we, as previously mentioned, fall back to the Norm approach.

3. Results

We study usage time-series data, based on 5-minute snapshots of the aggregated number of PlanetLab slices allocated across the whole network. Data from two months (November-December 2005) were used. Training and future prediction horizons corresponding to predictions roughly from 2 hours to 3 days into the future were evaluated.

3.1. *Modelling*

In Figure 1 we can see that the MaxEnt approximation improves the accuracy of the CDF fit substantially compared to the normal distribution technique. SSE is the sum of the squares

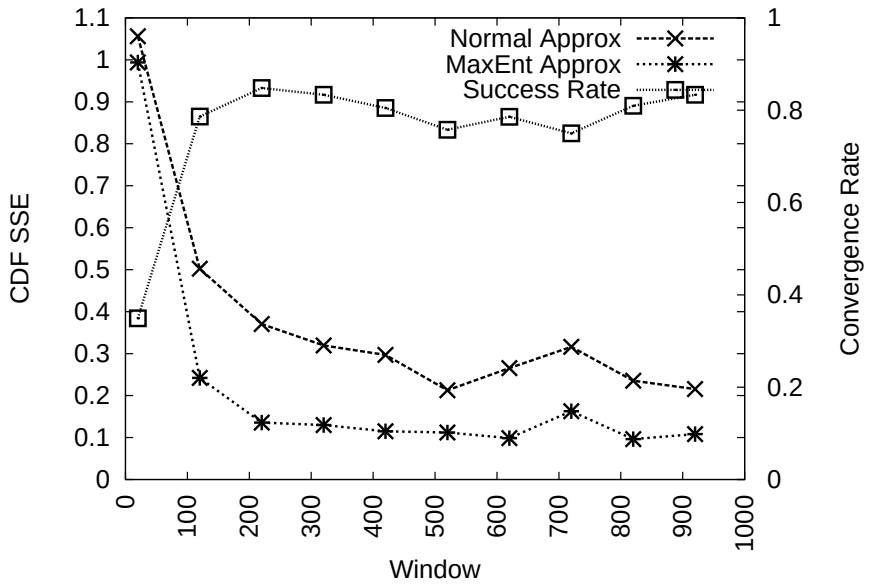


Figure 1. PlanetLab Density Approximation

of the errors when plotting the CDF with a granularity of 100 data points. The windows correspond to number of 5-minute snapshots used to predict the same number of 5-minute snapshots into the future.

We can see that the MaxEnt approximation does not converge in the case of the window size 50 in more than 35% of the cases. We wanted to investigate why, and performed a correlation test on the range of the data values in the window, the standard deviation of the data, and the likelihood of convergence. We obtained correlation coefficients 0.56, and 0.55 for data range and standard deviation respectively which are significant at the 1%-level according to a t-Student test. Intuitively this may be caused by the integral calculations used in the MaxEnt fit being too short to find the underlying entropy maximizing distribution. As a clarification, convergence of the MaxEnt approximation is defined by the error when fitting to the moments expected is less then a certain value ϵ. With the PlanetLab data we saw that an ϵ of 100 worked best, but there is always a tradeoff between accuracy and convergence rate.

3.2. *Predicting the Cumumlative Distribution*

Figure 2 shows an example of an interval estimation and how the different CDF functions compare. The window size in this case was two hours. We can see that the entropy model gives a much better fit to the non-normal behavior of the curve. The histogram estimation (with 100 bins) is quite a coarse grained estimation, and requires more state to be maintained as opposed to just three running moments as in the entropy case.

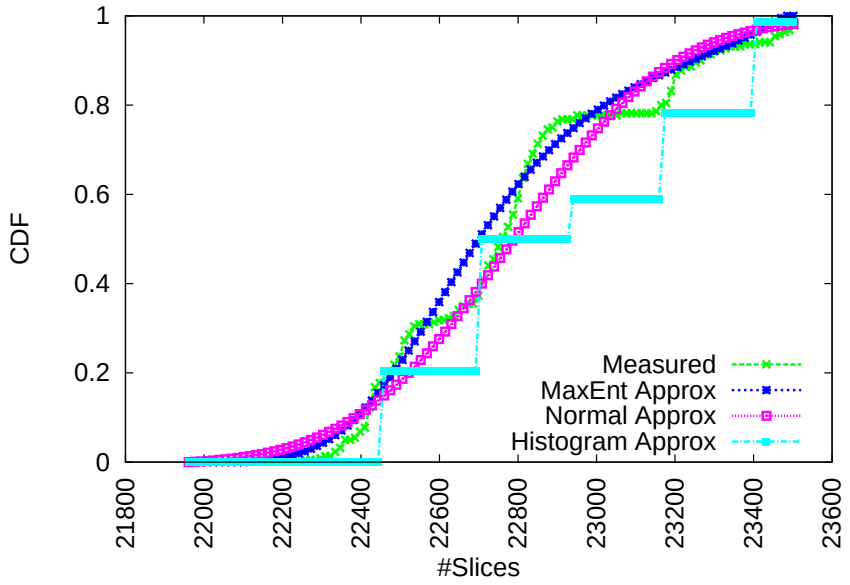


Figure 2. PlanetLab Density Approximation CDF

3.3. *Predicting Bounds*

A bit surprisingly we see in Figure 3 that the MaxEnt model does not produce better prediction results over time than the normal approximation. The AR curve is provided for reference. It does not make sense to use the AR model unless it predicts better than predicting the outcome of the previous period since it also requires all the data points to be kept in history. Since this is not the case for these long-interval predictions it provides no added value in this situation. Another severe limitation of AR is that it numerically due to large Matrix computations is not feasible to predict more than roughly 300 data points into the future. Note that in the graph this is shown by the AR SSE being set to 0 for window sizes greater than 300.

An explanation to why the MaxEnt model cannot benefit from its more accurate density approximations when predicting future densities can be seen in Figure 4. Each CDF in the Figure is taken in a subsequent interval so the t1 curve contains the distribution of all the data points from the start of the measurement to time t1, the t2 curve has all the data points between t1 and t2, etc. The mean point of the density moves back and forth in an unpredictable manner. Another indicator of this is the high SSE value of the benchmark prediction (predicting last periods CDF for the next) in Figure 3 (around 11) compare to the values in 1 (around 0.2).

It is then more encouraging to see that the 99th percentile MaxEnt estimates in Figure 5 are more accurate than with Norm. We should also note here that the training was done on the maximum amount of history data available and not just the previous period to do

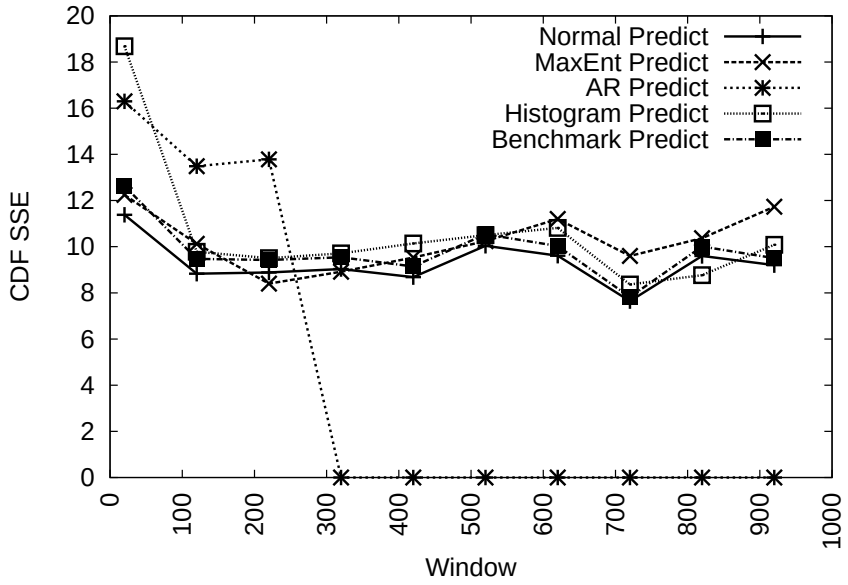


Figure 3. PlanetLab Density Prediction

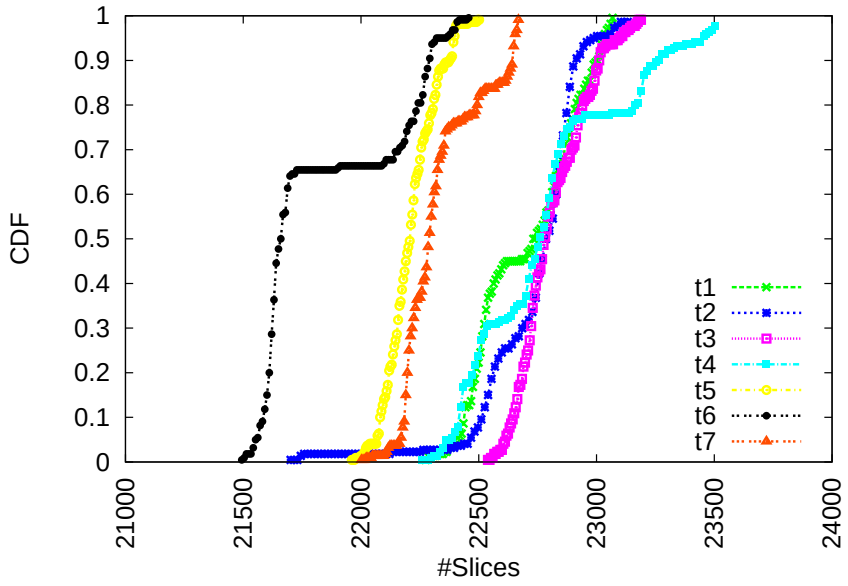


Figure 4. PlanetLab Density Variance

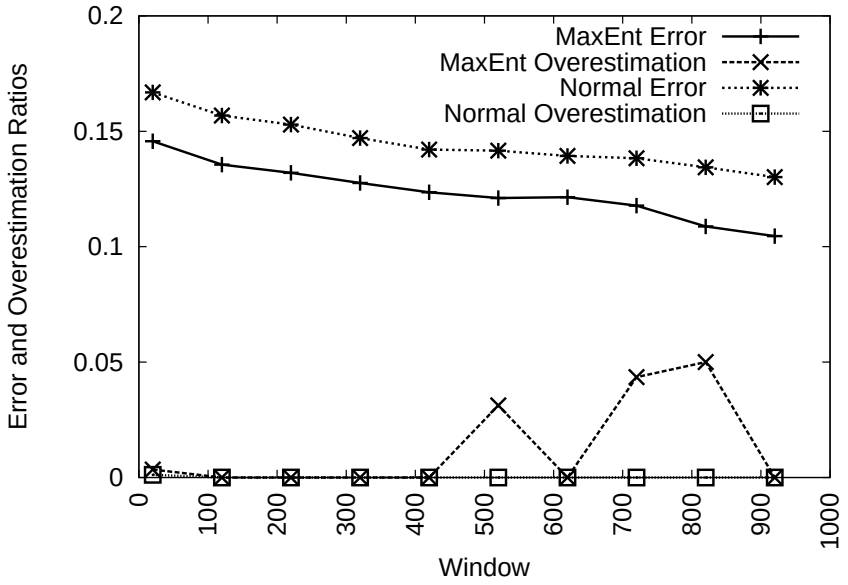


Figure 5. PlanetLab Tail Prediction

more of a worst case estimation of the tail as opposed to an overall accurate one. The error presented in Figure 5 is calculated as the difference between the measured value and the approximation divided by the measured value.

4. Conclusions

Although the statistical prediction algorithms that we examine here were not able to accurately predict future demand in the PlanetLab data set, we found that the MaxEnt algorithm was able to accurately predict bounds on future demand.

Some areas for future work are to examine the performance of MaxEnt in a live system and for systems with different applications and user behaviors than PlanetLab. Ultimately we hope to examine the performance of the algorithms in a live economic Grid system.

Given the fluidity of PlanetLab usage and the lack of a pricing mechanism to moderate usage, the accuracy of the MaxEnt algorithm gives us optimism that prediction algorithms will be accurate in real economic systems. We believe that this will ultimately lead to more stable, more economically efficient systems.

References

1. R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE, Special Issue on Grid Computing*, 93(3):479–484, March 2005.
2. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

3. C. A. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of WWW client-based traces. Technical Report TR-95-010, Boston University Department of Computer Science, Apr. 1995. Revised July 18, 1995.
4. David C. Parkes and Ruggiero Cavallo and Nick Elprin and Adam Juda and Sebastien Lahaie and Benjamin Lubin and Loizos Michael and Jeffrey Shneidman and Hassan Sultan. ICE: An Iterative Combinatorial Exchange. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.
5. E. Jaynes. Information Theory and Statistical Mechanisms. *Physics Review*, 106:620–630, 1957.
6. L. V. Kale, S. Kumar, M. Potnuru, J. DeSouza, and S. Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society.
7. K. Lai. Markets are Dead, Long Live Markets. *ACM SIGecom Exchanges*, 5(4):1–10, July 2005.
8. L. Ljung. *System Identification: Theory for the User*. Prentice Hall, December 1998.
9. L. Peterson, T. Anderson, D. Culler, , and T. Roscoe. Blueprint for Introducing Disruptive Technology into the Internet. In *First Workshop on Hot Topics in Networking*, 2002.
10. O. Regev and N. Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of 1st International Conference on Information and Computation Economies*, pages 148–157, 1998.
11. C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A Distributed Computational Economy. *Software Engineering*, 18(2):103–117, 1992.
12. R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, page 10046.2, Washington, DC, USA, 2001. IEEE Computer Society.
13. X. Wu and T. Stengos. Partially adaptive estimation via the maximum entropy densities. *Econometrics Journal*, 8(3):352–366, 2005.
14. L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu. Gridis: An incentive-based grid scheduling. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 65.2, Washington, DC, USA, 2005. IEEE Computer Society.

A Statistical Approach to Risk Mitigation in Computational Markets

Thomas Sandholm
KTH – Royal Institute of Technology
Center for Parallel Computers
SE-100 44 Stockholm, Sweden
sandholm@pdc.kth.se

Kevin Lai
Hewlett-Packard Laboratories
Information Dynamics Laboratory
Palo Alto, California 94304
kevin.lai@hp.com

ABSTRACT

We study stochastic models to mitigate the risk of poor Quality-of-Service (QoS) in computational markets. Consumers who purchase services expect both price and performance guarantees. They need to predict future demand to budget for sustained performance despite price fluctuations. Conversely, providers need to estimate demand to price future usage. The skewed and bursty nature of demand in large-scale computer networks challenges the common statistical assumptions of symmetry, independence, and stationarity. This discrepancy leads to underestimation of investment risk. We confirm this non-normal distribution behavior in our study of demand in computational markets.

The high agility of a dynamic resource market requires flexible, efficient, and adaptable predictions. Computational needs are typically expressed using performance levels, hence we estimate worst-case bounds of price distributions to mitigate the risk of missing execution deadlines.

To meet these needs, we use moving time windows of statistics to approximate price percentile functions. We use snapshots of summary statistics to calculate prediction intervals and estimate model uncertainty. Our approach is model-agnostic, distribution-free both in prices and prediction errors, and does not require extensive sampling nor manual parameter tuning. Our simulations and experiments show that a Chebyshev inequality model generates accurate predictions with minimal sample data requirements.

We also show that this approach mitigates the risk of dropped service level performance when selecting hosts to run a bag-of-task Grid application simulation in a live computational market cluster.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance—*Modeling and prediction*

HPDC'07, June 25–29, 2007, Monterey, California, USA.

General Terms

Management, Performance

Keywords

QoS, Service Level Management, Resource Allocation

1. INTRODUCTION

Large scale shared computational Grids allow more efficient usage of resources through statistical multiplexing. Economic allocation of resources in such systems provide a variety of benefits including allocating resources to users who benefit from them the most, encouraging organizations to share resources, and providing accountability [29, 15, 5, 31].

One critical issue for economic allocation systems is predictability. Users require the ability to predict future prices for resources so that they can plan their budgets. Without predictability, users will either over-spend, sacrificing future performance by prematurely spending the risk-hedging buffer, or over-save, sacrificing current performance. Both lead to dissatisfaction and instability. Moreover, the lack of accurate information precludes rational behavior, which would disrupt the operation of the many allocation mechanisms that depend on rational behavior.

There are three parts to predictability: the predictability provided by the allocation mechanism, the predictability of the users' behavior, and the predictability provided by statistical algorithms used to model the behavior. In this work we focus on the last part. The first part was investigated in [11] and examples of mechanisms addressing the second part include [7, 30].

The goal of this paper is to examine the degree to which future QoS levels, guarantees, and resource prices can be predicted from previous demand on a shared computing platform. Ideally, we would use the pricing data from a heavily used economic Grid system, but such systems have not yet been widely deployed. Instead, we examine PlanetLab[20], a widely-distributed, shared computing platform with a highly flexible and fluid allocation mechanism. The PlanetLab data set has the advantage of encompassing many users and hosts and having very little friction for allocation. However, PlanetLab does not use economic allocation, so we substitute usage as a proxy for pricing. Since many economic allocation mechanisms (e.g., Spawn[25], Popcorn[21], ICE[9], and Tycoon[16]) adjust prices in response to the demand, we believe that this approximation is appropriate.

We also analyze job traces from the Royal Institute of Technology in Stockholm (KTH), Ohio Supercomputing Center (OSC), and San Diego Supercomputer Center (SDSC) to contrast the PlanetLab load with deployments exhibiting more traditional high performance computing user behavior.

In these traces, we examine three key statistical metrics computed on the time-varying demand: distribution skewness, correlation over time, and volatility over time (heteroskedacity). Skewness captures whether most of the distribution's values are closer to the lower bound (right-skewed), upper bound (left-skewed), or neither (symmetric). Correlation over time measures the relative importance of old and recent samples in predicting future samples. Finally, heteroskedacity measures how variance changes over time. We find that that the traced demand distributions have significant skewness, long-term correlation, and changes in volatility over time.

Common modelling techniques fail under these conditions. We show using simulation that a predictor based on a Normal (Norm) Gaussian distribution model and a benchmark (Bench) predictor using the entire sample history both perform poorly with highly skewed data. In addition, high heteroskedacity reduces the accuracy of any predictions of the expected value of demand.

We develop a new predictor based on the Chebyshev (Cheb) model. It is distribution-free and provides worst-case scenario bounds independent of distribution symmetry. To handle time correlations, we sample statistics in running moments with exponential smoothing in different time windows (e.g., hour, day and week). This allows us to maintain less data as well as to adapt quickly to changing demand behavior.

To handle heteroskedacity, the Chebyshev-based predictor uses the running moments to predict demand bounds. We observe that users of a market-based system can substitute predictions of demand bounds for predictions of expected value. Market-based systems have mechanisms for insuring against future changes in prices (e.g., bidding more for resources). Demand bound predictions are sufficient for users to decide how much money to spend to mitigate risk. For example, a predicted price bound of $[\$9, \$50]$ per GHz with an expected price of \$10 conveys that, despite low expected demand, there is significant risk of a high price and the user should spend more to mitigate risk.

We use a live computational market based on the Tycoon[16] resource allocator to compare a strategy using the Chebyshev-based predictor to the default strategy of using the spot market price when scheduling jobs. Using a varying load over time, we show that the predictor outperforms the spot market strategy while imposing a negligible (0.001%) computational and storage overhead.

The remainder of this paper is structured as follows. In Section 2 we outline the requirements of the prediction problem we are addressing. We analyze the demand traces by comparing the dynamics of the the different systems under investigation in Section 3. In Section 4, we describe in more detail the design and rationale of our prediction approach and models. In Section 5, we present and discuss the setup and results of the simulations and experiments. We review related work in Section 6 and conclude in Section 7.

2. REQUIREMENTS

2.1 User Requirements

Consumers in a computational market need guidance regarding how much to spend to meet their performance requirements and task deadlines. The quality of this guidance is critical to both individual and overall system performance. The best prediction system would recommend users to invest as little as possible while meeting their QoS requirements with as high a probability as possible. Robust statistical performance guarantees are an alternative to reservations with absolute guarantees, which are vulnerable to speculation and a variety of other gaming strategies [6].

In the discussion below, the *bid* is the amount the consumer pays a provider for a resource share over a given period of time. A higher bid obtains a greater share, or *QoS level*. The *guarantee* is the likelihood that the delivered QoS is greater than the consumer's defined value. The guarantee is not a contract, but rather the estimated probability that a QoS level can be delivered.

The prediction system answers the following questions for the user:

QUESTION 1. *How much should I bid to obtain an expected QoS level with a given guarantee?*

QUESTION 2. *What QoS level can I expect with a given guarantee if I spend a given amount?*

QUESTION 3. *What guarantee can I expect for a given QoS level if I spend a given amount?*

Exactly which question(s) a user asks varies from job to job and from user to user. We assume that users can ask any question at any time.

We further assume that providers allocate shares among consumers using the proportional share model as follows:

Definition 1. $Q = \frac{bid}{bid+Y}$

where Y denotes the demand of the resource modeled as a random variable with an arbitrary unknown distribution, and Q is the random variable representing the obtained performance when requesting performance qos. We define guarantee as the probability that the obtained performance is greater than the requested performance:

Definition 2. $g = P(Q > qos)$

PROPOSITION 1. *To get performance guarantee g, we need to spend $\frac{CDF^{-1}(g)qos}{1-qos}$, where CDF^{-1} is the percent point function or the inverse of the cumulative distribution function for the demand.*

PROOF. Substituting the share Q in Definition 2 with the right side of the equation in Definition 1 gives

$$g = P\left(\frac{bid}{bid+Y} > qos\right) = P\left(Y \leq \frac{bid}{qos} - bid\right) =$$

$$CDF\left(\frac{bid}{qos} - bid\right)$$

the inverse CDF of the demand distribution can then be expressed as:

$$CDF^{-1}(g) = \frac{bid}{qos} - bid$$

which after rearranging gives:

$$bid = \frac{CDF^{-1}(g)qos}{1-qos}$$

□

This provides everything to answer Question 1, 2, 3. To obtain an expected QoS level with a given guarantee, a user should bid the following (Question 1):

$$bid = \left.\frac{CDF^{-1}(g)q}{1-q}\right|_{P(Q>q)=g}. \quad (1)$$

A user who bids a given amount and expects a given guarantee should expect the following QoS level (Question 2):

$$q = \left.\frac{bid}{bid + CDF^{-1}(g)}\right|_{P(Q>q)=g}. \quad (2)$$

A user who bids a given amount and expects a given QoS level should expect the following guarantee (Question 3):

$$g = CDF\left(\left.\frac{(1-q)bid}{q}\right)\right|_{P(Q>q)=g} \quad (3)$$

The main goal of our prediction method is to construct estimates of the CDF and CDF^{-1} (a.k.a. percent point function (PPF)) accurately and efficiently.

2.2 System Requirements

Two different approaches for estimating probability densities stand out: parameter-based and parameter-free estimation. In the parameter-free approach, one takes a random sample of data points, and smooths them to calculate the most likely real underlying distribution, e.g. using a least-squares algorithm. In the parameter-based approach, one assumes certain structural and behavioral characteristics about the real distribution and finds the parameters that best fit the measured data, e.g. using some maximum likelihood algorithm. In either case, sample measurements or data points are needed to calculate the density functions. Recording the history of demand in time-series streams for a large number of resources across a large number of computational nodes in real-time does not scale, in terms of state maintenance and distribution, and prediction model construction and execution. The scalability limitations force restrictions on the length of past and future prediction horizons and the number of resources that can be tracked. As a result, our goal is to use as few distribution summary data points as possible to make as flexible predictions as possible.

Studies of large-scale networked systems [19, 8] show that the underlying distribution of the demand is neither normal nor symmetric. Assuming that it is would result in underestimated risks, so accommodating bursty, skewed behavior is a necessity. Furthermore, we neither want to assume stationarity nor independence of the underlying distribution since consumers are interested in getting the most accurate estimate based on performing a task in the near future, and evaluate that option against waiting for better conditions.

There is a trade-off between performance and accuracy of the predictions, but there is also a similar trade-off between flexibility and evaluation capability. We would like to empower users to do rich customized predictions based on minimal summary statistics. They should be able to execute what-if scenario probes based on all three questions mentioned in Section 2.1. Different questions incur different prediction errors, which complicates the generic evaluation of model uncertainty and construction of prediction intervals.

3. DEMAND ANALYSIS

In this section we describe the data traces used in the simulations in Section 5. The load traces come from four shared compute clusters. The PlanetLab trace is available on-line at `http://comon.cs.princeton.edu/`. The San Diego Supercomputer Center, Ohio Supercomputing Center, and Stockholm Royal Institute of Technology traces are all available at `http://www.cs.huji.ac.il/labs/parallel/workload/`.

• **PlanetLab.** PlanetLab (PL) is a *planetary-scale*, distributed computing platform comprising approximately 726 machines at 354 sites in 25 countries, all running the same Linux based operating system and PlanetLab software stack, including a virtual machine monitor. The user community is predominantly computer science researchers performing large-scale networking algorithm and system experiments. Resources are allocated in a proportional share fashion, in virtual machine slices. The trace is from December 2005 to December 2006, and was obtained from PlanetLab CoMon node-level statistics. We calculate demand by aggregating the load value across all hosts for each 5-min snapshot interval available. This load measures the number of processes that are ready to run across all virtual machines. The load data was filtered to remove a periodic peak caused by synchronized rpm database updates across all slices appearing over a 30-min period every night, to get a more accurate representation of demand.

• **San Diego Supercomputer Center.** The San Diego Supercomputer (SDSC) trace was obtained from Dror Freitelson's parallel workloads archive [2]. It is the SDSC Blue Horizon workload from June 2000 to June 2001 (SDSC-BLUE-2000-3.1-cln.swf). The load is from the 144 node 8-way SMP crossbar Blue Horizon cluster using the LoadLeveler scheduler.

The demand is calculated in three steps. First, the CPU usage or load for a job is obtained as: $r_t(t_e - t_s)$, where r_t is the CPU time consumed by the job in percentage of the total run time, and t_e and t_s are the end time and start time in seconds since epoch respectively–all three values are available directly from the trace. Second, each job CPU usage is correlated to the time when it was submitted, thus effectively simulating that no queuing was done but all jobs could run with their actual load instantly. Finally, we aggregate the obtained CPU load value across all jobs running in every 5-min time interval. We did not analyze the utilization data directly because it could mask high demand under heavy load. The transformation also makes it comparable to proportional share systems such as PlanetLab and Tycoon, which are the primary focus of our work. Although this masks the needs of users who do not submit jobs when the queue wait-times are too long, we assume that these users would not spend money in an expensive computational market either. Consequently, we assume that these users do not contribute to demand.

• **Ohio Supercomputing Center.** The Ohio Supercomputing Center (OSC) trace was also obtained from the parallel workloads archive. It is the OSC Linux cluster workload from January 2001 to December 2001 (OSC-Clust-2000-2.swf). The site is a Linux cluster with 32 quad nodes and

Table 1: Central Moments of Traces (skewness > 2 is marked in bold to indicate a heavy tail)

	μ	$\frac{\sigma}{\mu}$	γ_1	γ_2
PL	3433	0.37	**4.06**	29.45
KTH	382	0.71	1.11	0.90
OSC	66	0.72	1.53	4.35
SDSC	3249	0.51	1.02	2.07

25 dual nodes with a total of 178 processors using the Maui scheduler. We perform the identical transformation from job workload to demand as with the SDSC data described above.

• **Royal Institute of Technology.** The Center for Parallel Computers at the Swedish Royal Institute of Technology (KTH) in Stockholm, provided us with the final workload trace. The trace is from a 100-node IBM SP2 cluster from October 1996 to September 1997 (KTH-SP2-1996-2.swf). Because CPU time was not recorded in the logs, the CPU load is set to the run time of the jobs. Apart from this the demand transformation is identical to the SDSC and OSC traces described above.

Next, we characterize the dynamics of the computational demand traces by examining the typical properties of time series distributions: symmetry, independence, and stationarity.

3.1 Distribution Symmetry

The first step towards characterizing the load and detecting anomalies is to look at the raw demand traces. Figure 1 shows that PlanetLab exhibits much thinner peaks that both appear and disappear more quickly than the peaks in the other traces. We attribute this behavior to the fact that PlanetLab jobs tend to be distributed systems or networking experiments that have lower resource requirement than scientific computing workloads.

Next, we measure the distribution symmetry of the demand for the different clusters in Figure 2. A distribution is right-skewed if its right tail is longer than its left and its mass leans to the left. We see that the PlanetLab load stands out as being more right-skewed than the others. All traces, however, show asymmetric right-skewed behavior, indicating that low demand is the most common state of the systems. Distribution models such as the Gaussian or Normal distribution assumes symmetry and will thus be inaccurate for all of the traces studied. The central moments are summarized in Table 1.

3.2 Dependence and Long Memory

One of the most common assumptions when studying time series and when sampling data to approximate distributions and densities is that the observations are IID. I.e. the sampled data points are independent and identically distributed. This allows the models to be trained once and then reused indefinitely when they have converged. It also simplifies the construction of confidence and prediction intervals. Because there is no bias in the samples they can be taken to be a good representation of the whole data set. The simplest way of testing dependence, seasonality and randomness of samples is to draw an auto-correlation function (ACF) plot with a series of increasing time lags. We study the correlations

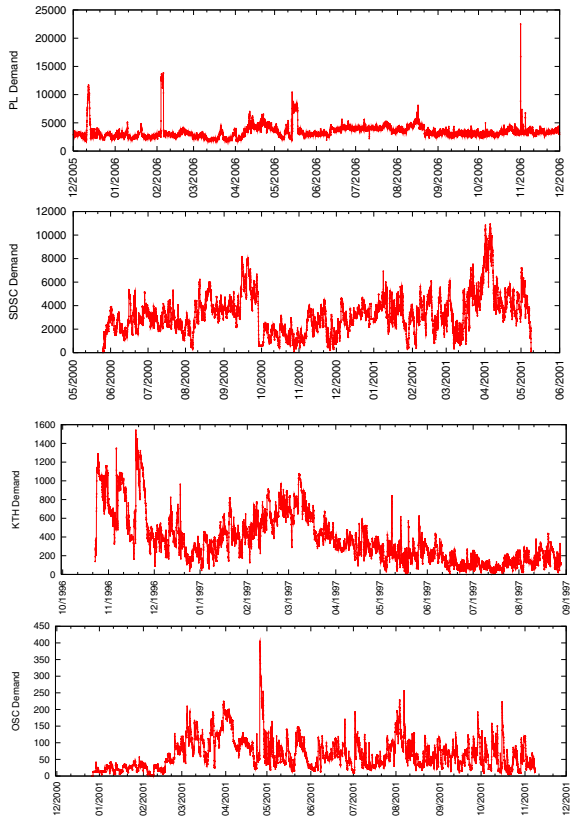


Figure 1: Demand History (Hourly)

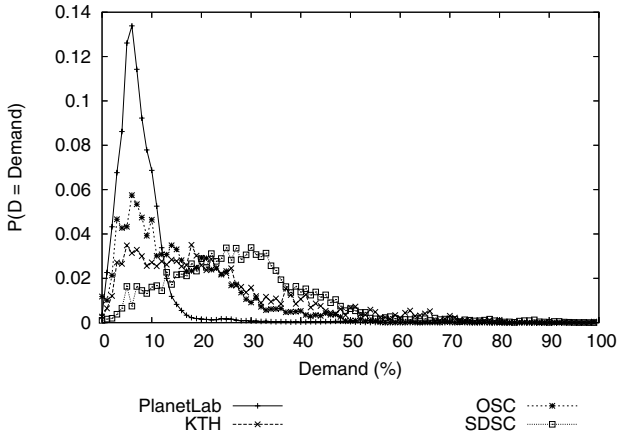


Figure 2: Demand Density

for lags up to 7 days. The plots in Figure 3, clearly show that the observations are not independent in time but rather show a strong and slowly decaying correlation to measures in the previous time interval. If the decay is slower than exponential the data is said to exhibit long memory. This is clearly the case for at least the KTH trace (within the weekly time lag). Long memory is a feature that gives standard auto-regressive models such as GARCH problems [18].

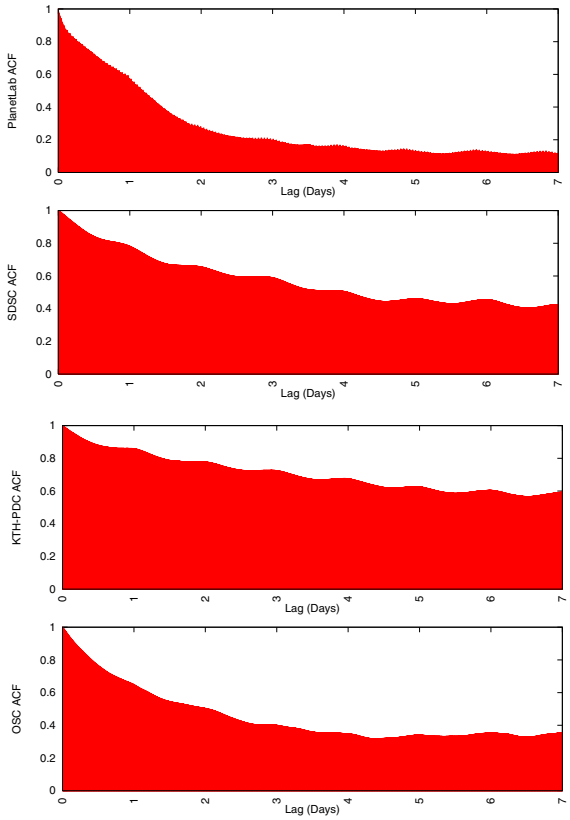


Figure 3: Auto-Correlation Functions

Another popular approach used to detect long-term correlations is the rescaled-range (R/S) analysis [19], which was influenced by Hurst's study of the Nile floods [14]. In general it provides a way to investigate how the values in a time series scale with time. The scaling index, known as the Hurst exponent is 0.5 if all the increments of the time series are independent. A series where a trend in the past makes the same trend in the future more likely is called presistent, and has a Hurst exponent greater than 0.5. Conversely, systems where past trends are likely to be reversed are called anti-persistent and scale with a Hurst exponent less than 0.5. If the R/S values (increment range, standard deviation ratio) for different time intervals are plotted against time on a log-log scale, the Hurst exponent appears as the slope of the fitted line. Figure 4 shows the R/S plot for the demand traces. A Hurst exponent around 0.92 fits all the traces under investigation, which indicates a very high correlation between past and future demand trends.

3.3 Heteroskedacity and Fat Tails

The last property that we investigate is the general volatility of the data which is crucial for making good risk assess-

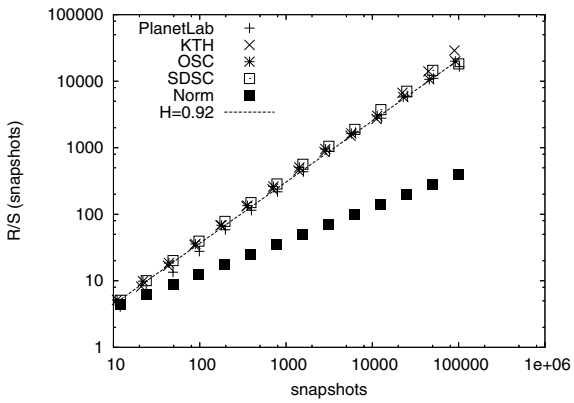


Figure 4: Rescaled-Range Dynamics and Hurst Exponent

ments. If the data is extremely skewed such as in power-law distributed data, both the mean and the variance can be infinite and thus some other measures of volatility need to be used. One popular approach is to look at the (absolute) log difference in the increments of the data. It turns out that even for very risky and volatile time series with power-law behavior like the stock-market, the absolute increments are predictable since high volatility data points tend to cluster [19]. A volatility graph showing the log-transformed increment differences over time is also another measurement of how Gaussian-like the data is. A Gaussian distribution produced by a Brownian-motion process has a near uniform distribution of increment sizes, whereas more unpredictable and thereby riskier processes have clusters of high volatility intermingled with lower volatility periods. In Figure 5 all of the traces show signs of changing volatility over time (heteroskedacity). High volatility instances also seem to be clustered. An analysis of how they are clustered is beyond the scope of this paper. The stock market has been shown to exhibit power-law scaling in the distribution of the volatility over time [19]. We therefore also look at the distribution tail behavior for our traces. A heavy-tailed or fat-tailed distribution will exhibit a longer tail of the complement of the cumulative distribution function (1-CDF) than the exponential distribution. According to this definition Figure 6 and Table 2 show that all traces are heavy-tailed in hourly volatility. PL and SDSC are also heavy-tailed in daily volatility.

This investigation of traces indicates that a multi-fractal time-scaling (trading time deformation) [18, 19] model may be appropriate. We, for example, note that the Hurst exponent obtained can be used to determine the fractal dimension [3], which is a typical measure of the *roughness* of the system in multifractal time-series. Figure 6 also indicates that a stretched exponential distribution [8] could be a good fit. However, an analysis of these more complicated distributional models are outside the scope of this paper.

4. PREDICTION APPROACH

In this section, we present our approach to providing accurate distribution predictions with an upper prediction bound. The method is agnostic to the model used to fit the time series data of demand, and the prediction error distribution.

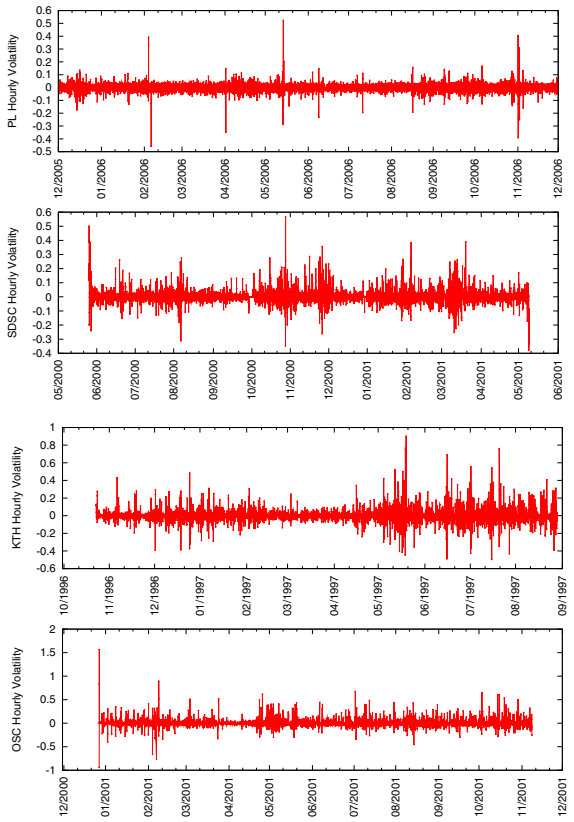


Figure 5: Hourly Demand Volatility

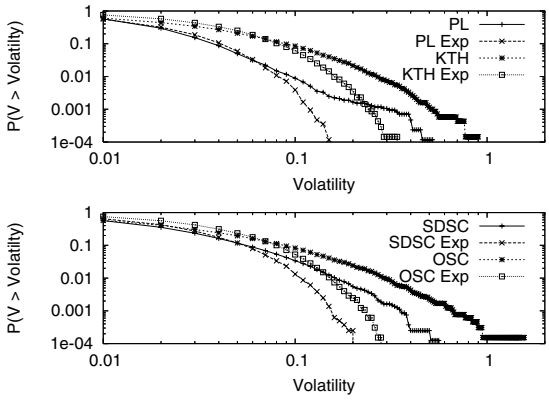


Figure 6: Heavy Tails for Hourly Demand Volatility

Table 2: Skewness of Volatility

	hour	day	week
PL	**7.45**	**4.55**	1.45
KTH	**4.40**	1.87	1.13
OSC	**6.41**	1.82	0.53
SDSC	**4.44**	**2.05**	1.63

There are two architectural components providing prediction capabilities: the *statistics collector* (SC), and the *prediction generator* (PG). The SC consumes a time series of prices and produces a statistical summary, which in turn is consumed by the PG. The statistical summary comprises instantaneous running (non-central) moments over configurable time horizons. In our case we provide hourly, daily and weekly summaries, as they correspond best to the length of the computational jobs run as well as the length of the horizon that is typically predictable. In addition to the moments, the summary also has the current price, a short history of moments for the most recent time periods, and the minimum and maximum measured price values.

The running non-central moments are calculated as follows:

$$\mu_{t,p} = \alpha\mu_{t-1,p} + (1-\alpha)x_t^p$$

where $\mu_{t,p}$ is the pth moment at time t, x_t the price at time t and $\alpha = 1 - \frac{1}{n}$, where n is the number of data points in the time horizon covered by the moments, and $\mu_{0,p} = x_0^p$. We refer to [22] for further details on how the central moments are calculated.

The PG component is instantiated with a predictor that uses the moments and the extremes to construct approximations of the cumulative distribution function (CDF), percent point function (PPF), and a function generating confidence intervals. The history of moments is used to construct prediction intervals.

Here we will describe a Gaussian (Norm), a Chebyshev (Cheb), and a sample-based predictor (Bench) which we use for benchmarking.

4.1 Gaussian Predictor

The Gaussian CDF (Φ) is readily available and can be calculated directly from the first two central moments. Since the inverse of the Gaussian CDF or PPF (Φ^{-1}) does not have a closed form, we use an approximation based on a rational minimax algorithm developed by Acklam [1]. The $100p$%-confidence interval is then calculated as $[\Phi^{-1}(0.5 - \frac{p}{2}), \Phi^{-1}(0.5 + \frac{p}{2})]$. An identical interval calculation can be done for all other predictors using their respective percent point functions. The prediction interval is calculated by applying Φ and Φ^{-1} on the history of moments.

4.2 Chebyshev Predictor

When predicting performance guarantees we are typically more interested in worst case scenario bounds as opposed to perfect data density fitting across all percentiles. One of the most prominent techniques to estimate bounds in probability theory is by means of the Chebyshev inequality [12], which states that:

$$P(|Y - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

where μ is the first central moment (mean) and σ the second central moment (standard deviation) of the underlying distribution. Rewriting the inequality as a CDF we get:

$$CDF(y) = 1 - \frac{1}{1+k^2}$$

where k is $\frac{y-\mu}{\sigma}$. For unimodal distributions we can tighten the bound further by applying the Vysochanskij-Petunin in-

equality [24], which for $k \geq \sqrt{\frac{8}{3}}$ gives:

$$CDF(y) = 1 - \frac{4}{9k^2}$$

Taking the inverse of the CDF we get:

$$PPF(p) = \begin{cases} \mu \pm \sigma\sqrt{\frac{1}{1-p} - 1} & , k < \sqrt{\frac{8}{3}}, \\ \mu \pm \sigma\sqrt{\frac{4}{9-p}} & , k \geq \sqrt{\frac{8}{3}}. \end{cases}$$

Since Chebyshev only gives us upper and lower bounds we cannot calculate the percentiles around the mean accurately, but this is not a great limitation in our case where we are primarily interested in worst-case scenario (upper) bounds.

The confidence and prediction intervals are calculated in the same way as in the Gaussian case.

4.3 Sample Bench Predictor

We use a benchmark predictor to compare our summary statistics predictors with a sample-based predictor. This predictor has access to the entire past history of data points and calculates the percentile points, cumulative distributions, and prediction bounds from the raw data sampled. The benchmark predictor could not be used in practice because of its prohibitive computational and storage requirements.

4.4 Multi-Host Predictions

We combine the results from Equation 1, Equation 2 and Equation 3 with our predictors to assess risk when making scheduling decisions across a set of hosts. For this purpose we extend an economic scheduling algorithm previously presented in [11, 22]. The purpose of this Best Response algorithm is to distribute a total budget across a set of hosts to yield the maximum utility for the user. The optimization problem, as seen by a user, is defined as:

$$\text{maximize } U = \sum_{j=1}^{n} w_j \frac{b_j}{b_j + y_j} \text{ subject to}$$

$$\sum_{j=1}^{n} b_j = bid, \text{ and } b_j \geq 0.$$

where w_j is the preference or weight for host j specified by the user, b_j is the bid that the user puts on host j, and *bid* the total budget of the user. We replace y_j, the spot price of host j, with the stochastic variable Y as modeled above. In Equation 1, which gives an expected performance value given a percentile and a (prediction) confidence level, we calculate Y with the PPF of the predictor. To calculate the *bid* given a performance level, a *guarantee* and a confidence level or to calculate the *guarantee* given a *bid* and a performance level, we numerically invert the Best Response algorithm. This allows us to probe different *bid* or *guarantee* levels until we get an acceptable performance match or we encounter budget constraints.

Users can use this model in a variety of ways. They can use the spot prices to determine whether they can meet their deadline, guarantee, and budget constraints, or if they should submit their job at a later time. Instead of the spot price, users can also use the statistical guarantees and prediction bounds to pick the hosts with the best sustained low-price behavior for long running tasks. We examine the effectiveness of the model for these use cases in the next section.

5. SIMULATION AND EXPERIMENT RESULTS

This section contains four different validators of our approach presented in the previous section. First, we run a simulation with generated random distributions against our predictors. Second, we run a prediction simulation using the compute cluster demand traces described in Section 3. Third, we run an experiment in our own live computational market cluster, Tycoon, comparing spot market scheduling with our extended risk mitigating scheduler described in the previous section. Finally, we run an efficiency experiment to measure the overhead incurred by the predictions.

5.1 Asymmetry Modeling Simulation

To validate the ability to approximate arbitrary skewed distributions we developed a generator capable of producing random data with distributions in a continuum from a right-skewed Pareto through Gaussian to a mirrored left-skewed Pareto with the first two moments kept stable and only the skewness varying. An example of distributions generated can be seen in the lower graph in Figure 7. The upper graph shows the result for the Cheb and Gauss predictors. As expected, Cheb gives better approximations for left and right-skewed distributions, whereas the Gauss predictor performs best for the near-symmetrical distributions (skewness near 0).

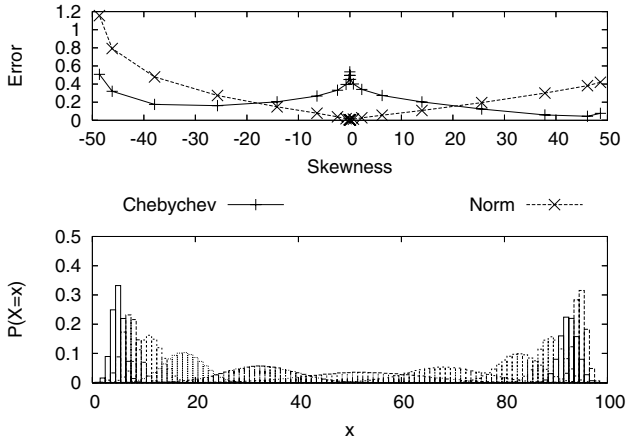


Figure 7: Fitting Skewed Distribution

5.2 Demand Prediction Simulation

We use the compute cluster demand traces from PlanetLab, KTH, SDSC, and OSC to study the ability of our prediction approach to give accurate risk assessments with the Cheb, Gauss, and Bench predictors. Recall that the Bench predictor simply uses all previous historical data to make PPF and prediction (confidence) interval estimates, whereas the other predictors base their estimates on the summary statistics generated by the statistics collector (SC) component (including running moments, and moment history).

The setup of the experiment is as follows. For each trace, we feed the time series data into the SC component configured for hourly, daily and weekly prediction horizons, and then try to make a prediction of the 95th percentile price

with a 90% confidence for the subsequent interval (for which no data is revealed) using the prediction generator (PG) instantiated with the Cheb, Gauss and Bench predictors. We then measure the delivered guarantee as the likelihood that at least 95% of the demand values are less than the predicted upper prediction confidence bound, which we denote the success rate (S). We also track the width of the prediction bound (B) as the difference between the actual 95th percentile demand and the predicted demand. The PG component is configured to track three historical running moment snapshots into the past of the first two non-central moments. The results are shown in Table 3. S and B are defined as:

$$S = P(\hat{f}(0.95) > f(0.95))$$

and

$$B = E\left(\frac{|\hat{f}(0.95) - f(0.95)|}{f(0.95)}\right)$$

where f is the actual percentile function of the price, and $\hat{f}$ is the predictor estimated percentile function of the price.

The Cheb predictor generates consistent and accurate success rates (S) for all traces across all prediction horizons. For daily and weekly horizons the prediction bound size (B) is in most cases very wide, which is likely to cause risk-averse users or users with a very limited budget to delay their job submissions. Both the Norm and the Bench predictors underestimate the risk grossly as seen by very low success rates, although the bounds are tight.

The Norm predictor would yield better success rates if we provided additional moment history snapshots. However, one of our requirements is to maintain system scalability and flexibility, so we must minimize the number of statistical summary points to reduce the size of the snapshots.

We note that using a horizon size of one and an infinite snapshot size in the SC component would make the summary statistics results converge to the results obtained by the Bench predictor.

5.3 Risk Mitigation Experiment

To experimentally validate our approach, we run a scheduling benchmark in a live Tycoon computational market. We submit jobs using the NorduGrid/ARC Grid meta-scheduler [23] and schedule locally using the extended Best Response algorithm described in Section 4. Tycoon uses the Xen virtual machine monitor [10] to host running jobs. Each job is run in a separate, dedicated, isolated machine.

The design rationale behind this experiment is to create a changing usage pattern that could potentially be predicted, and to study how well our approach adapts to this pattern under heavy load. We do not claim that the traffic pattern is representative of a real system. See the analysis in Section 5.2 to see how the Tycoon predictors handle real-world usage patterns.

The experiment consists of two independent runs with 720 virtual machines created on 60 physical machines during each 6 hour run. All jobs run on dedicated virtual machines that are configured based on the current demand, and job resource requirements. All jobs request 800MB of disk, 512MB of memory and $1 - 100\%$ CPU, depending on demand. The users are configured as shown in Figure 8. More specifically:

• **User 1 (Continuous)** continuously runs 60 parallel jobs with low funding on 30 physical hosts throughout the experiment run (6 hours). The set of hosts is static and separate from the bursty user's hosts.

• **User 2 (Bursty)** sporadically runs 60 highly funded 30 minute jobs every hour on the 30 physical hosts not used by User 1.

• **User 3 (Spot Market)** schedules and runs 30 jobs of 40 minutes each every hour based on spot market prices. The spot market user selects from any of the 60 hosts in the system.

• **User 4 (Predicting)** schedules and runs 30 jobs of 40 minutes each every hour based on the predicted 80th percentile prices using the PG/Cheb predictor consuming continuous statistical feeds from the SC component deployed on each compute node. The predicting user selects from any of the 60 hosts in the system.

All jobs run a CPU benchmark incrementing a counter with a frequency based on the allocated resource share. The value of the counter is our metrics for work completed. Both the spot market and predicting users have the same budget for purchasing resources.

The dynamic behavior of the system is as follows. The continous user's jobs run on the hosts in the left of Figure 8. The bursty user's jobs run on the right. The spot market user selects the host with the lowest spot price, so that the jobs will almost always run on one of the right hosts. On a few rare occasions, all of the right hosts will have a higher spot price than the left hosts. The predicting user selects based on historical prices. These tend to be lower for the left hosts since the bursty user avoids those hosts, so the predicting user's jobs will tend to use the left hosts.

The distribution of work completed by each job submitted by the spot market and predicting users are graphed in the top graph in Figure 9. The distribution for the predicting user is shifted to the right compared to the spot market

Table 3: Prediction Result of 95th Percentile with 90% Upper Prediction Bound. (S is the success rate and B the prediction bound.)

PL	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.93**	0.16	**0.95**	0.57	**0.93**	1.20
Norm	0.62	0.08	0.76	0.29	0.80	0.58
Bench	0.79	0.28	0.72	0.24	0.55	0.17
KTH	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.96**	0.38	**0.95**	0.92	**0.97**	0.91
Norm	0.87	0.22	0.87	0.47	0.76	0.44
Bench	**0.98**	3.25	**0.98**	1.97	**0.97**	1.40
OSC	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.94**	0.40	**0.94**	1.30	**0.94**	1.11
Norm	0.88	0.22	0.81	0.70	0.74	0.51
Bench	0.81	1.63	0.73	0.80	0.63	0.33
SDSC	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.95**	0.26	**0.96**	0.88	**0.95**	0.92
Norm	0.85	0.15	0.78	0.47	0.72	0.43
Bench	0.79	0.96	0.64	0.55	0.41	0.32

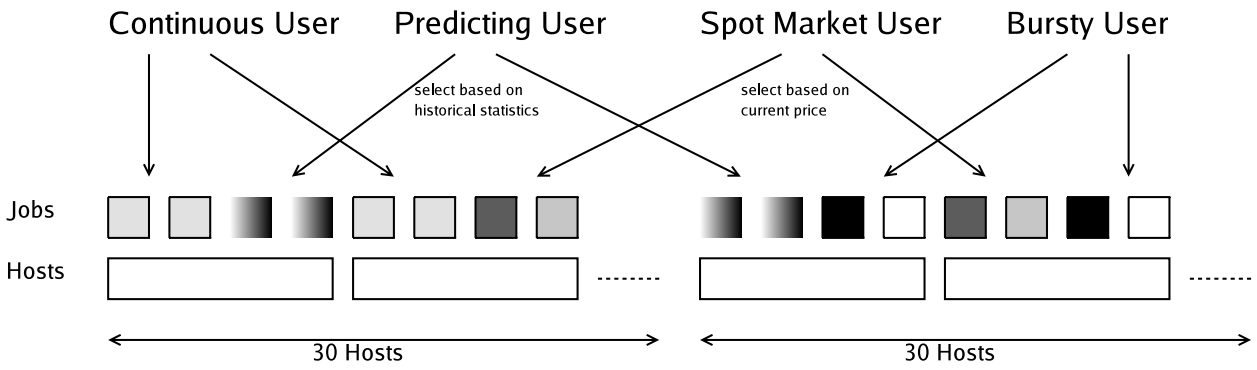


Figure 8: Risk mitigation experimental setup

user, showing that the predicting user finishes more work. The bottom graph shows the cumulative distribution function (CDF) of the work completed by the two users. The area between the two plots shows that fewer predicting jobs were impacted by the bursty user. On average, the jobs of the predicting user performed 22.7% more work. The spot market user's jobs on the far right of the CDF were able (by chance) to run on hosts at times when none of the bursty user's jobs were running. This is why they were able to complete so much work. In contrast, the predicting user almost never selects such hosts because it predicts (accurately in most cases) that the bursty user will soon return. Thus, risk mitigation both increases average performance and reduces variability in performance, even under heavy and spiky load, given that the spiky behavior is predictable over some time horizon (1 hour in this case).

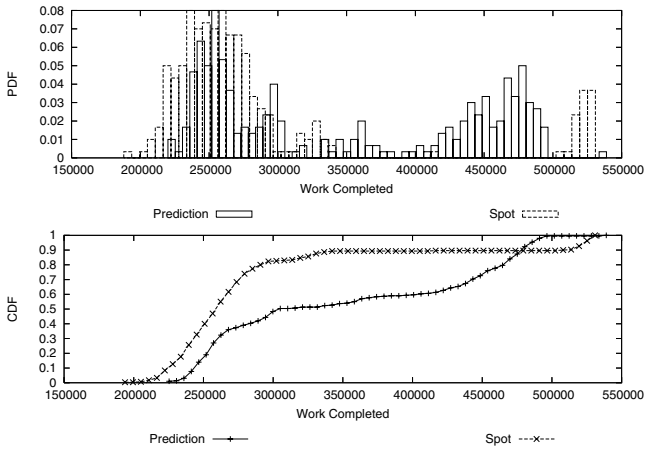


Figure 9: Risk Mitigation using Percentile Predictions

5.4 Prediction Efficiency Experiment

As a final experiment, we evaluate the overhead imposed by the prediction implementation presented in this paper compared to the standard spot market budgeting algorithm used in the previous experiment. The standard algorithm involves the following two steps: (1) get live host spot-market price information, (2) evaluate the bid distribution across the hosts given a total budget to maximize the aggregate performance. The prediction implementation extends step 1 by retrieving the summary statistics required to calculate the prediction bounds, and then extends step 2 by calculating the optimal bid distribution given a desired guarantee level and a given prediction confidence bound level. In the experiment the algorithms were run against a cluster of 70 hosts, using a guarantee level of 95% and a confidence bound of 90%. We interleave 400 runs using the two algorithms and measure the round-trip time of each operation.

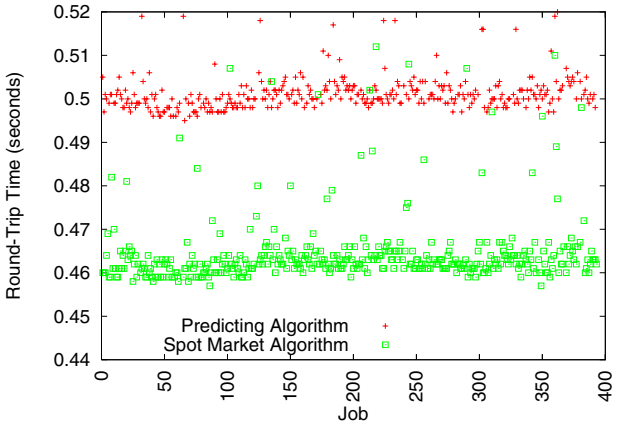


Figure 10: Round-Trip Times of Spot-Market vs Prediction-Based Host Selection and Budgeting

Figure 10 shows the results. The mean round-trip time for spot-market budgeting is .46 seconds and the mean for prediction is .5 seconds. The impact of this overhead on actual running time depends on how frequently the budgeting process is run. In the experiment in Section 5.3, we budget once an hour, so the overhead of the prediction algorithm over the spot market one is $(.5 - .46)/3600 = .001\%$. In other words, the overhead for 70 hosts is negligible, indicating that the algorithm will scale to tens of thousands of hosts.

6. RELATED WORK

MacKie-Mason et. al. [17] investigate how price predictors can improve users' bidding strategies in a market-based re-

source scheduling scenario. They conclude that simple predictors, such as taking the average of the previous round of auctions, improve expected bidder performance. Although the goal of this work is similar to ours, they investigate a different combinatorial allocation scenario where first price winner-takes-it-all auctions are employed, as opposed to the proportional share allocation in our work. Nevertheless, their results are encouraging.

Another use of economic predictions is described by Wellman et. al. [26], where bidding agents use the expected market clearing price in a competitive or Walrasian equilibrium. They employ *tatonnement* which involves determining users' inclination to bid a certain value given a price-level. Wellman et. al. compare their competitive analysis predictor to simple historical averaging and machine learning models. They conclude that strategies that consider both historical data and instance-specific data provide a competitive advantage. The conditional probability of price dynamics given a price-level would be additional useful information in our model. However, this is probably impractical in large-scale systems with users entering and leaving the market at will, and with large real-valued price ranges, so we assume this behavior is incorporated in the price history itself.

Wolski et. al. [28] describe the Network Weather Service (NWS) which has been used in large-scale Grid deployments to monitor compute jobs. Our work differs from NWS in both how statistics are collected and stored and how predictions are computed. NWS uses a multi-service infrastructure to track, store and distribute entire time-series feeds from providers to consumers via sensors and memory components (feed history databases). Our solution only maintains summary statistics and therefore is more light-weight. No persistent storage or searching infrastructure is required. For prediction, NWS uses simple moving average with static parameters. We use more general predictors that can handle any dynamics and adapt their parameters automatically. In addition, the focus in [28] is on predicting queue wait times, whereas we focus on predicting actual demand.

Brevik et. al. [4] present a Binomial Method Batch Predictor (BMBP) complementing NWS [28]. The approach is to assume that each observed value in the time-series can be seen as an independent draw from a Bernoulli trial. The problem is that this does not account for time correlations, which we have found to be substantial in our analysis. Brevik addresses the correlation by first detecting structural changes in the feed when BMBP generates a sequence of bad predictions and thereafter truncating the history which the predictor model is fit against. Our approach is to leverage the correlation by using biased samples of the most recent time intervals, which result in dynamic adaptation of 'structural' changes in the feed. The problem of monitoring and fixing prediction problems a posteriori as in BMBP is that the detection mechanism is somewhat arbitrary and a structural failure of the model could result in great losses, which could defeat the purpose of providing risk mitigating predictions [18].

Our prediction interval calculation was inspired by Haan and Meeker [13] but they also assume that random independent samples are drawn and that a large number of sample data points are used to yield tight prediction bounds. Neither of these two assumptions are true in our scenario. Our calculation of the prediction interval can be seen as more in the spirit of the simple empirical intervals proposed by Williams and Goodman [27]. Their empirical source is the previous sample point, whereas, we use summary statistics as input to the empirical predictions. This allows us to cover larger prediction horizons with greater confidence using fewer data points.

The data analysis of the distribution characteristics in Section 3 was inspired by the work by Mandelbrot on modeling risk in volatile markets [19]. The fat-tail behavior of the hourly volatility (not daily or weekly across all the traces) fits well with the volatility Mandelbrot has seen in the cotton-price, Deutschmark-Dollar exchange rate, and the stock price market dynamics, which he calls 'wild' randomness or chance.

7. CONCLUSIONS

All of the demand traces studied show non-Gaussian properties, which called for more generic distribution-free predictors. The clear correlation between subsequent hourly, daily and weekly time intervals of the traces suggests that the typical IID assumption is not valid and would lead to risk underestimation. This leads us to a model based on dynamically tracking running moments and the most recent snapshots of these moments instantiated by a worst-case bound, Chebyshev inequality influenced distribution estimator. Although this predictor does not generate tight prediction bounds for daily and weekly predictions, it is consistent in the confidence levels promised across all traces investigated, which makes it a good general indicator of the model uncertainty and reliability of the predictions. In highly volatile environments, making point predictions into the future is not possible with any level of accuracy, but high volatility periods are typically clustered. Thus, accurately estimating model uncertainty helps users to decide 1) whether to delay running their jobs until after the 'stormy' period has passed, or if they do decide to run, 2) how much insurance funding to spend.

The Bench predictor shows how poor predictions can be if one only relies on the available past history and ignores time correlations. Similarly, the Norm predictor exemplifies how inaccurate predictions can be if symmetric Gaussian distributions are wrongly assumed. The distribution agnostic Cheb predictor, on the other hand, provides superior predictions in cases where demand was heavily skewed and the volatility spiky, e.g. in the PlanetLab trace.

We have seen that our prediction approach can easily be deployed in scheduling scenarios where the most reliable hosts, delivering a good sustained performance over time, need to be picked for long-running jobs.

This work has focused primarily on helping consumers spend the right amount of money when purchasing resource shares, but the prediction approach with SC/PG/Cheb is general enough to be used by brokers pricing options or reservations as well, which is the focus of future work.

We would also like to study the effects different mixes of prediction, spot-market and reservation usage patterns have on the overall system efficiency and fairness.

8. ACKNOWLEDGMENTS

We thank our colleagues Scott Clearwater, Bernardo Huberman, Li Zhang, Fang Wu, and Ali Ghodsi for enlightening discussions. This work would not have been possible without the funding from the HP/Intel Joint Innovation

Program (JIP), our JIP liason, Rick McGeer, and our collaborators at Intel, Rob Knauerhase and Jeff Sedayao. We are grateful to Vivek Pai at Princeton University for making the PL trace available and helping us interpret it; Travis Earheart and Nancy Wilkins-Diehr at SDSC for making the SDSC trace available; and Lars Malinowsky at PDC for providing the KTH trace.

9. REFERENCES

[1] An Algorithm for Computing the Inverse Normal Cumulative Distribution Function. http://home.online.no/~ pjacklam/notes/invnorm/, 2007.

[2] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, 2007.

[3] M. Bodruzzaman, J. Cadzow, R. Shiavi, A. Kilroy, B. Dawant, and M. Wilkes. Hurst's rescaled-range (r/s) analysis and fractal dimension of electromyographic (emg) signal. In *Proceedings of IEEE Souteastcon '91*, pages 1121–1123, Williamsburg, VA, USA, 1991. IEEE.

[4] J. Brevik, D. Nurmi, and R. Wolski. Predicting bounds on queuing delay for batch-scheduled parallel machines. In *PPoPP '06: Proceedings of the 2006 ACM Principles and Practices of Parallel Programming*, New York, NY, USA, 2006. ACM.

[5] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE, Special Issue on Grid Computing*, 93(3):479–484, March 2005.

[6] Chaki Ng and Philip Buonadonna and Brent N. Chun and Alex C. Snoeren and Amin Vahdat. Addressing Strategic Behavior in a Deployed Microeconomic Resource Allocator. In *Proceedings of the 3rd Workshop on Economics of Peer-to-Peer Systems*, 2005.

[7] S. Clearwater and B. A. Huberman. Swing Options. In *Proceedings of 11th International Conference on Computing in Economics*, June 2005.

[8] S. Clearwater and S. D. Kleban. Heavy-tailed distributions in supercomputer jobs. Technical Report SAND2002-2378C, Sandia National Labs, 2002.

[9] David C. Parkes and Ruggiero Cavallo and Nick Elprin and Adam Juda and Sebastien Lahaie and Benjamin Lubin and Loizos Michael and Jeffrey Shneidman and Hassan Sultan. ICE: An Iterative Combinatorial Exchange. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[10] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[11] M. Feldman, K. Lai, and L. Zhang. A Price-Anticipating Resource Allocation Mechanism for Distributed Shared Clusters. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[12] W. Feller. *An Introduction to Probability Theory and its Applications, volume II*. Wiley Eastern Limited, 1988.

[13] G. J. Hahn and W. Q. Meeker. *Statistical Intervals: A Guide for Practitioners*. John Wiley & Sons, Inc, New York, NY, USA, 1991.

[14] H. Hurst. Long term storage capacity of reservoirs. *Proc. American Society of Civil Engineers*, 76(11), 1950.

[15] L. V. Kale, S. Kumar, M. Potnuru, J. DeSouza, and S. Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society.

[16] K. Lai. Markets are Dead, Long Live Markets. *SIGecom Exchanges*, 5(4):1–10, July 2005.

[17] J. K. MacKie-Mason, A. Osepayshvili, D. M. Reeves, and M. P. Wellman. Price prediction strategies for market-based scheduling. In *ICAPS*, pages 244–252, 2004.

[18] B. Mandelbrot, A. Fisher, and L. Calvet. The multifractal model of asset returns. In *Cowles Foundation Discussion Papers: 1164*. Yale University, 1997.

[19] B. Mandelbrot and R. L. Hudson. *The (Mis)behavior of Markets: A Fractal View of Risk, Ruin, and Reward*. Basic Books, New York, NY, USA, 2004.

[20] L. Peterson, T. Anderson, D. Culler, , and T. Roscoe. Blueprint for Introducing Disruptive Technology into the Internet. In *First Workshop on Hot Topics in Networking*, 2002.

[21] O. Regev and N. Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of 1st International Conference on Information and Computation Economies*, pages 148–157, 1998.

[22] T. Sandholm, K. Lai, J. Andrade, and J. Odeberg. Market-based resource allocation using price prediction in a high performance computing grid for scientific applications. In *HPDC '06: Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, pages 132–143, June 2006. http://hpdc.lri.fr/index.php.

[23] O. Smirnova, P. Erola, T. Ekelöf, M. Ellert, J. Hansen, A. Konsantinov, B. Konya, J. Nielsen, F. Ould-Saada, and A. Wäänänen. The NorduGrid Architecture and Middleware for Scientific Applications. *Lecture Notes in Computer Science*, 267:264–273, 2003.

[24] D. F. Vysochanskij and Y. I. Petunin. Justification of the 3 sigma rule for unimodal distributions. *Theory of Probability and Mathematical Statistics*, 21:25–36, 1980.

[25] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A Distributed Computational Economy. *Software Engineering*, 18(2):103–117, 1992.

[26] M. P. Wellman, D. M. Reeves, K. M. Lochner, and Y. Vorobeychik. Price prediction in a trading agent competition. *J. Artif. Intell. Res. (JAIR)*, 21:19–36, 2004.

[27] W. Williams and M. Goodman. A simple method for the construction of empirical confidence limits for economic forecasts. *Journal of the American Statistical Association*, 66(336):752–754, 1971.

[28] R. Wolski, G. Obertelli, M. Allen, D. Nurmi, and J. Brevik. Predicting Grid Resource Performance On-Line. In *Handbook of Innovative Computing: Models, Enabling Technologies, and Applications*. Springer Verlag, 2005.

[29] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, page 10046.2, Washington, DC, USA, 2001. IEEE Computer Society.

[30] F. Wu, L. Zhang, and B. A. Huberman. Truth-telling Reservations. http://arxiv.org/abs/cs/0508028, 2005.

[31] L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu. Gridis: An incentive-based grid scheduling. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 65.2, Washington, DC, USA, 2005. IEEE Computer Society.

Prediction-Based Enforcement of Performance Contracts

Thomas Sandholm[1] and Kevin Lai[2]

[1] KTH – Royal Institute of Technology
Center for Parallel Computers
SE-100 44 Stockholm, Sweden
sandholm@pdc.kth.se
[2] Hewlett-Packard Laboratories
Information Dynamics Laboratory
Palo Alto, CA 94304, USA
kevin.lai@hp.com

Abstract. Grid computing platforms require automated and distributed resource allocation with controllable quality-of-service (QoS). Market-based allocation provides these features using the complementary abstractions of proportional shares and reservations. This paper analyzes a hybrid resource allocation system using both proportional shares and reservations. We also examine the use of price prediction to provide statistical QoS guarantees and to set admission control prices.

Key words: Admission Control, Proportional Share, Computational Market

1 Introduction

Grid applications traditionally run on dedicated machines, with a fixed performance level that depends on the hardware configuration. In this model, the main source of uncertainty in predicting job deadlines is the queue waiting time. As a solution to heterogeneity, and low resource utilization various virtualized platforms are emerging, such as Xen, VMWare, and VServer. In a virtualized Grid, where the performance level is configured dynamically based on job requirements and current demand, the main source of uncertainty is the risk of not being allocated enough capacity. The allocation decisions are complicated by the scale, and distribution of the Grid resources, and the vast variability and complexity of the job requirements. Therefore, it is not feasible to make these decisions manually using static configurations or policies.

Market-based allocation is one form of allocation that is automated, distributed, and provides QoS. Market-based allocation supports two primary resource abstractions: proportional shares and reservations. A pure proportional share allocator always admits new resource requests and continuously reallocates resource shares in response to the current load. This fully utilizes the resources and always admits well-funded resource requests, but may cause an earlier request to fail a minimum resource requirement. In contrast, a pure reservation allocator fixes resource shares at purchase time. Admitted resource requests in a reservation system will always (modulo failure) meet their

resource requirements, but sometimes utilization is low, and sometimes well-funded requests will be rejected admittance.

In this paper, we examine a hybrid system that mixes both proportional share and reservation abstractions to achieve the best of both worlds: satisfying quality-of-service requirements for some applications while maximizing utilization and providing resource availability for latecomers. Using simulation, we explore how such a hybrid system performs for different workloads.

In addition, we examine how prediction algorithms affect the result. Prediction of future load is critical to efficient resource allocation. Proportional share allocators require it so that purchasers can get statistical QoS guarantees. Reservation allocators require it to set the prices for reservations. However, the effect of universal prediction on a system is not obvious. For example, if low prices are predicted for a particular hour of CPU time, then many resource consumers may try to buy it, thus ruining the accuracy of the prediction.

We base this analysis on previous work on predicting demand in computational markets [1, 2], where we evaluate different prediction techniques to give accurate percentile bounds for expected demand for arbitrary probability distributions. We assume here that we have an approximation for the cumulative distribution function (CDF) of the demand. Furthermore, we assume a computational market where proportional share resource allocations are enforced (e.g., Tycoon [3])

Our contribution in this work is twofold: 1) we highlight and visualize issues with statistical guarantees in performance contracts using simulations, and 2) we propose and implement a solution to these issues using contract admission control.

The paper is organized as follows: Section 2 provides an overview of the mathematical models used to analyze and simulate our resource allocation scenario, Section 3 presents and discusses the design and results of our simulations, Section 4 reviews related work, and finally Section 5 sums up our findings with some concluding remarks.

2 Model

2.1 Statistical Guarantees

We are interested in analyzing what bids individually rational resource consumers should place on their tasks, given that they need a certain performance level to finish within a deadline. Different guarantee-levels can then be compared based on the price consumers have to pay for obtaining a performance level.

To formalize the model we use the following standard probability theory notations:

$$x \in X, P(x) = P(X = x) \tag{1}$$

$$D(x) = \int_{x_{\min}}^{x} P(\varepsilon) d\varepsilon \tag{2}$$

where P is the probability function (a.k.a. PDF), and D the probability distribution function (CDF). To find performance levels based on guarantees it is also useful to look

at the inverse of the distribution function, or percent point function (PPF), defined as:

$$D^{-1}(D(x)) = x \tag{3}$$

The proportional share resource allocation model is defined as:

$$q = \frac{b}{b+c} \tag{4}$$

where q is the performance level or QoS in terms of resource share $(0, 1)$, given a consumer's bid, b, and a measured price, c, of a resource. The price is the sum of all existing bids on the resource.

A rational consumer would hence bid

$$b = \frac{cq}{1-q} \tag{5}$$

for any measured price, c, to maintain a service level q. However, in a competitive computational market the price adjusts dynamically to the resource demand, and can thus be viewed as a random variable C, which changes continuously over time. Since, q depends on c it can also be seen as a random variable, Q. The guarantee of delivering a certain QoS level to the consumer, g, will be expressed in terms of this random variable Q.

$$q \in Q, c \in C \tag{6}$$

$$g = P(\frac{b}{b+C} > q) = P(C < \frac{b}{q} - b) = D_c(\frac{b}{q} - b) \tag{7}$$

where D_c is the price distribution function. Now using the inverse of the price distribution function we can calculate the bids to place given a QoS level and a guarantee

$$D_c^{-1}(g) = b(\frac{1}{q} - 1) \tag{8}$$

which gives

$$b = \frac{D_c^{-1}(g)}{\frac{1}{q} - 1} = \frac{D_c^{-1}(g)q}{1-q} \tag{9}$$

The intuition behind this is that the probability of getting a service level greater than a certain value is the same as the probability of the price being below a particular value, or

$$P(Q > q) = P(C < c) \tag{10}$$

2.2 Admission Control

Now, we would like to offer an admission control service with more than a statistical guarantee for an additional fee. We calculate this new price as:

$$b' = \frac{D_c^{-1}(g+r)q}{1-q} \tag{11}$$

where b' is the price a user needs to pay to get share q with guarantee g, and r is the fee parameter. Note that the fee is not simply added to or multiplied with the bid, but included in the percent point calculation of the price. This ensures that the admission control service is more expensive when there is a high price difference in offering a higher guarantee, in order to account for the expected loss the provider makes when refusing new consumers due to admission control.

In our model, a share of a resource can be requested with either an absolute guarantee paying the admission control fee, or with a statistical guarantee paying the spot (current) market price. The admission controller makes sure that no request is accepted that violates previously admitted requests with absolute guarantees. Whether a violation would occur as a result of admitting a new request is determined by enumerating and evaluating bids and required shares for all active previously admitted requests for the same resource. Consequently, all requests for the resource will need to go through the same admission control path in order to ensure reservation-like guarantees. We note that price volatility in this model is paid for directly by the user, and the admission controller operates in the interest of the provider to keep the prices at a higher level to compensate for not being able to preempt existing low-paying allocations in the event of higher-paying requests. Alternatively, the admission controller could be separated entirely from the resource being provisioned and operate like an insurance agent to put in spot market bids on the resources, and then dynamically update the bids using an insurance fund. For simplicity of evaluation and implementation we chose not to study this more advanced form of admission control here.

If strict admission control is implemented for all users only one guarantee level can be provided. To allow any number of guarantee levels, we strictly enforce only a portion of the allocation request, and make the remaining portion subject to statistical guarantees.

3 Simulations

In our simulations we study the price guarantees and dynamics, using varying levels of statistical and admission control guarantees offering multiple competing consumers service-level guarantees under different work-load situations.

The setup is as follows. A number of concurrent competing consumers submit jobs with inter-arrival-times (IAT) from an exponential distribution and performance requirements drawn from a normal distribution. The performance requirement is obtained from the number of work units that needs to be completed within a given deadline, and it translates to the share, q, of a resource that the consumer will bid for.

To simulate the fact that some users do not care about guarantees, but are only interested in best-effort service we designate a certain proportion of the work-load to be *best-effort* jobs. Those jobs are submitted by calculating the bid a consumer should spend based on the assumption that the price stays at the current mean value. This technically gives the guarantee, $g = 0.5$. All other jobs try to get a guarantee $g \geq 0.6$, and we then measure the guarantees obtained and the price paid under different levels of best-effort jobs. Each run of the simulated workload was configured with a single guaranteed service level, i.e. all jobs competing with best-effort jobs in a simulation

run request the same guarantee level. We then measure and graph the average bid and obtained guarantee for a group of eigth subsequent jobs (based on completion time) requesting a certain guarantee level.

The guarantee obtained in a simulation run is calculated by measuring whether the current share of a job is greater than the required share each second that the job runs. The proportional share allocations are also recalculated each second. We configured the mean of the overall required shares to be higher than the available capacity in order to simulate resource contention and consumer competition.

The general simulation configuration is summarized in Table 1 and Table 2. #C is the number of consumers, #J is the number of jobs per consumer, t the deadline, and BE is the portion of best-effort jobs.

Table 1. General Configuration (All times in seconds)

#C	#J	q	IAT	g	t
4	32	$N(5.5/16, .25)$	$Exp(8)$	$(0.6, 0.9)$	16

Table 2. Individual Simulation Configuration

Simulation	BE	Strategy
I	0.75	statistical guarantee
II	0.25	statistical guarantee
III	0.25	admission control

3.1 Simulation I: 75% Best-Effort with Statistical Guarantees

In the first simulation we look at a work-load with a high portion of best-effort jobs (75%) that can make way for the smaller portions of jobs requiring guarantees. No admission control is used in this simulation, just statistical guarantees. In Figure 1, where each marked point is an average of eight subsequently completing jobs, we see that there is a clear separation left to right and from bottom to top between the different guarantee levels. Jobs with higher guarantee requirements were bidding more (x-axis) and also obtained a higher guarantee (y-axis). This tells us that statistical guarantees worked well when giving consumers their guarantees in this scenario.

We also study the price dynamics. In Figure 2 we can see that the price is stationary although it has a high variance. Note that the first two minutes are not shown because this time is used to bootstrap the simulations. In Figure 3 the variation is high but stable, the skew is positive and varies between 0 and 0.5. A positive skew of the price

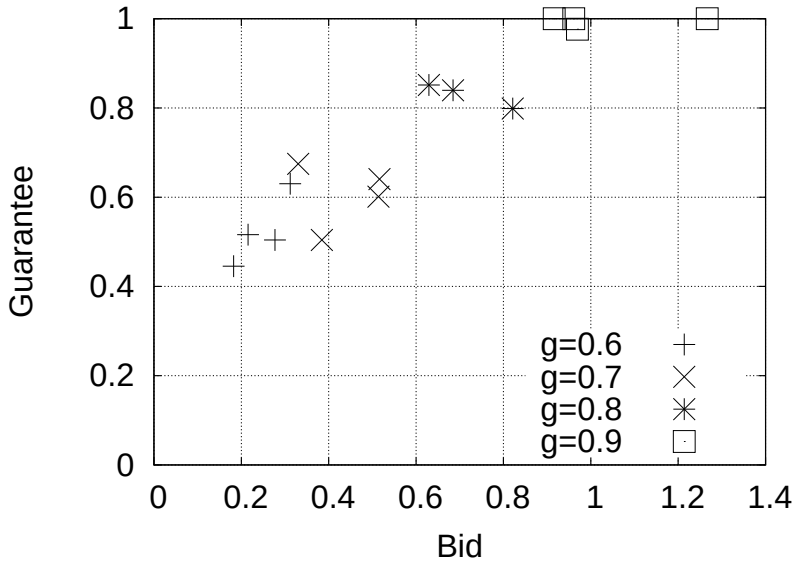


Fig. 1. Bids vs. obtained guarantees for statistical guarantees and 75% Best-Effort jobs

distribution means that more jobs pay a higher price for a guarantee level than would normally (e.g. by Gaussian distribution models) be expected from the mean and the variance. Skewness can thus be viewed as an indication of how risky the computational market is[4].

3.2 Simulation II: 25% Best-Effort with Statistical Guarantees

We now decrease the portion of best-effort jobs to 25% and consequently the portion of jobs requiring guarantees increases to 75%. In Figure 4 we can see that the guarantees obtained for the different guarantee-levels are seemingly randomly layered. The higher bids and requested guarantees do not necessarily yield a higher obtained guarantee as before. This can be explained by the load being too high for the provider to offer everyone the required guarantees.

Looking at the price fluctuations in Figure 5, there is a clear trend of inflation in particular for $g = 0.9$ (bottom right). Also note that simply compensating for the bid based on expected inflation would just accelerate this trend. In Figure 6 we see that both the variance and the skewness of the price distribution exhibit similar behavior as in Simulation I.

3.3 Simulation III: 25% Best-Effort with Admission Control Guarantees

Finally we run a simulation with the same load configuration as in the previous simulation, i.e, 25%, best-effort jobs, but now we offer admission control for all non best-effort

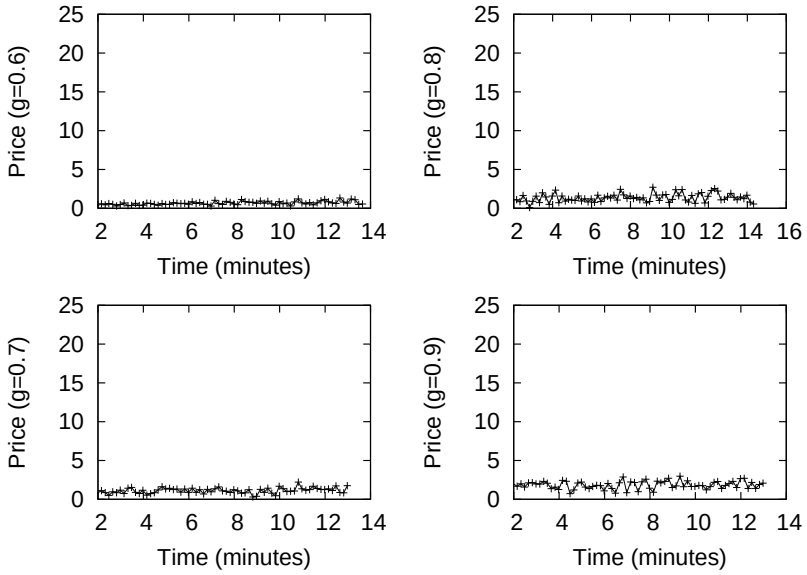


Fig. 2. Price over time for statistical guarantees and 75% Best-Effort jobs

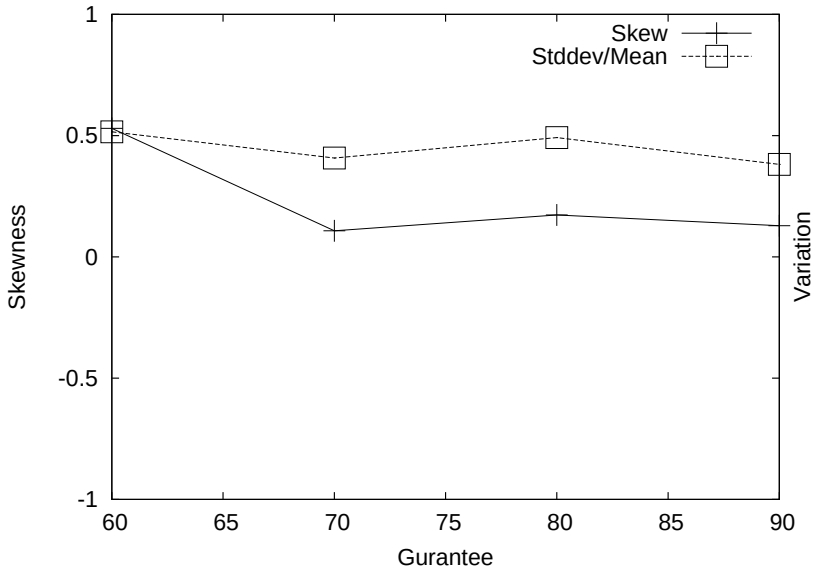


Fig. 3. Price skewness and variation for statistical guarantees and 75% Best-Effort jobs

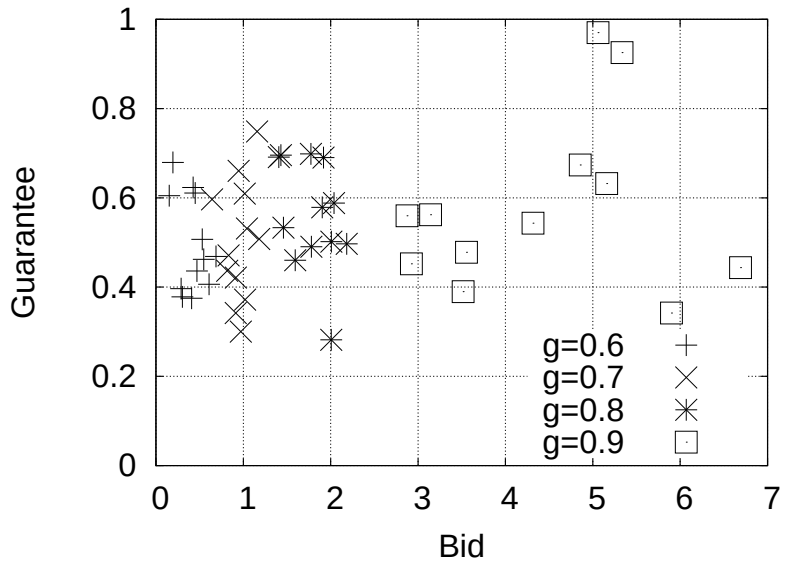


Fig. 4. Bids vs. obtained guarantees for statistical guarantees and 25% Best-Effort jobs

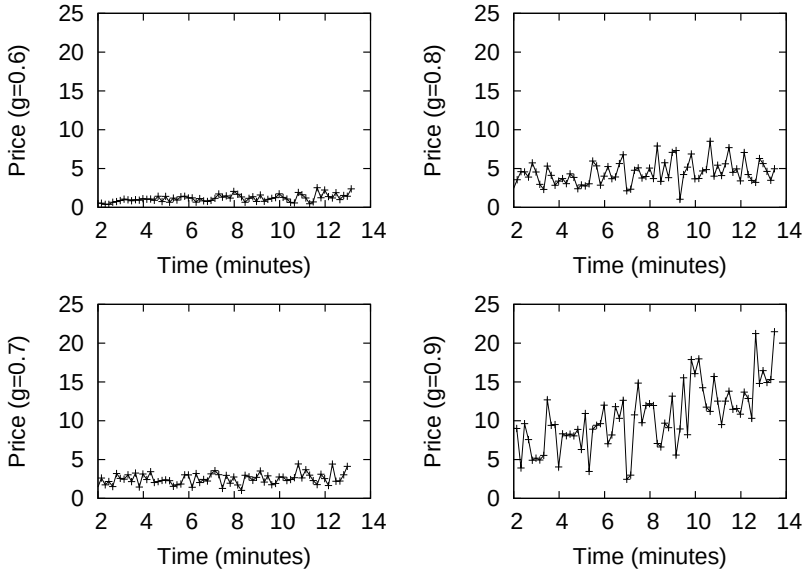


Fig. 5. Price over time for statistical guarantees and 25% Best-Effort jobs

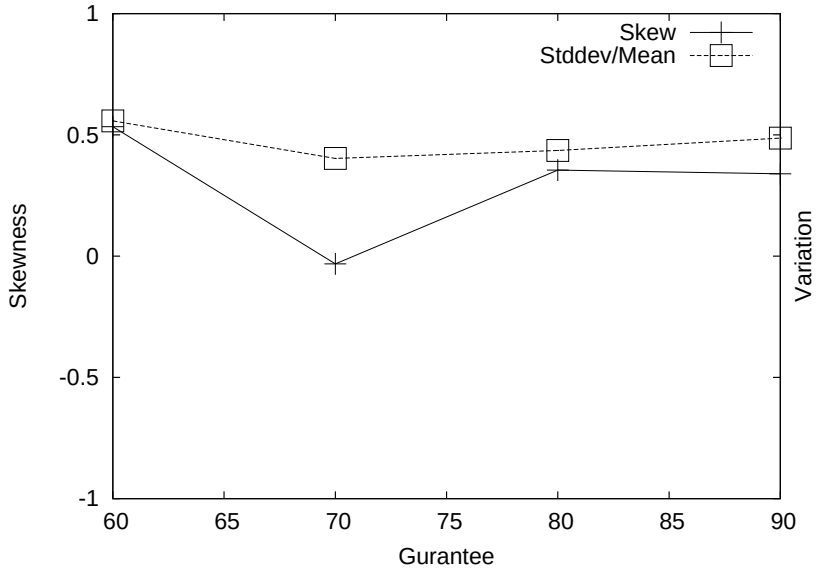


Fig. 6. Price skewness and variation for statistical guarantees and 25% Best-Effort jobs

jobs. An admission control fee of $r = 0.05$ percent points and an enforcement portion of 30% was used. To simulate the important task of an admission control mechanism to allow users to defer their job submissions based on admission results, we defer and resubmit all guarantee jobs that cannot get at least 70% of their work load guaranteed. The time to wait before resubmission is determined randomly with a uniform distribution ranging $1 - 10$ seconds. In Figure 7 it is now again apparent that higher bids also give higher guarantees. Although the separation is not as clear as in Simulation I, it is clearly better than in Simulation II. The separation received is related to the proportion of the job that is strictly enforced. In the case of the entire job being strictly enforced all requested levels result in a 100% guarantee. If the enforcement proportion is made too low, the reulsts will converge to those of Simulation II, that is, requested guarantees cannot be met reliably.

Figure 8 indicates that the inflation is now gone, and Figure 9 shows that the price distribution variation and skewness are similar to the previous two simulations. The penalty for the higher guarantees for some users rests partly on the best-effort jobs and partly on the fact that only a portion (70%) of the entire job run is strictly reserved. We should note that the overall load in this simulation is lower and thus the average bid for the jobs that are let through are obviously lower due to some jobs being refused to run by the admission control. The main point here, though, is that we can add admission control as a compromise between reservations and best-effort allocations in scenarios when statistical guarantees fail.

We summarize the results of the simulations in Table 3. Although the price distribution variation is the same in all the simulations, Simulation II exhibits a higher variance

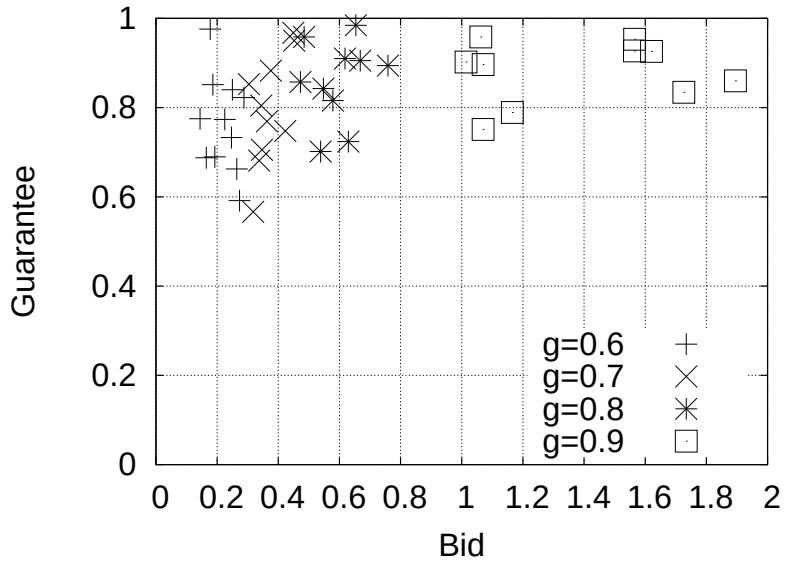


Fig. 7. Bids vs. obtained guarantees for admission control guarantees and 25% Best-Effort jobs

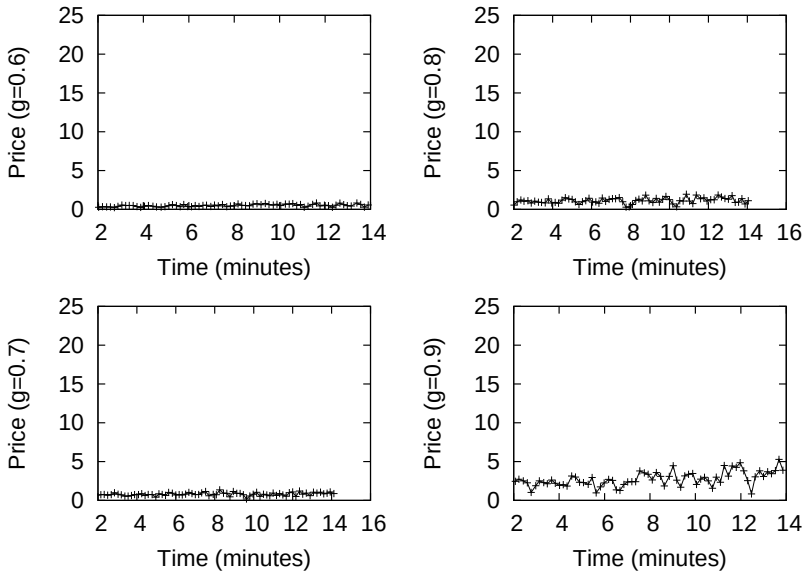


Fig. 8. Price over time for admission control guarantees and 25% Best-Effort jobs

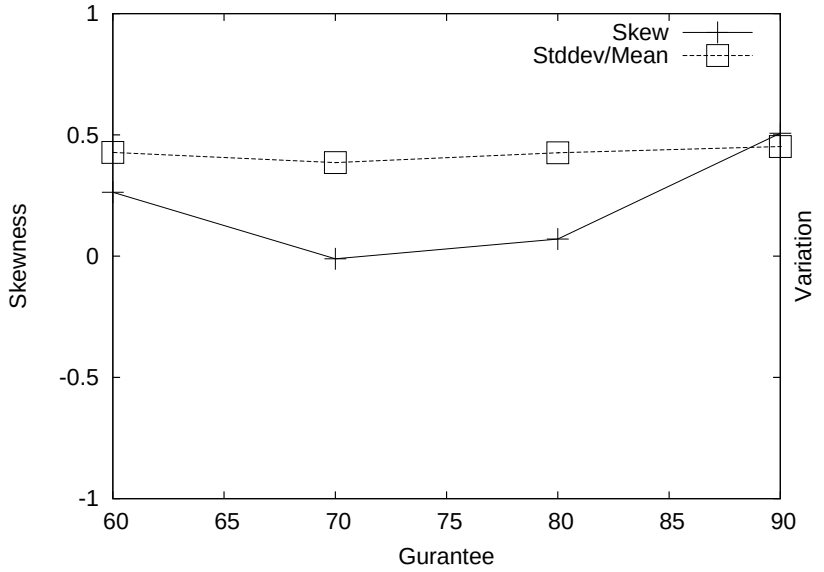


Fig. 9. Price skewness and variation for admission control guarantees and 25% Best-Effort jobs

in guarantee levels delivered, in addition to not being able to deliver on the requested guarantee level.

Table 3. Summary of mean and variation of obtained guarantee levels when requesting 60, 70, 80, and 90% guarantees. All values are in percent.

Simulation	60		70		80		90	
	μ	σ/μ	μ	σ/μ	μ	σ/μ	μ	σ/μ
I	52	15	61	12	83	3	99	1
II	50	22	50	27	56	22	58	34
III	76	14	79	16	86	11	88	8

4 Related Work

There is a substantial body of work on Internet Protocol quality-of-service enforcement, represented by the two IETF specifications IntServ [5], and DiffServ [6]. The IntServ specification takes the approach of reserving paths for individual users, and thus does not scale as well as the DiffServ approach, which is based on marking individual packets with different *per-hop behaviors* in a stateless and decentralized architecture. Wang [7] gives an overview of lessons learned and the pros and cons of the reservation approach which can be implemented with IntServ versus the proportional

share approach which can be built on top of DiffServ. The conclusion was that fixed allocations over a point-to-point path incur too much overhead for most of the web traffic, it is difficult to determine the resource requirements a priori, inter-ISP relationships make end-to-end reservations complicated, and traffic policing breaks down in the event of partial allocation failures. All of these factors result in many IP reservation providers over-provisioning their network capacity, leading to poor utilization. Wang therefore makes a case for a proportional share model [8] where each user receives a proportional share of the currently available bandwidth according to her contribution or spending. We are facing the same issues and trade-offs when allocating computational resources across large distributed systems. However, new virtualization technology and the fact that many of the resources are localized (e.g. CPU, memory, disk) makes it worth revisiting the reservation concepts.

One of the most critical parts of the IntServ architecture is the admission control component, and consequently there has been an extensive effort on designing efficient algorithms for deciding which packets are to be dropped versus served, and how routers and switches should be configured to shape the traffic according to the QoS levels promised to users. Knightly and Shroff provide an evaluation of the different admission control algorithms available for IP traffic shaping in [9]. The dilemma of denying access to flows that might have been served leading to underutilization compared to serving requests that will break existing QoS contracts makes it hard to use coarse statistical bounds and too simplified assumptions about traffic flow distributions. Put differently, both accuracy maximization and risk minimization are desired. The algorithms that accounted for economies of scale and not simply looked at the statistical properties of individual flows were shown to perform much better on average. Again, our admission control decision differs from the IP flow one, in that we can, through virtualization, more directly enforce that an admitted request stays within its bounds. Our decision is thus more about making sure that the provider does not lose out on utilization or profit by admitting low priority tasks prematurely.

MacKie-Mason et. al. [10] investigate how price predictors can improve users' bidding strategies in a market-based resource scheduling scenario. Their conclusion is that even very simple predictors, such as taking the average of the previous round of auctions, help improving expected bidder performance. Another interesting result is that the main reason the predictor strategies outperform memory-less strategies is the fact that the binary decision of whether to participate in an auction can save the bidder more money than accurately estimating exactly how much to bid to obtain a certain performance level. Although, the high-level goal of this work is strikingly similar to ours they investigate a very different allocation and auction scenario, where combinatorial preferences exist and there is a risk of only receiving subsets of the preferred resources. Furthermore, first price winner-takes-it-all auctions are employed, as opposed to proportional share auctions in our work. Nevertheless, their results are encouraging. Another successful use of economic predictions to optimize bidding strategies is described by Wellman et. al. in [11], where bidding agents determine their bids and auctions to enter based on the expected market clearing price in a competitive or Walrasian equilibrium. To find this price they employ the process of *tatonnement* which involves determining users' inclination to bid a certain value given a price-level. Wellman et. al. compare

their competitive analysis predictor to simple historical averaging and machine learning models as employed in the Trading Agent Competition (TAC) and conclude that strategies not only considering background history data but also instance-specific data in the predictions provided a competitive advantage. Finally, their competitive predictor performed on-par with the best machine learning predictor. The conditional probability of price dynamics given a certain price-level would be very useful to collect in our case too to get a full picture of the usage pattern. However, in large-scale systems with users entering and leaving the market at will, and large real-valued price ranges it quickly becomes impractical for our purposes, so we assume this behavior is incorporated in the price history itself.

5 Conclusions

We have studied the effects of bidding for virtualized resource shares using price predictions and admission control. For the predictions to be effective there must either be a sufficiently large portion of best-effort bidders, who can decrease their shares when there is contention, or an admission control mechanism refusing access to requests that would break the existing QoS contracts.

Whether a consumer should spend extra money on getting a higher level of guarantee through an admission control contract, thus depends on the contention among consumers requiring high guarantees. Price history and price distribution analysis serve as good indicators for determining whether this is the case. Conversely, providers would be interested in knowing how to partition their resources between the admission control market versus the best effort market depending on the price fluctuation characteristics and usage pattern.

Future work includes reproducing the simulation results in experiments in a live Grid market deployment (presented in [2]), more in-depth analysis of how providers can dynamically partition their resources for contract markets, and adding more sophisticated option and risk-hedging reservations to the admission control mechanism presented here.

6 Acknowledgments

We thank our colleagues Bernardo Huberman, Li Zhang, Fang Wu, Ali Ghodsi and Scott Clearwater for fruitful discussions. This work would not have been possible without the funding from the HP/Intel Joint Innovation Program (JIP), our JIP liason, Rick McGeer, and our collaborators at Intel, Rob Knauerhase and Jeff Sedayao.

References

1. Sandholm, T., Lai, K.: Evaluating Demand Prediction Techniques for Computational Markets. In: Proceedings of the International Workshop on Grid Economics and Business Models (GECON), Singapore (2006)

2. Sandholm, T., Lai, K., Andrade, J., Odeberg, J.: Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications. In: Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC). (2006)
3. Lai, K., Rasmusson, L., Adar, E., Sorkin, S., Zhang, L., Huberman, B.A.: Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. Technical Report arXiv:cs.DC/0412038, HP Labs, Palo Alto, CA, USA (2004)
4. Mandelbrot, B., Hudson, R.L.: The (Mis)behavior of Markets: A Fractal View of Risk, Ruin, and Reward. Basic Books, New York, NY, USA (2004)
5. Braden, R., Clark, S., Shenker, S.: Integrated services in the internet architecture. RFC 1633, IETF (1994)
6. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services. RFC 2475, IETF (1998)
7. Wang, Z.: A case for proportional fair sharing. In: IWQoS '98: Proceedings of the Sixth International Workshop on Quality of Service, IEEE (1998) 33–35
8. Wang, Z.: Usd: Scalable bandwidth allocation for the internet. In: HPN. (1998) 351–361
9. Knightly, E.W., Shroff, N.: Admission control for statistical qos: Theory and practice. ieeenet **13**(2) (1999) 20–29
10. MacKie-Mason, J.K., Osepayshvili, A., Reeves, D.M., Wellman, M.P.: Price prediction strategies for market-based scheduling. In: ICAPS. (2004) 244–252
11. Wellman, M.P., Reeves, D.M., Lochner, K.M., Vorobeychik, Y.: Price prediction in a trading agent competition. J. Artif. Intell. Res. (JAIR) **21** (2004) 19–36

Autoregressive Time Series Forecasting of Computational Demand

Thomas Sandholm
KTH – Royal Institute of Technology
School of Information and Communication Technology
SE-16440 Kista, Sweden
sandholm@kth.se

Abstract

We study the predictive power of autoregressive moving average models when forecasting demand in two shared computational networks, PlanetLab and Tycoon. Demand in these networks is very volatile, and predictive techniques to plan usage in advance can improve the performance obtained drastically.

Our key finding is that a random walk predictor performs best for one-step-ahead forecasts, whereas ARIMA(1,1,0) and adaptive exponential smoothing models perform better for two and three-step-ahead forecasts. A Monte Carlo bootstrap test is proposed to evaluate the continuous prediction performance of different models with arbitrary confidence and statistical significance levels. Although the prediction results differ between the Tycoon and PlanetLab networks, we observe very similar overall statistical properties, such as volatility dynamics.

1 Introduction

Shared computational resources are gaining popularity as a result of innovations in network connectivity, distributed security, virtualization and standard communication protocols. The vision is to use computational power in the same way as electrical power in the future, i.e. as a utility. The main obstacle for delivering on that vision is reliable and predictable performance. Demand can be very bursty and random, which makes it hard to plan usage to optimize future performance. Forecasting methods for aiding usage planning are therefore of paramount importance for offering reliable service in these networks.

In this paper we study the demand dynamics of two time series from computational markets, PlanetLab[1] and Tycoon[2]. Our main objective is to study the prediction abilities and limitations of time series regression techniques when forecasting averages over different time periods. Here we focus on hourly forecasts that could be applied for scheduling jobs with run times in the order of a few hours, which is a very common scenario in these systems. The main motivation for this study was that an exponential smoothing technique used in previous work [12], was found to perform unreliably in a live deployment.

The general evaluation approach is to model the structure of a small sample of the available time series, and assume the structure is fixed over the sample set. Then perform predictions with regularly updated model parameters and benchmark those predictions against a simple strategy using the current value as the one-step-ahead forecast (assuming a random walk).

We focus our study on the following questions.

- Can a regression model perform better than a strategy assuming a random walk with no correlations in the distant past?
- How much data into the past are needed to perform optimal forecasts?
- How often do we need to update the model parameters?

The answers to these questions depend on both the size of the sliding window used for the forecast and on the length of the forecast horizon. Our goal is to give general guidelines as to how forecasts should be performed in this environment.

When predicting demand in computational networks instantaneous, adaptive, flexible, and light-weight predictors are required to accurately estimate the risk of service degradation and to quickly take preemptive actions. With the increased popularity of virtualized computational markets such as Tycoon, this need for prediction takes a new dimension. Successful forecasts can now reduce the cost of computations more directly and explicitly. However,

[1] http://www.planet-lab.org
[2] http://tycoon.hpl.hp.com

high volatility and non-stationarity of demand complicates model building and reduces prediction reliability.

The main objective of this study is to investigate which time series models can be used when predicting demand in computational markets, and how they compare in terms of predictive accuracy to simpler random walk and exponential smoothing models. Since modeling and parameter estimations need to adapt quickly to regime shifts, a simple fixed static model of the entire series is not likely to produce any good results. In this work we make a compromise and fix the structure of the model but update the parameter estimates continuously.

The contribution of this work is threefold:

- we perform ARIMA modeling and prediction of Tycoon and PlanetLab demand,

  about predictor model performance,

- and we identify common statistical properties of PlanetLab and Tycoon demand.

The paper is structured as follows. In Section 2 our evaluation approach is discussed, and in Section 3 we model and predict the PlanetLab series. In Section 4 we perform the same analysis for the Tycoon series. Then we compare the analyses in Section 5 and discuss related work in Section 6 before concluding in Section 7.

2 Evaluation Method

In this section, we describe the method used to construct models and to evaluate the forecasting performance of models of the time series studied.

2.1 Modeling

We first construct an autoregressive integrated moving average (ARIMA) model of a small sample of the time series in order to determine the general regression structure of the data. The rationale behind this approach is that the majority of the data should be used to evaluate the forecasting performance. During forecasting the model parameters are refit, and to compensate for possible changes in structure we evaluate a number of similar benchmark models. Furthermore, in a real deployment, we ideally want to re-evaluate the regression structure infrequently compared to the number of times the structure can be used for predictions to make it viable. The sample used for determining the regression structure is discarded in the forecasting evaluation to keep the predictions unbiased. Conversely, no measured properties of the time series outside of the sample window are used when building the models of the predictors.

The general model and the benchmark models are then fit to partitions of the data in subsequent time windows. In each time window the model parameters are re-evaluated. The fitted model then produces one, two, and three-step-ahead forecasts. The forecasts are thus conditioned on the assumption of a specific structure of the model. The size of the time windows are made small enough to allow a large number of partitions and thus also independent predictions, and kept big enough for the ARIMA maximum likelihood fits to converge.

2.2 Forecast

The fitted ARIMA model structure is compared to two standard specialized ARIMA processes. The first benchmark model used is the random walk model (RW), ARIMA(0,1,0), which always produces the last observed value as the forecast. The second model is the Exponentially Weighted Moving Average (EWMA), a.k.a. the exponential smoothing model, which can be represented as an ARIMA(0,1,1) or IMA(1,1) process producing forecasts with an exponential decay of contributions from values in the past. This representation is due to Box et al. [2] who showed that the optimal one-step-ahead forecast of the IMA(1,1) model with parameter θ is the same as the exponential smoothing value with factor $\lambda = 1 - \theta$.

For each set of time-window predictions performed, the mean square error (MSE) is computed. To facilitate comparison, the MSEs are normalized against the random walk model as follows

$$\hat{\epsilon} = ln(e_m/e_b) \tag{1}$$

where e_m is the MSE of the model studied, and e_b is the MSE of the benchmark. Thus an $\hat{\epsilon} > 0$ means that the model generated more accurate forecasts than the benchmark. Hence, we have

$$F_{m,b} = Pr(e_m \leq e_b) = \int_{-\infty}^{0} f_{\hat{\epsilon}} \tag{2}$$

where $f_{\hat{\epsilon}}$ is the probability density function (PDF) of $\hat{\epsilon}$. Thus we have constructed a statistic for evaluating the models based on the cumulative distribution function (CDF) of the log ratio of the model and the RW benchmark MSEs, which we call *normalized distribution error* or NDE. This statistic is similar in spirit to the MSE measurement itself, but to avoid a bias towards symmetric error distributions, we base our statistic on the median as opposed to the mean. One might argue that highly incorrect predictions, therefore, are not penalized strongly enough, but we are more interested in the reliability aspect of predictions here, i.e., which model can be trusted to perform better in most cases. If the error distribution has many outliers it should be reflected in the width of the confidence bound instead. We thus focus next on building such unbiased confidence bounds.

2.3 Statistical Test

With the NDE statistic we have a metric to decide when a model performs better than a benchmark, but in order to render claims of statistical significance and prediction confidence bounds, a measure of error variance is needed. Due to a limited set of original data points (one MSE for each sample window size), the approach is to use bootstrap sampling based on the empirical distribution of $\hat{\epsilon}$. Using (2) the null hypothesis is $H_0 : F_{m,b} > .5$, that is, the studied model predicts more accurately than the benchmark in a majority of the cases. The alternative hypothesis H_a is then obviously that the studied model performs worse than the benchmark in a majority of the cases. The bootstrap algorithm is as follows

1. Calculate the $\hat{\epsilon}$ values for the n_w different sample windows
2. Pick n_s samples of size n_w from the $\hat{\epsilon}$ values **with replacement**
3. Calculate the $\alpha/2$ and the $1-\alpha/2$ per cent points from the empirical distribution function of the selected samples, as the lower and upper confidence bounds respectively
4. Reject the null hypothesis and accept the alternative hypothesis if the upper bound is $< .5$, and accept the null hypothesis and reject the alternative hypothesis if the lower bound is $> .5$ at the 100α per cent significance level. If the bound overlaps with $.5$ we say that the model performs *on par* with the benchmark.

R code which implements this test is available in Appendix A. This Monte Carlo bootstrap algorithm is used for two reasons, first to avoid making any assumptions about the distribution of the normalized MSEs in the test, and second to easily map MSE uncertainty to bounds on our NDE statistic. The NDE bound $[lower, upper]$ can be interpreted as there being a $100(1-\alpha)$ per cent likelihood of the model performing better than the random walk model in $100{\cdot}lower$ per cent to $100{\cdot}upper$ per cent of the cases.

In the following sections we apply this evaluation method to the PlanetLab and Tycoon series.

3 PlanetLab Analysis

PlanetLab (PL) is a *planetary-scale*, distributed computing platform comprising approximately 726 machines at 354 sites in 25 countries, all running the same Linux based operating system and PlanetLab software. The user community is predominantly computer science researchers performing large-scale networking algorithm and system experiments. The time series is from December 2005 to December 2006. We calculate demand by aggregating the load value across all hosts and averaging in hourly intervals with a 5-min sample granularity. This load measures the number of processes that are ready to run on machine.

3.1 Model

We select the first month of the trace (707 values out of 8485) as our sample to construct the general ARIMA model. The sample series is shown in Figure 1. There is one big spike in the sample, and we might be tempted to treat it as an outlier, but as seen from the full trace these spikes are quite common and thus need to be accounted for in our model. We instead perform a Box-Cox [1] transform to address non-stationarity in variance. The Box-Cox plot for the sample is shown in Figure 1(c). A λ value of 0.8 is thus used to transform the series prior to the ARIMA analysis. This λ value is somewhere between a $\sqrt{Z_t}$ and a Z_t (no) transform. From Figure 2 we note that the ACF has a slow decline in correlation, and that the PACF is near unit root in lag 1. Now to formally test for unit root we perform the augmented Dickey-Fuller test [5], and obtain a t-statistic of -2.0294 which has an absolute value less than the 5 per cent critical value -3.41, so we cannot reject the null hypothesis of a unit root.

Therefore, we difference the series and then see that the differenced ACF in Figure 2(c), does not exhibit any clearly significant correlations. Hence, we model the series as as an ARIMA(0,1,0) process or random walk. We note that there appears to be small significant seasonal correlations around lags 6,8,10,14 and 16. But we decide to ignore those because of our small sample size, and to keep the predictor simple. To summarize, the entertained model is

$$(1-B)Z_t = a_t \tag{3}$$

where B is the backshift operator and a_t is the residual white noise process. A Box-Ljung test [8] of serial correlations of the residuals of this model gives a χ^2 value of 228.297 and a p-value of $4.974 \cdot 10^{-12}$ for 100 degrees of freedom, so there is still structure unaccounted for. Our tests showed that at least an ARIMA(16,1,0) model was needed before the Box-Ljung test succeeded, which is not practical for our purposes, so we stick to our ARIMA(0,1,0) model. Because this model is one of our standard benchmarks (RW) we also add an ARIMA(1,1,0) model to our evaluation to simplify comparison.

3.2 Forecast

We now compare the MSE of the one-, two- and three-step-ahead forecasts of the RW, EWMA, and ARIMA(1,1,0) models. The time windows used for predictions range from 100 to 150 hours. Each empirical normalized MSE distribution thus has 50 measurements. The

evaluation of the forecasts of the ARIMA(1,1,0), and the exponential smoothing models against the random walk model can be seen in Figure 3. We note that a value less than 0 in the plot means that the model predictor performed better than the random walk predictor. We observe that both the ARIMA(1,1,0) and the exponential smoothing model predictors seem to perform better than the random walk predictor for the two and three-step ahead predictions. We further note that there are more high peaks than deep valleys both for ARIMA(1,1,0) and EWMA, and that the EWMA peaks are lower. This pattern indicates that the RW model is more immune to extreme level shifts, and that EWMA handles these shifts better than ARIMA(1,1,0).

Next, we use the statistical test constructed in the previous section to verify the significance of the differences.

3.3 Statistical Test

Table 1 shows the NDE bound results for the PlanetLab models at significance level 5% where n_s was set to 1000. The random walk row displays the errors in proportion to the true value observed, calculated as

$$\bar{\epsilon} = \frac{1}{T}\sum_{t=1}^{T}\frac{|\hat{y}_t - y_t|}{y_t} \tag{4}$$

where $\hat{y}_t$ is the predicted value at time t and y_t is the actual value; and T is the number of time windows used in the test (50). We see that the errors ranged from 4.07% to 6.98% with the longer horizon forecasts performing worse. From the NDE statistic bounds for the ARIMA(1,1,0) and EWMA rows in Table 1 we can conclude that the ARIMA(1,1,0) model generates predictions on par with the random walk model, for one and two-step-ahead predictions, and better at significance level 5 per cent for three-step-ahead forecasts. The EWMA model performs better for longer forecasts but not at a significant enough level to pass our test. To summarize, the only strong conclusion we can draw from these simulations is that the ARIMA(1,1,0) predictor performed better than a random walk predictor for three-hour ahead forecasts, but in general the RW model selected performs relatively well.

4 Tycoon Analysis

Tycoon is a computational market where resources, such as CPU, disk, memory, and bandwidth can be purchased on demand to construct ad-hoc virtual machines. The price of the resources is in direct proportion to the demand, in that the cost of a resource share is dynamically calculated as the ratio between the bid a user places on the resource and the bids all other users of that resource place. The Tycoon network currently comprises about 70 hosts. Usage is sparse and spiky, and is mostly generated from different test suites that are designed to evaluate the system. A trace was recorded of the aggregated CPU price in hourly intervals with a 10-min granularity during a period of 17 days in July-August 2007.

Table 1. PlanetLab Model NDE Bounds at 5% Significance Level with Random Walk (RW), Exponentional Smoothing (EWMA) and ARIMA(1,1,0) models, using 1,2 and 3-step ahead (SE) Forecasts.

	1 SE	2 SE	3 SE
RW	.0407	.0554	.0698
ARIMA(1,1,0)	$[.353, .627]$	$[.471, .725]$	$[.540, .800]$
EWMA	$[.314, .588]$	$[.373, .647]$	$[.392, .667]$

4.1 Model

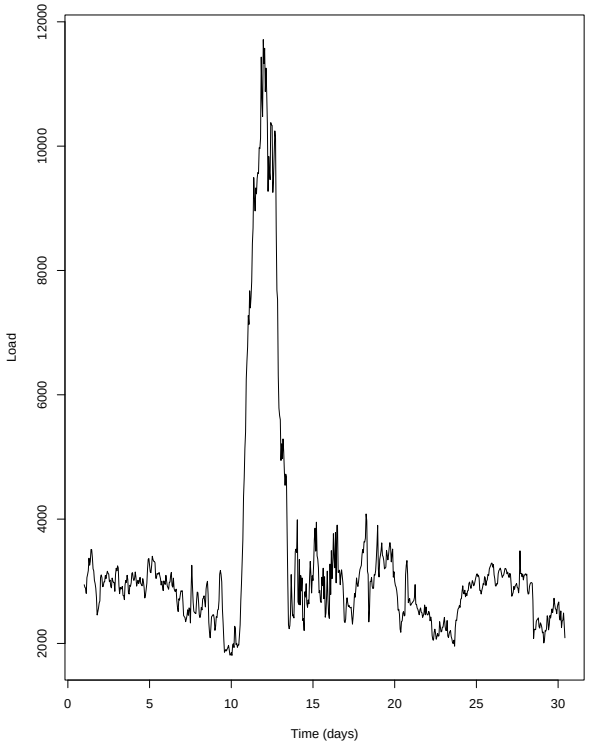
We select the first five days of the trace (119 values out of 404) as our sample to construct the general ARIMA model. The sample and the full series are shown in Figure 4. Due to suspected non-stationarity in variance a Box-Cox transform is again performed. As seen in Figure 4(c), the λ value obtained was -3. We note that the ACF decays slowly and the PACF has a high first lag in Figure 5. So we again difference the series. Now the ACF shows only one significant lag, so we can model it as an IMA(1,1) process. The correlations of the residuals of this model can be seen in Figure 5(d). To summarize, the entertained model is

$$(1 - B)Z_t = (1 - \theta B)a_t \tag{5}$$

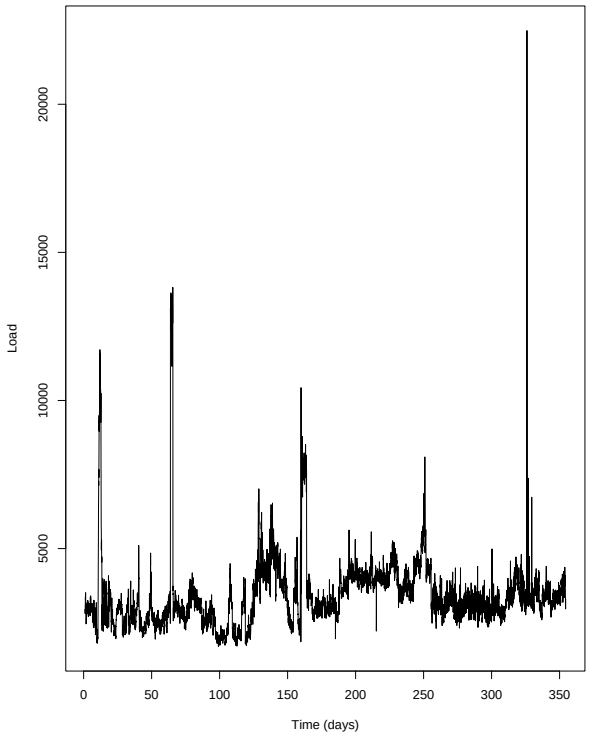
where B is the backshift operator and a_t is the residual white noise process. The θ coefficient was found to be statistically significant at a 5 per cent significance level, and was fit to $.511$. A Box-Ljung test of serial correlations of the residuals of this model gives a χ^2 value of 87.758 and a p-value of $.804$ for 100 degrees of freedom, hence we conclude that the model does not have any serial correlations and is accurate. Because this model is one of our standard benchmarks (EWMA) we also add an ARIMA(1,1,0) model to our evaluation to simplify comparison.

4.2 Forecast

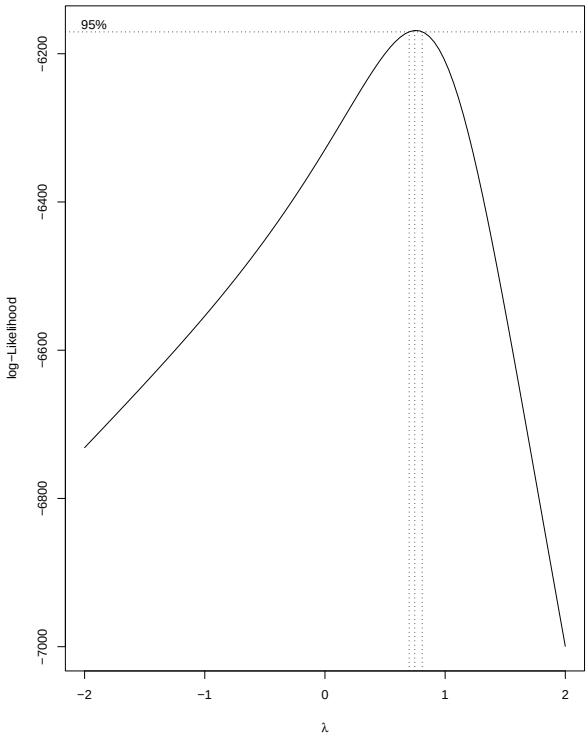
We now compare the MSE of the forecasts of the RW, EWMA, and ARIMA(1,1,0) models. The model parameters are evaluated before each forecast. The time-window used for model fitting ranged from 50 hours to 100 hours into the past, and thus again 50 measurements were generated.



(a) Sample Series

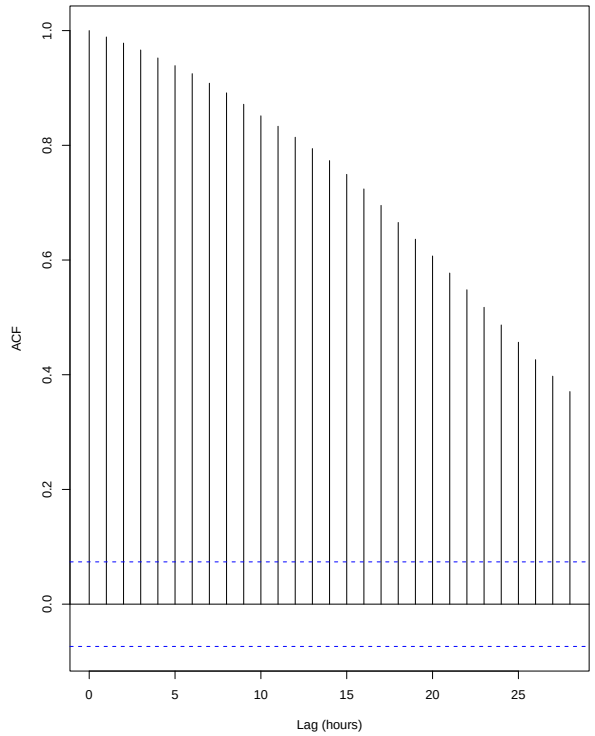


(b) Full Series

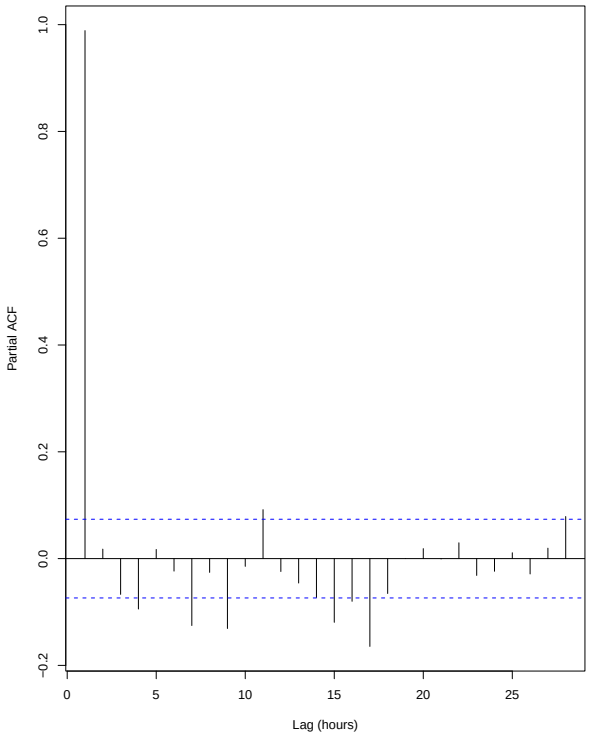


(c) Box-Cox Transform

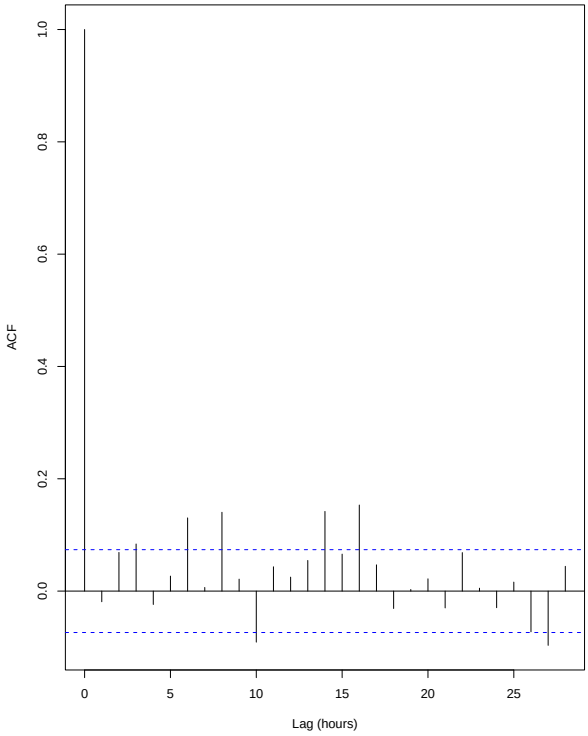
Figure 1. PlanetLab Series



(a) Autocorrelation Function

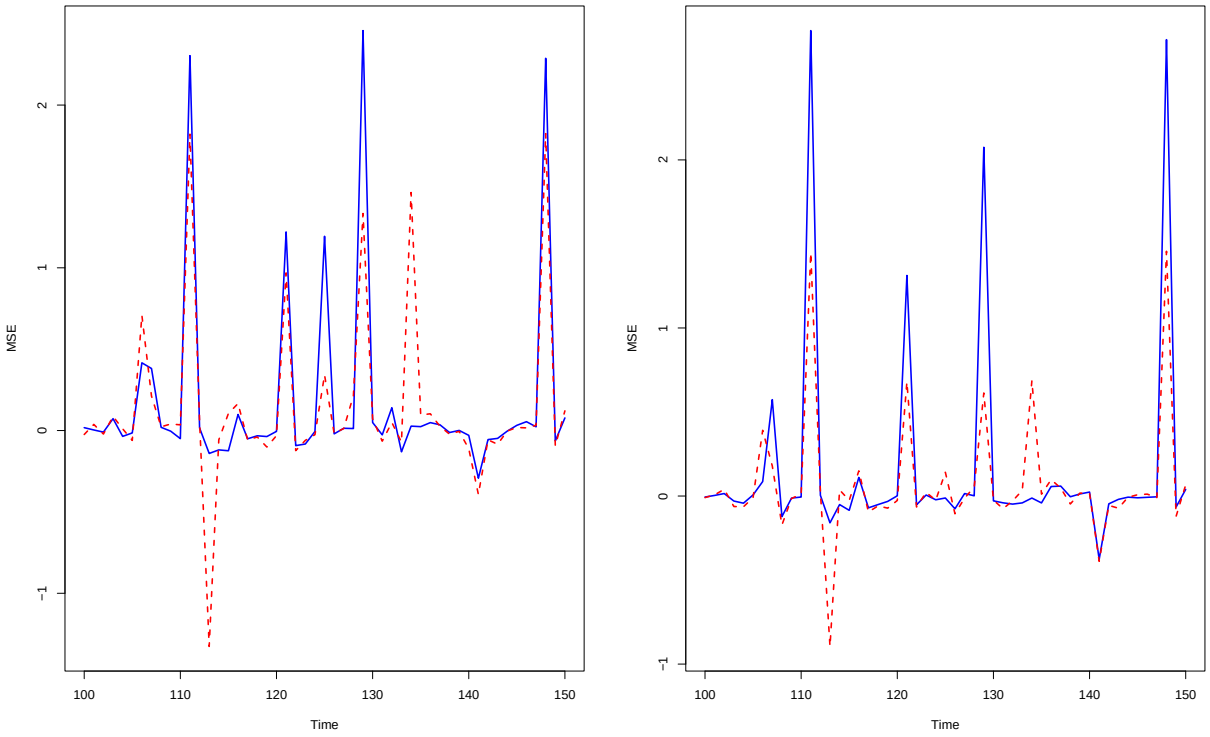


(b) Partial Autocorrelation Function



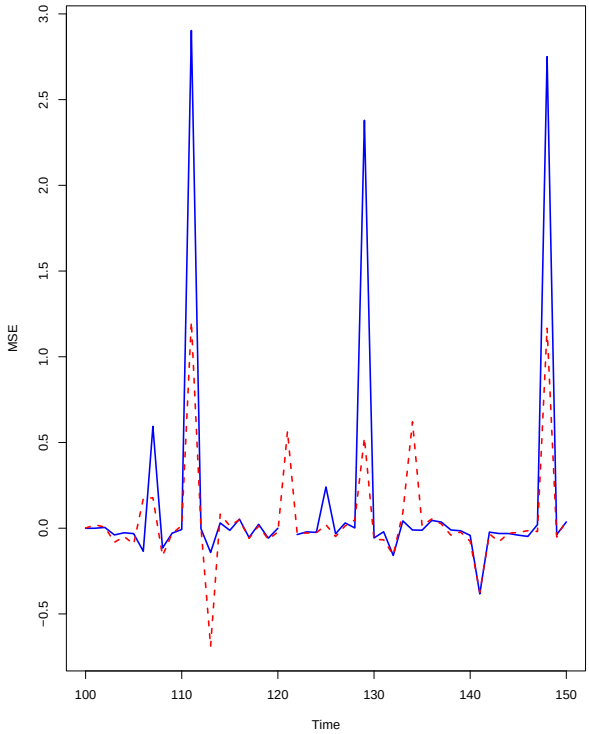
(c) Differenced Autocorrelation Function

Figure 2. PlanetLab Autocorrelation Functions



(a) one-step

(b) two-step



(c) three-step

Figure 3. PlanetLab ARIMA(1,1,0) and Exponential Smoothing (dotted line) vs. Random Walk Model Forecast Errors

Table 2. Tycoon Model NDE Bounds at 5% Significance Level with Random Walk (RW) and Exponentional Smoothing (Exp) Benchmarks, using 1,2 and 3-step ahead Forecasts.

	1 SE	2 SE	3 SE
RW	.144	.199	.243
ARIMA(1,1,0)	[.333, .588]	[.392, .647]	[.431, .706]
EWMA	[.255, .510]	[.353, .627]	[.412, .686]

Table 3. Mean Normalized Quartiles and Range

	Min	Q1	Median	Q3	Max
PlanetLab	.494	.811	.936	1.12	6.55
Tycoon	.452	.763	.860	1.10	5.70

Table 4. Volatility Characteristics

	Coef of Variation	Skewness	Kurtosis
PlanetLab	.362	4.03	28.29
Tycoon	.511	3.67	24.38

The evaluation of the ARIMA(1,1,0), and the exponential smoothing models against the random walk model is shown in Figure 6. We recall that a value less than 0 in the plot means that the model predictor performed better than the random walk predictor. It is not as clear as in the PlanetLab series that RW has fewer extremes of bad predictions. However the ARIMA(1,1,0) model does seem to produce less extreme peaks and valleys than EWMA, i.e. the opposite of what was observed for the PlanetLab data. Due to high volatility it is difficult to draw any conclusions about which model performs best from these plots, so we again have to resort to our statistical test.

4.3 Statistical Test

Table 2 shows the NDE bound results for the Tycoon models at significance level 5 per cent where n_s was set to 1000. We see that the RW model performed much worse for this time series compared to in the PlanetLab series. Average errors range from 14.4 per cent to 24.3 per cent. This apparent difficulty in predicting the series also reflects the results. We see that both the ARIMA(1,1,0) and EWMA models performed on par with RW for all forecasts. So at the 5 per cent significance level no strong conclusions can be drawn about which model performed best. We however note, for the three step-ahead forecasts, that ARIMA(1,1,0) is close to being significantly better than RW, and for one step-ahead forecast, EWMA is close to being significantly worse than RW. The same pattern is apparent here, as in the PlanetLab data; the higher order ARIMA models perform better for longer forecasts.

5 Series Comparison

In this section we compare the dynamics of the PlanetLab series to the Tycoon series using the full traces, and give both quantitative and qualitative explanations to the differences.

Table 3 shows the range and the quartiles of the series, normalized by the series mean. The Tycoon series has a median which is further away from the mean, and the range of values is slightly tighter. The narrower range is expected because of the time horizon difference in the two series. However, overall the statistics for PlanetLab and Tycoon are strikingly similar. This is a bit surprising since Tycoon is just in an early test phase with very limited usage and demand, whereas PlanetLab is a mature system that has been in operation for several years.

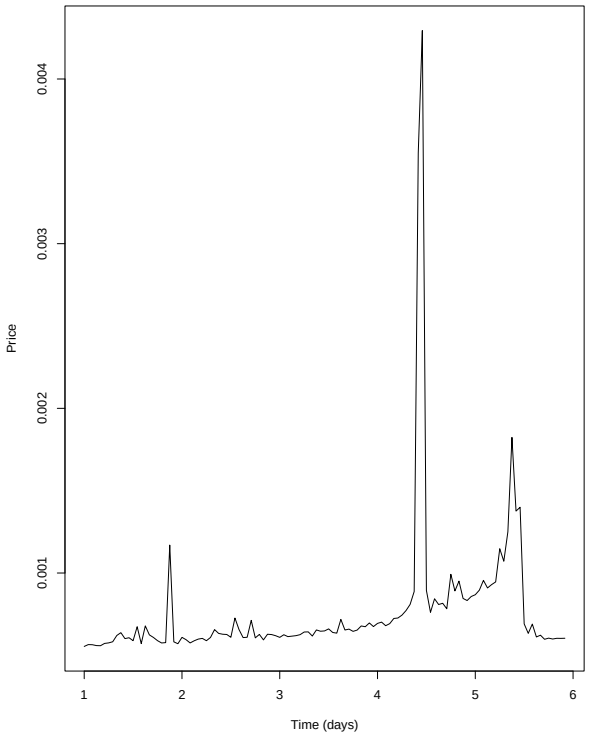
The volatility statistics of the two series are compared in Table 4. We conclude that the variance is higher in Tycoon, but the right tail of the PlanetLab series distribution is longer, and the PlanetLab series is also more prone to outliers. Again it is remarkable how closely the tail and outlier behavior of the much smaller Tycoon sample follows the PlanetLab statistics. To determine whether any of these series exhibit heteroskedasticity, we take the squared residuals from an ARIMA model of the full series and fit an AR model. Then according to Engle [7] heteroskedasticity exists if

$$1 - \chi_s^2((n-s)R^2) < \alpha \tag{6}$$

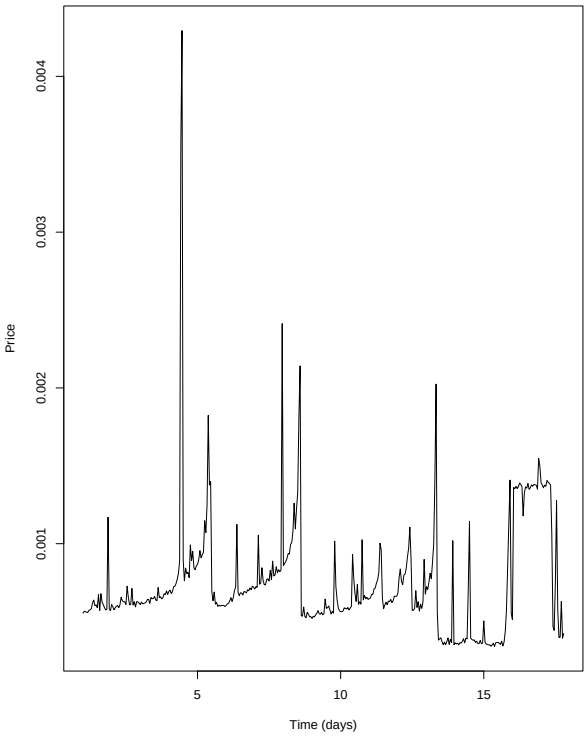
where n is the number of values in the series, s the order of the AR model fit to the squared residuals, χ^2_{df} is the χ^2 density function with df degrees of freedom, and α is the significance level. The complete PlanetLab series follows an ARIMA(3,1,0) model and the complete Tycoon series follows an IMA(1,2) model. The residuals and their squares of these models can be seen in Figure 7.

We find that both the PlanetLab and Tycoon series pass the significance test at the 5 per cent significance level. Furthermore, both the Tycoon and the PlanetLab squared residuals follow AR(3) models, i.e., they have very similar volatility dynamics structure.

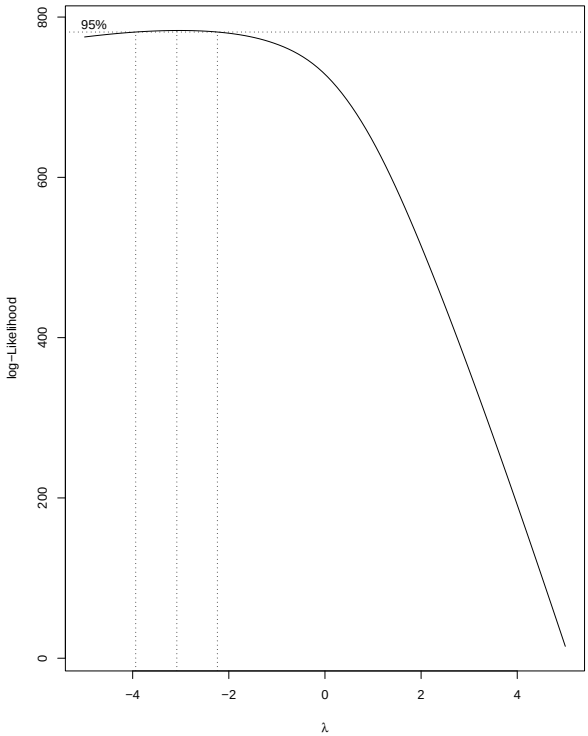
It is easy to see that this heteroskedasticity could cause more outliers and higher kurtosis in a static model. Intuitively, if the first moment fluctuates, the second moment increases, and similarly if the second moment fluctuates there is a greater likelihood of more spikes or AR model



(a) Sample Series

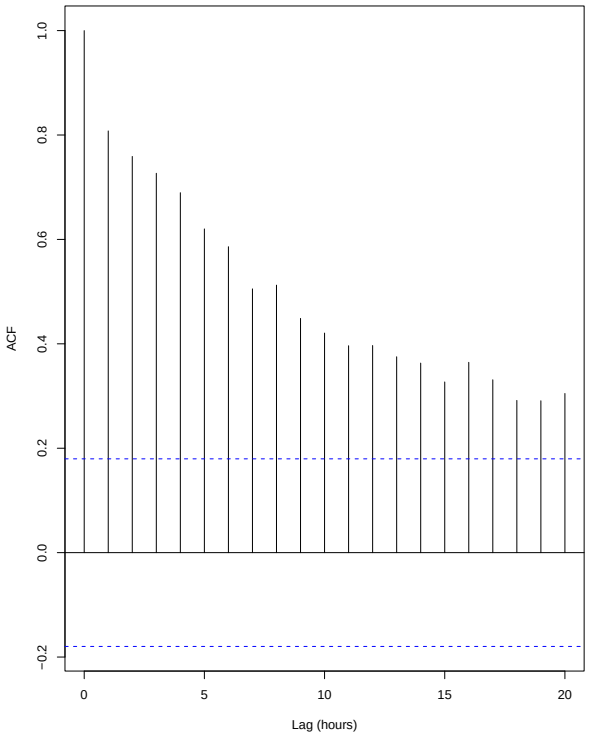


(b) Full Series

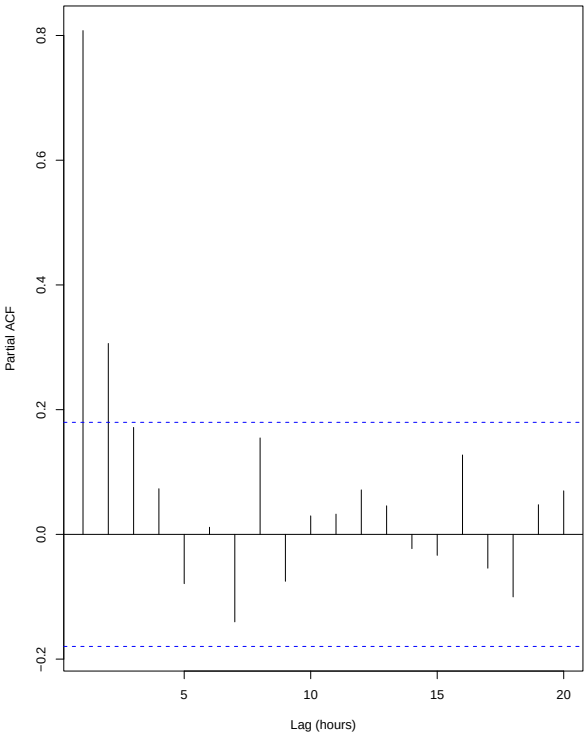


(c) Box-Cox Transform

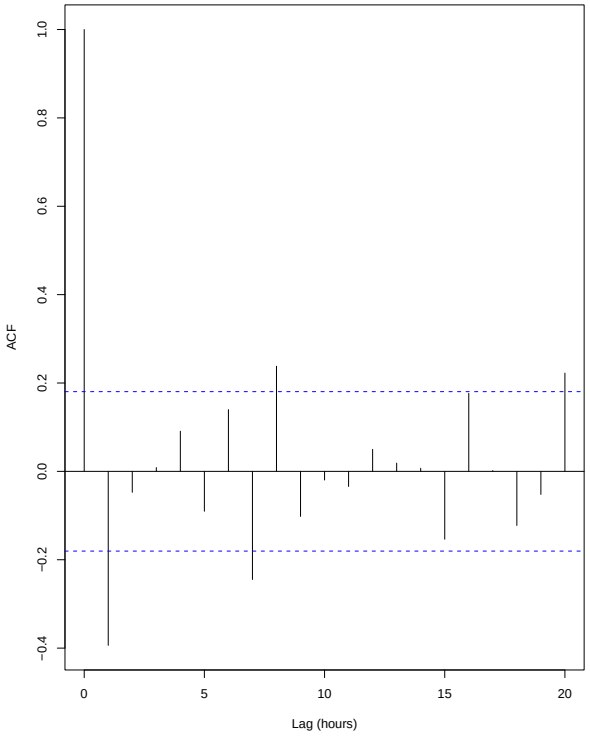
Figure 4. Tycoon Series



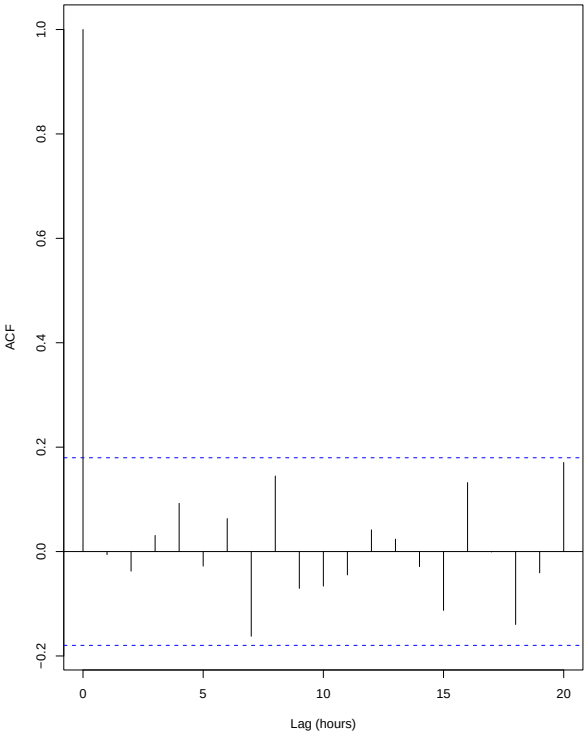
(a) Autocorrelation Function



(b) Partial Autocorrelation Function

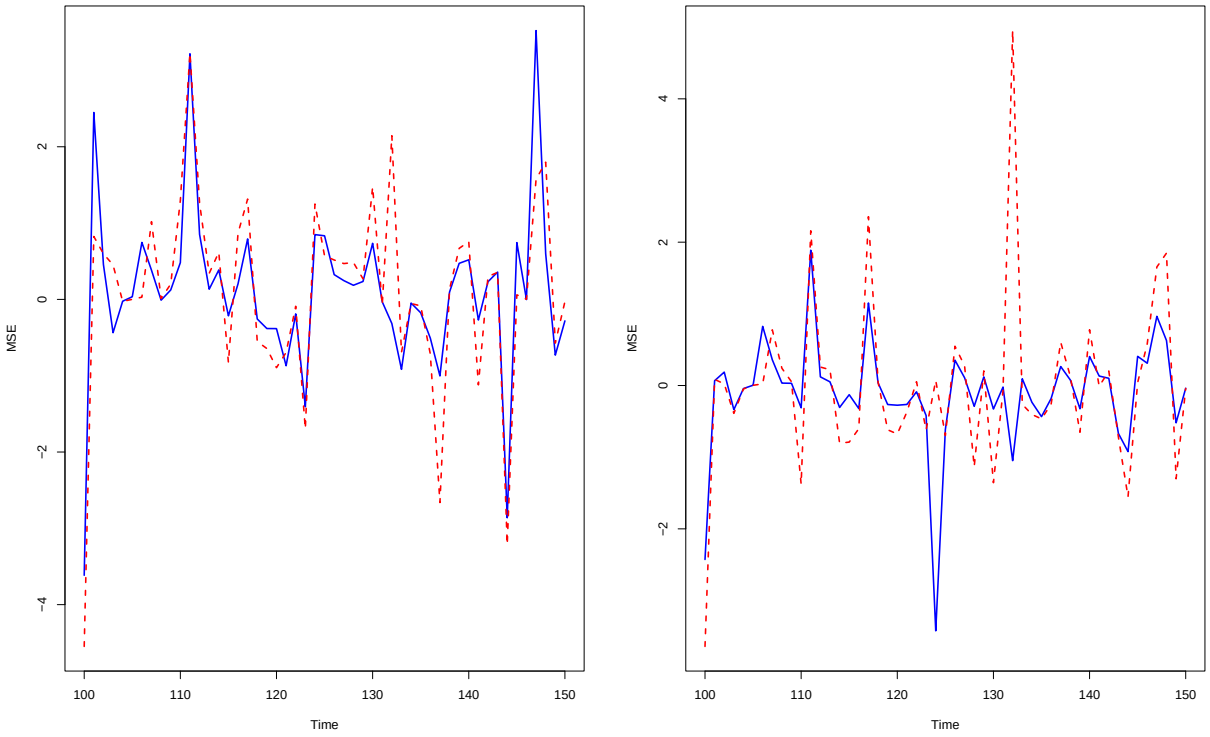


(c) Differenced Autocorrelation Function



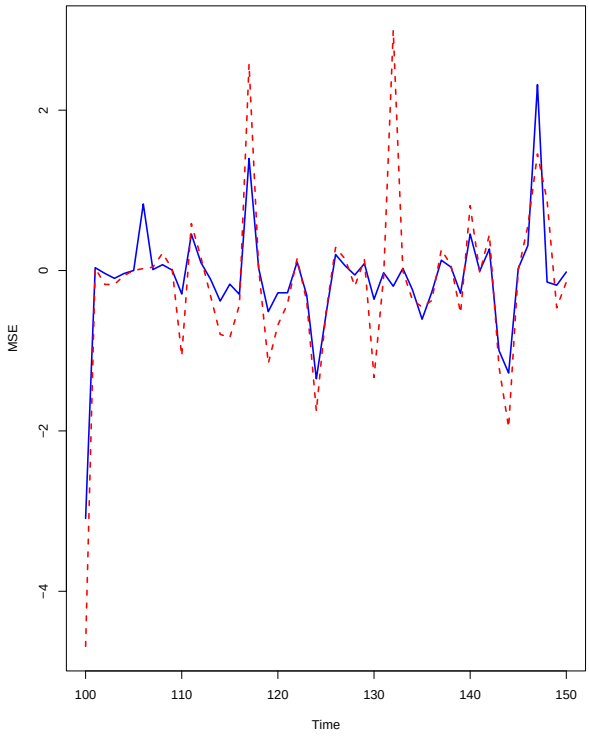
(d) ARIMA(1,1,0) Residuals Autocorrelation Function

Figure 5. Tycoon Autocorrelation Functions



(a) one-step

(b) two-step



(c) three-step

Figure 6. Tycoon IMA and Exponential Smoothing (dotted line) vs. Random Walk Model Forecast Errors

outliers, which would increase the kurtosis. High volatility and dynamics in structure could also explain why ARIMA predictions assuming static volatility and regression structure perform so poorly compared to a simple random walk predictor. However, we note that a random walk predictor does not accurately estimate risk of high demand, which is more apparent for forecasts with a longer future time horizon. An alternative approach to studying volatility and risk over time is the approach of measuring long term memory or dependence. This was done in [11], and we found that non-Gaussian long term dependencies did exist, which could cause so called workload flurries with abnormally high demand.

The ARIMA(1,1,0) model performs better in PlanetLab than in Tycoon, which may indicate that PlanetLab has longer memory of past values than Tycoon. This may be attributed to the shorter sample period and the nature of the applications currently running on Tycoon; mostly short intense test applications.

6 Related Work

The algorithm used for the statistical test of significant differences in predictor performance was inspired by the Monte Carlo bootstrap method introduced by Efron in [6] and popularized by Diaconis and Efron in [4]. The bootstrap method is typically used as a non-parametric approach to making confidence claims. We, use it to expand a short sample into a bigger one without any distributional assumptions about the MSE terms. A more typical usage is to shrink a large sample into multiple smaller random samples that are easier to make statistical claims about collectively.

Tycoon usage has not been statistically investigated before. Previous work on the computational market characteristics of Tycoon has used PlanetLab and other super computing center job traces as a proxy for expected market demand [12] or made simple Gaussian distribition, and Poisson arrival process assumptions [11].

In this work we support the study of PlanetLab as a proxy for Tycoon demand, by verifying a large number of statistical commonalities, both in terms of structure of series and in terms of optimal predictor strategies. Chun and Vahdat [3] have analyzed PlanetLab usage data but not from a predictability viewpoint. Their results include observations of highly bursty and order of magnitude differences in utilization over time, which we also provide evidence for. We note that the PlanetLab trace that Chun and Vahdat studied was from 2003.

Oppenheimer et al. [10] also analyze PlanetLab resource usage and further evaluate usage predictors and conclude that mean reverting processes such as exponential smoothing, median, adaptive median, sliding window average, adaptive average and running average all perform worse than simple random walk predictors and, what they call, *tendency predictors* which assume that the trend in the recent past continues into the near future. They further notice no seasonal correlations over time due to PlanetLab's global deployment. We do see some seasonal correlations in our initial time series analysis but not significant enough to take advantage of in predictions. Further, our evaluation approach follows the traditional ARIMA model evaluation method, and we provide a statistical test to verify and compare prediction efficiency. One major difference between our studies and thus also the conclusions is that Oppenheimer et al. only considered one-step ahead predictions whereas we also consider two, and three-step ahead predictors to do justice to the models considering correlations beyond the last observed step. We finally note that they studied PlanetLab data from August 2004 to January 2005, whereas we studied more recent data from December 2005 to December 2006.

7 Conclusions

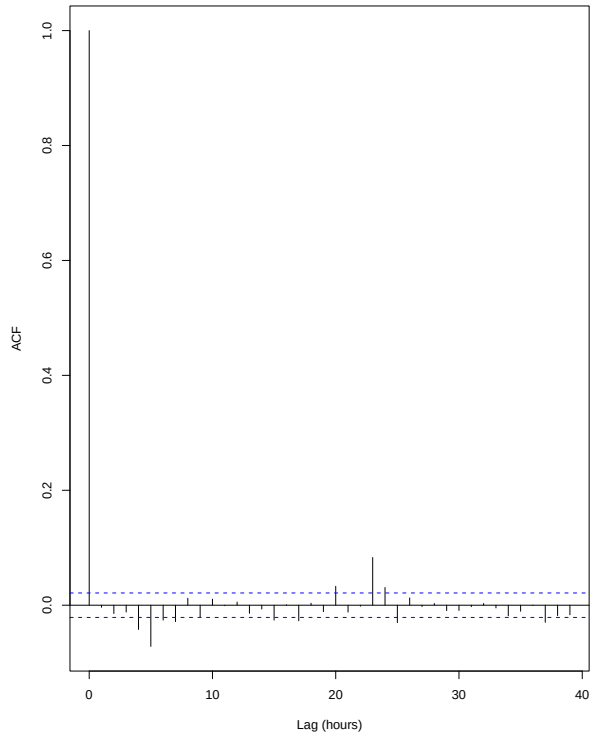
This work set out to study the predictive power of regression models in shared computational networks such as PlanetLab and Tycoon. The main result is that no significant evidence was found that higher order regression models performed better than random walk predictions. The exception was for three-step ahead predictions in PlanetLab where an ARIMA(1,1,0) model outperformed the random walk model.

The study also shows the difficulty in composing a model from a sample and then using this model in predictions if the structure of the series is changing over time as in the Tycoon case.

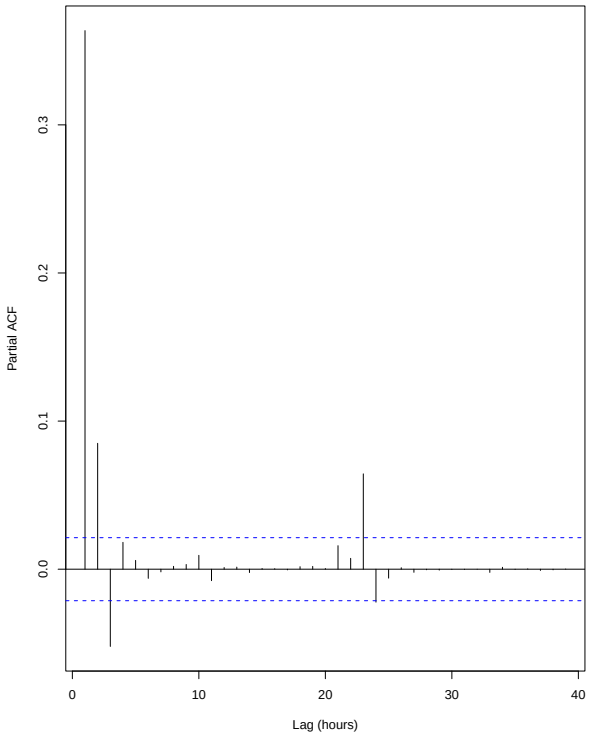
Our study highlighted a number of statistical similarities between Tycoon and PlanetLab, such as volatility structure, outlier likelihood, and heavy right tails of density functions, which motivates further studies and comparisons of workloads to improve forecasting.

The ARIMA models were refitted for every 50 to 150 hours to provide as accurate models of the recent past as possible, but the overall structure of the model was fixed as the one obtained from the fit of the sample series. Larger fitting windows were tested for the PlanetLab data without any effect in the results, but larger windows could not be tested for the Tycoon series due to the limited trace time frame (17 days). There was however a clear pattern that the higher order ARIMA models performed better in the two and three-step ahead forecasts compared to the random walk model.

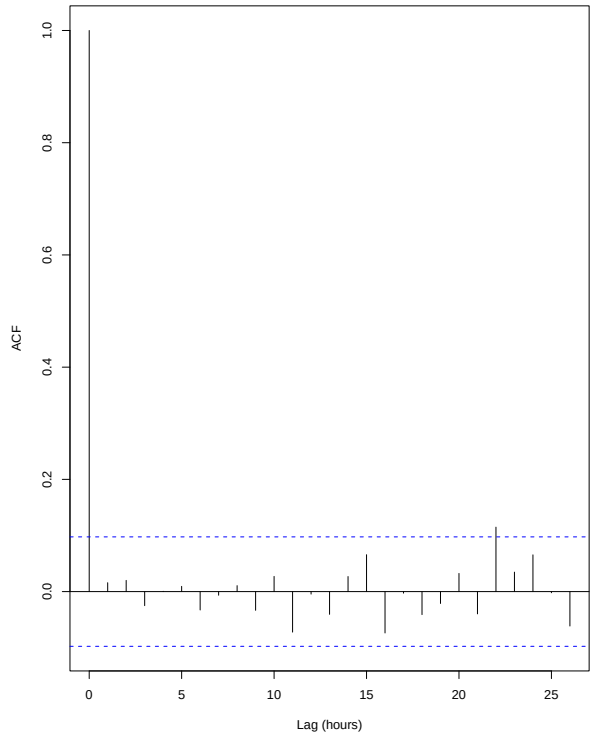
To summarize, we have exemplified the difficulties in modeling significant regressional parameters for computational demand dynamics, even if the model is very generic and the model parameters are re-estimated frequently. It



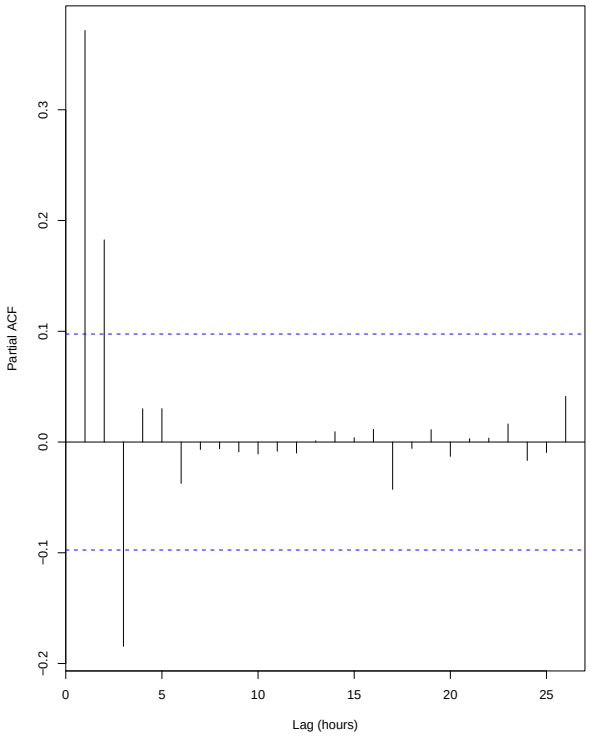
(a) PlanetLab Residual Autocorrelation Function



(b) PlanetLab Squared Residual Partial Autocorrelation Function



(c) Tycoon Residual Autocorrelation Function



(d) Tycoon Squared Residual Partial Autocorrelation Function

Figure 7. PlanetLab and Tycoon Volatility Analysis

was found difficult to improve on the random walk process model for one-step-ahead forecasts, which is a bit surprising (and contradictory to the main hypothesis in [9]) given that RW processes, in theory, should generate a normal distribution of demand whereas the actual measured demand distribution was very right skewed and heavy tailed, both in the PlanetLab and the Tycoon series.

We do however see that higher order regressional parameters can improve the two-step and three-step ahead forecasts. More work is needed to determine how these models should be discovered and dynamically updated. One possible extension is to see if there is an improvement in predictor performance if the model is allowed to changed dynamically as well as the parameters based on observed ACF and PACF behavior. More work is also needed to determine the computational overhead of the more complicated regressional models and the calculations of fits and predictions. Accurate random walk predictors can be built very easily with virtually no overhead, so the improvement in accuracy needs to be significant to be worthwhile. This work does however show that there is a potential for improvement of longer forecasts.

Acknowledgments

I would like to thank Professor Magnus Boman for his detailed comments on earlier versions of this paper; and Professor Raja Velu and Kevin Lai for providing the inspirational ideas underlying this study.

References

[1] G. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society*, B(26):211–252, 1964.

[2] G. Box, G. M. Jenkins, and G. Reinsel. *Time Series Analysis, Forecasting and Control (3rd ed.)*. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[3] B. N. Chun and A. Vahdat. Workload and failure characterization on a large-scale federated testbed. Technical report, Intel Research Berkley Technical Report IRB-TR-03-040, 2003.

[4] P. Diaconis and B. Efron. Computer-intensive methods in statistics. *Scientific American*, (6):116–130, 1983.

[5] D. A. Dickey and W. A. Fuller. Likelihood ratio statistics for autoregressive time series with a unit root. *Econometrica*, (49):1057–1072, 1981.

[6] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[7] R. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50:987–1007, 1982.

[8] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrica*, (65):553–564, 1978.

[9] B. Mandelbrot and R. L. Hudson. *The (Mis)behavior of Markets: A Fractal View of Risk, Ruin, and Reward*. Basic Books, New York, NY, USA, 2004.

[10] D. Oppenheimer, B. Chun, D. Patterson, A. C. Snoeren, and A. Vahdat. Service placements in a shared wide-area platform. In *USENIX'06: Annual Technical USENIX Conference*, 2006.

[11] T. Sandholm and K. Lai. Prediction-based enforcement of performance contracts. In *GECON '07: Proceedings of the 4th International Workshop on Grid Economics and Business Models*, 2007.

[12] T. Sandholm and K. Lai. A statistical approach to risk mitigation in computational markets. In *HPDC '07: Proceedings of the 16th ACM International Symposium on High Performance Distributed Computing*, 2007.

A Bootstrap Test R-Code

```
predict_arima <- function(x,ord, window, horizon, mse, lambda) {
    n = (length(x)-window-2)/window
    errors=c()
    for (i in 0:n) {
        start_index = i * window
        stop_index = start_index + window -1
        outcome_index = start_index + window
        model = arima(boxcox_transform(x[start_index:stop_index],lambda), \
                                  order=ord,method="ML")
        pred = predict(model, n.ahead=horizon)
        if (mse) {
            errors = c(errors, (x[outcome_index+horizon-1] - \
                                  boxcox_inverse(pred$pred[horizon],lambda))^2)
        } else {
            errors = c(errors, abs((x[outcome_index+horizon-1] - \
    boxcox_inverse(pred$pred[horizon],lambda))/x[outcome_index+horizon-1]))
        }
    }
    mean(errors)
}
evaluate_arima <- function(x,ord,from,stop,step,walk,horizon,mse,lambda) {
arima_mse = c()
walk_ind = 1
to = round((stop - from)/step) + from
  for (i in from:to) {
    window = from + ((i-from)*step)
    pred = predict_arima(x,ord,window,horizon,mse,lambda)
    if (length(walk) > 0) {
        pred = pred / walk[walk_ind]
        walk_ind = walk_ind + 1
    }
    arima_mse=c(arima_mse,pred)
  }
  arima_mse
}
bootstrap_test <- function(x,sample_size,alpha) {
  n=length(x)
  x_sample = c()
  for (i in 1:sample_size) {
    x_sample=c(x_sample,ecdf(sample(x,n,replace=T))(0))
  }
  sort_sample = sort(x_sample)
  c( sort_sample[round(sample_size*alpha/2)], \
     sort_sample[round(sample_size*(1-alpha/2))] )
}
evaluate_walk_exp <- function(x,ord,horizons,from,to,step,alpha,samples,lambda)
{
  exp_errors = c()
  arima_errors = c()
  walk_errors = c()
  x_evals = c()
  x_exps = c()
  for (i in 1:horizons) {
    walk_error = evaluate_arima(x,c(0,1,0),from,to,step,c(),i,mse=F,lambda)
    walk_errors = c(walk_errors, mean(walk_error), mean(walk_error))
    x_walk = evaluate_arima(x,c(0,1,0),from,to,step,c(),i,mse=T,lambda)
    x_eval = evaluate_arima(x,ord,from,to,step,x_walk,i,mse=T,lambda)
    x_exp = evaluate_arima(x,c(0,1,1),from,to,step,x_walk,i,mse=T,lambda)
    x_evals = cbind(x_evals, x_eval)
    x_exps = cbind(x_exps, x_exp)
    # Pr(EXP < RW)
    exp_errors = c(exp_errors,bootstrap_test(log(x_exp),samples,alpha))
    # Pr(ARIMA < RW)
    arima_errors = c(arima_errors,bootstrap_test(log(x_eval),samples,alpha))
  }
  errors = cbind(walk_errors,arima_errors,exp_errors)
  attr(errors,'arima') = x_evals
  attr(errors,'exp') = x_exps
  errors
}
```

Admission Control in a Computational Market

Thomas Sandholm
School of Information and Communication Technology
KTH – Royal Institute of Technology
SE-16440 Kista, Sweden
sandholm@kth.se

Kevin Lai
Information Dynamics Laboratory
Hewlett-Packard Laboratories
Palo Alto, CA 94304, USA
kevin.lai@hp.com

Scott Clearwater
P.O. Box 1252
Los Altos, CA 94023, USA
clearway@comcast.net

Abstract

We propose, implement and evaluate three admission models for computational Grids. The models take the expected demand into account and offer a specific performance guarantee. The main issue addressed is how users and providers should make the tradeoff between a best effort (low guarantee) spot market and an admission controlled (high guarantee) reservation market. Using a realistically modeled high performance computing workload and utility models of user preferences, we run experiments highlighting the conditions under which different markets and admission models are efficient. The experimental results show that providers can make large efficiency gains if the admission model is chosen dynamically based on the current load, likewise we show that users have an opportunity to optimize their job performance by carefully picking the right market based on the state of the system, and the characteristics of the application to be run. Finally, we provide simple functional expressions that can guide both users and providers when making decisions about guarantee levels to request or offer.

1. Introduction

In large-scale shared systems, such as cross-organizational Grids, the *best-effort* provisioning model has dominated because of its scalability, high utilization, and high availability. *Elastic* applications that can tolerate variability in performance thrive in this model. Examples include batch-processing applications like data mining, and data warehousing. Other, *inelastic* applications require more stringent *quality-of-service* (QoS) assurances for their users. Examples of these are interactive applications like web applications, and media servers. These applications require an admission control mechanism to prevent high load from violating QoS assurances. However, admission control reduces scalability, reliability, and utilization and introduces the risk of being rejected admittance.

The key difference between best-effort and admission control is that best-effort imposes more costs (in the elasticity requirement) on applications whereas admission control imposes more costs (in scalability, reliability, utilization, and rejections) on the system provider. Nonetheless, for many applications, the cost to implement elasticity is high. As a result, maximizing system efficiency requires both models, but also incentives for applications to use the best-effort model. Economic systems (e.g., [12], [27], [23], [8], [18]) use payment as the incentive. This encourages applications that can implement elasticity to use best-effort while also accommodating those that cannot. In addition, admission control in a priced system provides an opportunity for the provider to do price discrimination and thereby sustain its profitability.

We examine such a hybrid system (based on the Tycoon[18] market-based resource allocator) in this paper. Having two resource models in a system introduces several questions:

What service model should an application use? Most applications are neither purely elastic nor purely inelastic. Instead, a typical application has a specific tolerance for delay. How easily the system can meet delay requirements varies over time as the system load changes. Moreover, an application must balance its delay tolerance with its willingness to pay for performance and the current cost of resources, which constantly changes in a market-based system.

How much of a resource should a service provider allocate to each model? The goal of the service provider is to maximize its profits. It does this by packaging its limited resources in a ratio of best-effort and admission control at prices that are desirable to applications. The ratio and the prices change over time as applications come and go. The provider must also increase the price of admission control resources to include the expected opportunity cost of future rejected admissions.

Previous studies of admission control have been in non-economic systems, been tied to a specific scheduling discipline, and focused on simulations driven by simplified distributional models. Instead, we focus on the admission control problem in an economic system and our work is independent of scheduling. Moreover, we construct and use workload models derived from a HPC production cluster and we evaluate an admission control broker in a live computational market cluster, using all the components of the production system.

Our key contribution is a set of admission control models based on statistical properties of the demand. Admission decisions rely both on expected risk of future rejections, and existing commitments. The risk estimator uses recent price history to calculate a statistical bound (Chebychev upper bound) on the likelihood of a certain deviation from the average demand. The price for a reservation is thus fully dynamic and correlated with the demand, while inelastic applications still obtain high guarantees.

In our experiments, we measure *economic efficiency*, which is the ratio of the utility obtained to the optimal utility (with a theoretical admission model that has full knowledge of future jobs). We show that the economic efficiency of a system that only provides best-effort is 80 per cent at low load, but only 55 per cent at high load. In the high load scenario, inelastic applications have only 35 per cent efficiency, while elastic applications have 90 per cent efficiency. In contrast, an admission control model offering absolute guarantees is 75 per cent efficient, regardless of load and application elasticity. Hence, providers have an incentive to choose and partition the admission model based on resource contention, and users have an incentive to pick an admission model that fits their job preferences. The results presented in this paper aid both the provider and the user in making these decisions dynamically.

The rest of the paper is structured as follows. In Section 2 we describe the workload model used in the experiments. The admission models are presented in Section 3. Section 4 discusses the experiments, Section 5 analyzes the experiment results, Section 6 compares our approach to related research, and finally Section 7 provides concluding remarks.

2. Workload Model

The goal of the workload model is twofold. It should be representative of workloads observed in a shared cluster production system, and it should enable efficient study of parameter spaces in our system. All jobs are allocated and consume real system resources in a full deployment of a computational market. Therefore, the workloads must be short enough to make the results easily reproducible while capturing as much of the statistical traits of the original trace as possible. Pure simulations [1] are easier to evaluate statistically, but our main contribution in this work is the evaluation of a real, distributed deployment.

Instead of using over-simplified assumptions and generalizing behavior across different clusters and sites, we chose to capture individual traces in more precise models. Here we present the results from using the SDSC (San Diego Supercomputer Center) Blue Horizon cluster trace from April 2000 to January 2003, containing 223402 jobs [11].

Distributional models for job inter-arrival time (IAT), runtime (RUN), and number of CPUs used per job (CPU) were constructed directly from the trace. The techniques used to model distributions are very generic and can be applied to a wide range of workload traces. We chose to focus solely on the SDSC trace for three reasons. First, the trace is the longest one available from the parallel workloads archive, and thus gives us the best statistical stability. Second, our main interest is in studying the parameter space of different admission policies under varying load, and hence utilizing a single workload model simplifies the matrix of configurations to be investigated. Third, the trace showcases characteristics that have been observed in many other HPC traces [11].

To synthesize workloads with the same statistical properties as the original trace, we construct a functional representation of the inverse of the cumulative distribution function (CDF), sometimes called the percent point function (PPF) or the quantile function. Many distributions, such as the Normal or Gaussian distribution do not have simple analytical representations of the PPF, which also effects our choice of model. To generate the desired workload the PPF is applied to a series of uniformly distributed random variables in the interval $(0, 1)$.

Inter-Arrival Time. The inter-arrival times were found to follow two different regimes. This bimodality was also found by just studying the last $1/10$ of the trace as well as in parallel archive workload traces from KTH and OSC, not presented here. The pattern found was that jobs submitted within about 10 seconds of each other were overrepresented and followed a separate distribution compared to all other jobs. Standard distributional models do not handle multimodality and we therefore had to represent this distribution

[1] see [25] for a similar simulation-based evaluation

with a mixture model. Each distribution in the mixture was fit to the data using a standard maximum-likelihood estimation (MLE) algorithm. The resulting model, which we call hyper log-weibull (HypWeib), is represented as follows:

$$Z = \alpha X + (1 - \alpha)e^Y \quad (1)$$

where Z is the random variable of the IAT distribution, and α was found to be 0.29. Furthermore, if $X \sim Weibull(16.1489, 1.3462)$ and $Y \sim Weibull(5.93123, 4.95969)$ then $Z \sim HypWeib$. The Weibull CDF is easy to invert arithmetically into a PPF, and therefore our HypWeib distribution is also easily representable as a PPF, which can be used to generate a synthetic IAT workload.

Runtime. The runtime data was also found to be bimodal. Jobs with runtimes less than 30 seconds followed a different distribution than the rest. Since these jobs are unlikely to have produced any useful work and constitute less than 7 per cent of the data, we do not represent them in our runtime model. No standard distribution was found with the maximum-likelihood algorithm that approximated the empirical CDF satisfactorily, therefore we used a polynomial linear least-squares fit of the CDF. To simplify the arithmetic inversion into a PPF we used a 2^{nd} order polynomial as follows:

$$CDF(x) = -0.0125x^2 + 0.3018x - 0.8184. \quad (2)$$

We call this fit the PolyLin distribution in the discussion below.

Job Size. From the density function of number of CPUs per job (CPU) it was apparent that values with base 2 were overrepresented. Only about 1 per cent of the jobs did not follow this pattern. We therefore neglect these jobs in our model for simplicity. Again the log base 2 transformed values do not follow a standard CDF well, but since it is a discrete variable and the number of different values with more than 3 per cent density are only six, the distribution which we call Binary Bin (BinBin) can be easily and accurately represented as a histogram as follows:

$$Z = 2^{X+2} \quad (3)$$

where Z is the random variable in the CPU distribution, X is distributed as the histogram density function $p(x_i) = v_i$, where x is the vector $\{1..6\}$ and v is the vector $\{0.508, 0.144, 0.126, 0.137, 0.048, 0.035\}$. Alternatively, this decline in binary exponents can be modeled as a Zipf(1.4), with a slightly worse fit but with one instead of six parameters.

Model Evaluation. To obtain quantitative statistics for how well the models fit the data we utilize the Two-sided Kolmogorov-Smirnov (KS) test. A bootstrap technique [9] is used to mimic the experiment setup of small representative sample runs. We draw 100 random samples from the empirical and our synthesized workloads, before comparison. For each pair of samples we calculate the KS value (max absolute difference between CDFs), and test the null hypothesis that the two samples are drawn from the same distribution at a 5 per cent significance level. We then record the success-rate, where the null hypothesis could not be rejected, and the average KS values for the sample tests as well as a KS test performed on the entire data sets. As a benchmark we also perform the hypothesis test on the trace with itself. It is commonly assumed that the job submissions follow a Poisson process (submissions are spread uniformly and independently over some time interval), which results in the IAT being exponentially distributed. As an additional comparison we therefore also model the IAT with an MLE fitted Exponential (Exp) distribution with mean 378.648. Similarly power-law distributions have been observed for process runtimes so we also fit the runtime data to a Pareto (Par) distribution with location parameter 1.77332, scale parameter 213.171 and threshold parameter 0, again obtained using MLE. Table 1 summarizes the goodness-of-fit results. From these results we can conclude that our three models, HypWeib, PolyLin, and BinBin, all provide accurate fits to the SDSC trace data. The KS test clearly rejects the Exp model as a good representation of the IAT distribution. Furthermore, we calculate the coefficient of variance ($CV = \sigma/\mu$) of the IAT data to be 3.86, which also rules out a Poisson process, which has an expected CV of 1. The Par model is a good fit to the runtime data, but not as good as the PolyLin fit. PolyLin provides a slightly worse fit than HypWeib and BinBin most likely due to the fact that we ignored 7 per cent of the shortest running jobs for simplicity reasons. To summarize, when taking 1000 samples (with replacement) of 100 values each from the synthesized distribution we can get representative values, according to the KS statistic at a 5 per cent significance level, in 87 per cent of the samples or more for all of our models.

Job Value. No information about user-specified job valuations are available in the trace that we study. Therefore, a set of standard distribution models are used. Three distributions, equal importance (Equal), normal (Norm), pareto power-law (Pareto) are modeled. Equal importance distributions occur when all jobs and all users have the same importance. This model is exemplified in the PlanetLab network. A normal distribution is the most common assumption as it can represent all populations produced by aggregating individually independent random variables with finite mean and variance, in accordance with the central limit theorem. Finally, the Pareto distribution was originally used to model the distribution of incomes, and similar power-law relationships have been observed in various large-scale network metrics, such as popularity of web sites.

Correlations. Supercomputer jobs are known to exhibit long-term correlations (or long memory) over time that

Table 1. Goodness-of-fit Results using Kolmogorov-Smirnov tests against the Trace data.

KS Data 1	KS Data 2	KS success-rate	KS complete data	KS sample mean
Trace IAT	Trace IAT	0.948	0	0.1161
HypWeib	Trace IAT	0.897	0.0557	0.1337
Exp	Trace IAT	0.034	0.2406	0.2901
Trace RUN	Trace RUN	0.944	0	0.1174
PolyLin	Trace RUN	0.869	0.0718	0.1371
Par	Trace RUN	0.818	0.1075	0.1565
Trace CPU	Trace CPU	0.989	0	0.0861
BinBin	Trace CPU	0.982	0.0165	0.0880
Zipf	Trace CPU	0.974	0.0644	0.1040

could lead to periods of anomalously high load [16]. A metric often used to quantify the correlation is the Hurst [13] and Mandelbrot [20] R/S statistic or Hurst exponent. A memoryless process (such as white noise) would have a Hurst exponent of 0.5 whereas processes exhibiting long memory would have exponents close to 1. Exponents less than 0.5 indicate *anti-correlation*, i.e. a past trend is likely to be reversed. We measured a Hurst exponent of 0.735 for the IAT series in the SDSC data, indicating moderate long term correlations, which corresponds well to previous work [16]. To adequately represent a given Hurst exponent in synthesized data, a very large number of data points need to be generated, which makes it impractical for our experiments. Instead we run our experiments under a couple of different load and IAT configurations to represent both regular operation periods and high load periods. We also ran some experiments where we induced short-term time correlations in the IAT value series with a simple sort and permute algorithm, and found that the results were largely the same, with the exception that a purely statistical admission model performed slightly better, as expected. We further note that no significant cross-correlations were found between the inter-arrival time, runtime and job size properties. Due to space limitations and the complexity of a thorough treatment, the correlation issue is left out-of scope and as the focus of future work.

3. Admission Models

In this section three different admission models are described, *best effort* (BE), *statistical admission control* (SAC), and *capacity admission control* (CAC). All of these models take a contract request and produce either an accepted or a rejected contract. The contract request has two parts, *service level requirements* and *resource requirements*. The service level requirements contain the budget a consumer is willing to pay for the resources, as well as total number of work units (CPU cycles in our experiment) needed per node, parallelism (number of nodes), deadline and type of contract (BE, SAC or CAC). The resource requirements specify detailed min and max bound preferences for resources such as CPU, memory, disk, and bandwidth. The resource requirements are enforced and continuously evaluated by the market-based resource allocator (Tycoon's Xen virtualization layer), and the service level requirements are enforced at submission time and evaluated at completion time by the admission model implementation. An approved contract contains a list of nodes selected to run the job and their individual funding levels.

Best Effort. In the best effort model the contract is only rejected if the current spot market price is too high to get the resources specified in the resource requirements part of the contract. The service level part is only used to validate the contract *a posteriori*. Existing jobs can be preempted by new arrivals that pay more. Note that preemption here refers to performance or resource share degradation only, not necessarily that the preempted job is stopped. The resource share obtained is:

$$q = \frac{b}{b+c} \tag{4}$$

where b is the spending rate (e.g. \$/s) derived from the budget and the deadline, and c is the current cost or price of the resource defined as: $c = \sum_i b_i$ where b_i is the current spending rate of consumer i.

Statistical Admission Control. In the statistical admission control model the contract is rejected if the budget is less than an estimated future percentile of the price, or if the current spot market price is too high to get the required resources. Existing jobs can be preempted. The resource share obtained with this model will not drop below q_{min} with a statistical guarantee g where

$$q_{min} = \frac{b}{b + F^{-1}(g)} \tag{5}$$

and $F^{-1}(g)$ is an estimate of the inverse of the cumulative price distribution function for a guarantee in the interval $(0..1)$. As an approximation of F^{-1} we use the Chebychev bound for a given mean and standard deviation of the price. More details on this technique can be found in [24].

Capacity Admission Control. The capacity admission control model performs the same statistical check as in the SAC model to ensure that a minimum price which is higher than the spot market price is paid for a capacity controlled contract. If the statistical check succeeds, an additional check is made to ensure that no currently active contracts are violated. A contract violation is detected by checking the resource shares obtained for all running jobs if the new job were to be admitted. If the share is below what is required to process the total number of work units for any job, the contract of that job is violated. Existing jobs can thus not be preempted in this model. The admission test (which must be true for a contract request to be accepted) is:

$$\forall s \in S : \sum_{h}^{n} \frac{b_h(s)}{b_h(s) + b_h(r) + c} \geq q_s \qquad (6)$$

where S is the set of all existing contracts including the requested contract, n is the number of hosts, $b_h(s)$ is the bid on host h in contract s, $b_h(r)$ is the bid on host h in the requested contract r, and q_s is the minimum performance share promised in contract s. Enforcement may be done on a host by host basis or across all hosts in the contract. We chose the latter in our experiments to allow hosts with a higher than promised performance to compensate for slower hosts in the same contract, and thereby cause fewer rejections.

4. Experiments

4.1. Configuration and Setup

Each job runs a CPU intensive benchmark application until the runtime has expired or the job has completed. Each user gets a budget of $100 to split among its jobs, according to relative value, runtime, and size (CPUs). After the job has finished we record the number of work units completed, which is directly proportional to total number of CPU cycles consumed. The SAC and CAC admission controllers both use the 60 per cent Chebychev bound as price rejection threshold (users who are not willing to pay more are rejected).

A work plan generated using the workload model described in Section 2 serves as input to each experiment run. The work plan contains parameters for each job to be run including: job id, IAT, runtime, CPUs, value, model, and user id. The model parameter is best effort (BE), statistical admission control (SAC), or capacity admission control (CAC).

An experiment run comprises a 2 hour trace of 50 jobs executed on a market-based cluster of 15 hosts and 30 CPUs. 10 users submit 5 jobs each with 4-10 CPUs/job, 2-16min runtime/job, and 30s-8min IAT/job. An identical trace is run with the BE, SAC, and CAC admission models. In order to make statistical claims about differences among the admission policies, each experiment run is repeated with five random traces generated with the same workload model configuration. The rationale behind this setup is to capture both aleatory (randomness within a distributional model) and epistemic (variations due to external factors, in our case live system behavior not captured in our model) uncertainty common in risk modeling [21].

The results from five different workload model configurations are presented below: Pareto job value distribution under low load, Pareto job value distribution under high load, equal job values under high load, Normal job value distribution under high load, and finally a random admission benchmark configuration under high load. A single run of all the experiments including epistemic repetitions, thus took about 150 hours (6 days and 6 hours), which explains why we limited the repetitions to five.

The different load levels were obtained by generating traces with minimum/mean IAT $90/153.66$, and $30/92.36$ seconds for low, and high load respectively. The load levels were set to result in moderate and extensive contention for resources. As the moderate contention workload is less sensitive to the job characteristics in general and the job valuation in particular, we only present the results from the Pareto distribution under low load here.

4.2. Metrics

We are mainly interested in the economic efficiency and fairness of the system, but some basic properties of the admission controller must be met. In particular an admission controller which rejects too many requests will cause underutilization of resources. Similarly, an admission controller that does not decrease the number of contract violations is of little value. We therefore compare contract violations, resource utilization and contract rejections as system metrics independent of the economic metrics.

Violations. This metric measures the ratio between the jobs that have been admitted to those that meet their deadlines. *Violations* $= \frac{j_s}{j_a}$, where j_s is the number of jobs that successfully meet their deadlines, and j_a is the number of jobs that are admitted.

Utilization. Utilization is calculated as the average work units processed per second, compared to a theoretical maximum utilization if no resources were idle at any point during the experiment run. *Utilization* $= \frac{w}{w^*}$, where w is the number of work units completed in the experiment, and w^* is the theoretical maximum.

Rejections The rejection metric is calculated as the ratio of jobs in the experiment to jobs being rejected at admission time. *Rejections* $= \frac{j_r}{j_t}$, where j_r is the number of jobs being rejected, and j_t is the total number of jobs.

Demand Correlation. As a measure of global fairness, we consider the correlation of the demand and the price, seen as cost by the user and profit by the provider. Because the *best effort* model does not reject jobs at admission time with our experimental workloads, it is used as as an approximation for demand. *Demand Correlation* $= \rho_{X,Y}$, where $\rho_{X,Y}$ is the Pearson correlation coefficient of the random variables X and Y, X is the time series of 5min averages of the price in the *best effort* run, and Y is the corresponding time series of the admission mechanism being measured (SAC, or CAC).

Efficiency. As an aid to measuring economic efficiency, the utility function quantifies the payoff a user gets from running a job. The first part of the utility function represents users who do not get any payoff unless the job is fully completed. We call this the *inelastic utility*.

$$U_i(w_d, p, v) = \begin{cases} v - p & \text{if } w_d \geq w \\ -p & \text{if } w_d < w \end{cases} \quad (7)$$

where, w_d is the amount of work completed at the deadline, p is the price paid for running the job, v is the valuation of the job (money bid on job), and w is the work requested in the performance contract. The second part of the utility function represents users who get an incremental payoff based on how much work they completed. We call this the *elastic utility*.

$$U_e(w_d, p, v) = min(w_d/w, 1)v - p \quad (8)$$

A weighted linear combination of (7) and (8) is then the final utility function

$$U = \alpha U_i + (1 - \alpha)U_e \quad (9)$$

Taking the sum of all utilities across all jobs, j, gives us a metric of the *social welfare*, which compared to the optimal utility (with off-line knowledge) can be seen as the economic efficiency of the system. *Efficiency* $= \frac{1}{J}\sum_j^J \frac{U(j)}{U^*(j)}$, where U^* is a theoretical optimal utility obtained from an off-line admission control strategy with no charges applied to (9). More elaborate utility functions could have been used, such as time-decaying utility, or any other quasi-linear utility transformation. We did run some experiments with time-decaying utility as well but found that it complicated the model without providing additional qualitative results. Our main point is to show the difference between elastic and inelastic applications with different market models, and we found the above functions to capture this aspect in the simplest and most accurate way.

To determine the reliability of the results, we perform a statistical test based on the mean and the standard deviation of the metrics. Since only 5 repetitions (experiment samples) can be used because of time constraints the standard z-test (e.g $[\mu - 1.96\sigma, \mu + 1.96\sigma]$ as the 5 per cent significance confidence bound) cannot be performed and we have to resort to the pairwise t-student test with sampled standard deviation.

4.3. Results

All the graphs presented in this section show the average across the five experiment repetitions, with one standard deviation marked with error bars. Non-overlapping error bars thus give a visual indication of a significant result.

System Metrics. Figure 1 shows that the BE model violates a large portion of the contracts as a result of heavy load. SAC addresses this problem but still causes a sizeable portion of violations, whereas CAC has virtually no violations (a small portion of violations may still occur because of our choice to base the admission test on the overall performance of a job as opposed to on a per node basis). The Pareto and Normal value distributions cause more violations for both BE and SAC compared to when all jobs are valued equally. CAC on the other hand is less sensitive to value distribution. Contrasting this result with Figure 1(b), we cannot see any clear evidence that the utilization is significantly effected neither by the value distributions, the load, nor the admission model used. However, the variation in utilization is higher with a Pareto value distribution under high load than with other experiment setups. Further, the rejection ratios in Figure 1(c) cannot fully explain the significant difference in contract violations. For example, in the high-load Pareto value distribution experiment 37 per cent of the jobs were rejected with CAC, compared to none for BE, but 61 per cent of the contracts were violated with BE, compared to 1 per cent with CAC. It is also important to point out that both CAC and SAC reject fewer contracts under low load, which proves that the admission decisions adapt to the current demand.

A quantitative summary of the significance of these results is presented in Table 2. It shows a pairwise t-student test at a 5 per cent significant level, and four degrees of freedom (critical t-value 2.1318) with the null hypothesis that the mean value of the metrics (violations and utilization) are the same. A rejection of the null hypothesis (a significant difference exists) is marked in bold.

Based on these results the main conclusion is that SAC and CAC improved contract fulfillment significantly without causing significant underutilization.

Economic Metrics. Turning to the economic metrics of fairness and efficiency, we can see in Figure 1(d) that the demand correlation coefficient is high for both the SAC and CAC models (about $0.6 - 0.8$ for all experiments). This result indicates that the admission control models are not biased towards the consumer nor the provider, in their pricing structure, and can thus be considered globally fair. The effi-

ciency graphs in Figure 2 show the sensitivity to the elastic versus inelastic utility models. Typically a consumer with a completely inelastic ($\alpha = 1$) utility would choose the CAC or possibly the SAC model. We see that the admission models showed no significant difference in efficiency under low load. However, when there is high contention for the resources and there is high variability in the valuation of jobs (Pareto or Norm), the BE model efficiency deteriorates significantly with an increasing portion of inelastic utility. SAC does not suffer as much from efficiency loss as BE, but it has the same decreasing trend, whereas CAC is completely independent of the utility model chosen (As seen by the straight lines in the graphs). We further note that the efficiency of the SAC and BE models do not decline as much when all jobs have equal value, compared to when they have Pareto or Normal value distributions.

The utility function used to calculate efficiency benefits high-valued jobs with low cost. Comparing Figure 2(a) with Figure 2(b), we can see that the CAC model and to some extent also the SAC model maintained a high level of service for high-valued jobs when the system went from low to high resource contention, a typical trait of a good admission control model.

As previously mentioned, lower utilization and fewer contract violations may be explained as an inherent result of more rejections, as opposed to a feature of the admission control model. To study to what extent the behavior of the SAC and CAC models can be explained by this inherent effect, we modify the SAC model to just randomly reject as many requests as in the Pareto high-load CAC scenario (about 37 per cent). If we first compare the CAC model to the random admission control model (RAC) in Figure 3, we see that the contract violations are considerably higher with the random model although the number of rejections are the same. This shows that the CAC model makes better decisions regarding which jobs to reject. The utilization and the demand correlation (market fairness) are however similar. Looking at the efficiency it is clear that the random model, which does not consider the valuation of jobs, performs worse both than the original SAC implementation and CAC, for both elastic and inelastic jobs. We thus conclude that the inherent effect of rejections cannot explain the contract violation or efficiency properties of the admission control models that we have implemented. We note that it is customary to investigate equilibria of the system when making claims about efficiency in economics. In an experiment which mimics real usage, with randomly configured batch jobs entering and leaving the system continuously, it is very hard to observe and reproduce such stable states. As an alternative we derived our claims from statistically stable states using techniques such as the t-student test. In the following section we will complement the statistical verification with an analytical verification of the results.

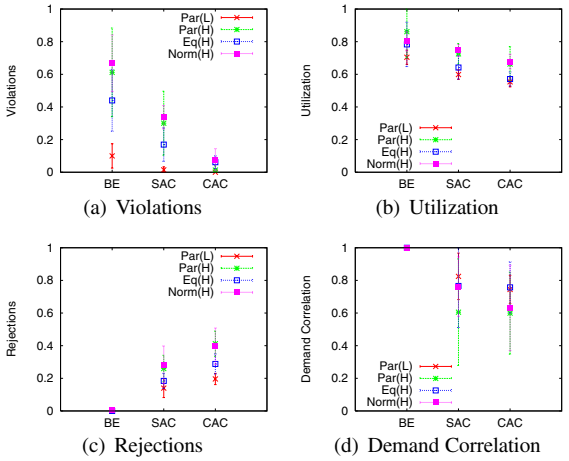


Figure 1. Violations, Utilization, Rejections and Demand Correlation with different Value Distributions (Pareto/Equal/Normal) and Load (High/Low)

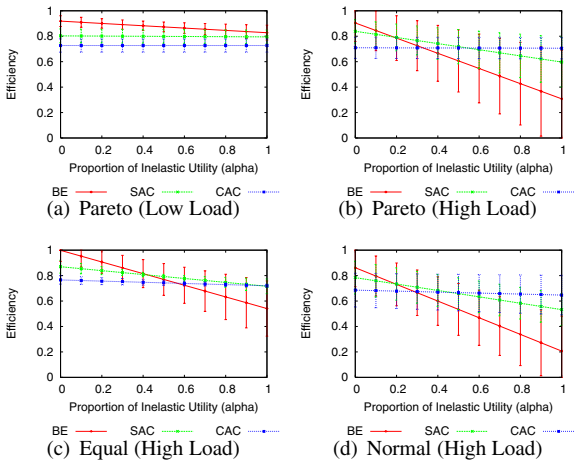
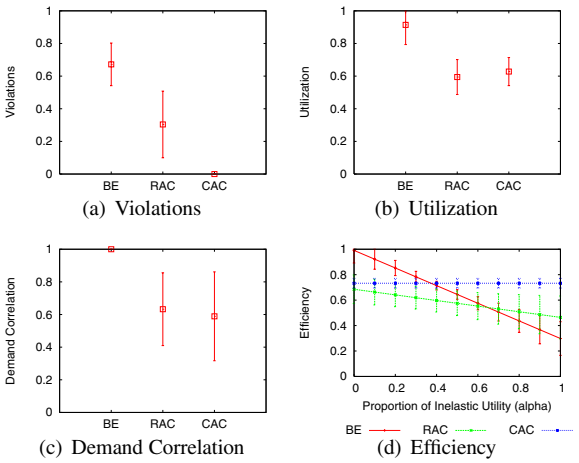


Figure 2. Efficiency with different Value Distributions and Load

Table 2. T-student test (t-values) of mean-value difference for contract violations and utilization. Significant differences (at a 5% level) are marked in bold. Positive values indicate that the mean of the first series compared is higher.

BE/SAC	Par(L)	Par(H)	Equal	Norm
Violations	**2.48**	2.08	**2.82**	**3.94**
Utilization	**4.52**	1.67	2.06	1.79
BE/CAC	Par(L)	Par(H)	Equal	Norm
Violations	**2.99**	**4.90**	**4.39**	**7.09**
Utilization	**6.30**	2.05	**3.33**	**3.74**
SAC/CAC	Par(L)	Par(H)	Equal	Norm
Violations	1.71	**3.25**	**2.20**	**5.91**
Utilization	**2.35**	0.72	1.89	**2.60**



(a) Violations (b) Utilization (c) Demand Correlation (d) Efficiency

Figure 3. Random Admission Control compared to Best Effort and Capacity Admission Control

5. Results Analysis

First, we explain the difference in efficiency between admission control and best effort with inelastic utility analytically. Let $p_w \in [0,1], p_a \in [0,1]$, be the probability of fulfilling the contract, and the probability of admitting a job, respectively. Using the inelastic utility function in (7) we obtain $U = p_a(p_w(v-c) - (1-p_w)c)$. We now study two cases, U_b where all requests are admitted at the expense of some not fulfilling their contracts (e.g. best effort), i.e. $p_a = 1$, and U_g where all jobs fulfill their contract at the expense of some being rejected (e.g. capacity admission control) , i.e. $p_w = 1$. Now by comparing

$$U(p_a = 1) = U_b = p_w(v-c) - c(1-p_w) = p_w v - c \quad (10)$$

and

$$U(p_w = 1) = U_g = p_a(v-c) \quad (11)$$

we get

$$U_g > U_b \Leftrightarrow p_a v - p_a c - p_w v + c > 0 \quad (12)$$

Setting $q = c/v$ we have

$$U_g > U_b \Leftrightarrow p_a > \frac{p_w - q}{(1-q)}. \quad (13)$$

Using the experimental values of the runs from Figure 2 we for instance have for
(a):

$$0.80 < \frac{0.90 - 0.10}{1 - 0.10} \approx 0.89 \Rightarrow U_b > U_g \quad (14)$$

(b):

$$0.63 > \frac{0.39 - 0.14}{1 - 0.14} \approx 0.29 \Rightarrow U_g > U_b \quad (15)$$

(c):

$$0.71 > \frac{0.56 - 0.12}{1 - 0.12} = 0.5 \Rightarrow U_g > U_b \quad (16)$$

and (d):

$$0.60 > \frac{0.33 - 0.10}{1 - 0.10} \approx 0.26 \Rightarrow U_g > U_b. \quad (17)$$

This analytical result explains and mimics the relative positions of the CAC and BE curves in Figure 2 very well for high values of α (inelastic utility). With a similar line of reasoning and setting $p_d = w_d/w$ (work completion ratio), we can show that

$$U_g > U_b \Leftrightarrow p_a > \frac{p_d - q}{(1-q)} \quad (18)$$

when using the elastic utility function in (8).

To summarize the analytical and experimental results, poor work completion (low values of p_w and p_d) and high cost (high values of q) contribute to a higher efficiency of the admission control strategies compared to the best effort strategy if the number of rejections are kept low (high p_a), which corresponds well to intuition.

Based on these results a user with an inelastic job could compare the cost for a resource on a reservation market (SAC or CAC) and on a spot market (BE) to the valuation of the job to get q, and calculate or estimate the likelihood of fulfilling the contract on the spot market (p_w) versus the likelihood of getting rejected on the reservation market (p_a). The two probabilities may be measured by the user, made available by a 3^{rd} party monitoring service or the provider itself. Now if $p_a > \frac{p_w - q}{(1-q)}$ submitting the job to the reservation market is more efficient.

Conversely, based on historical data, the provider could map the current demand, approximated by the measured IAT, to values of p_a and p_w. A provider that would like to optimize the efficiency without any bias towards elastic or inelastic utility users could evaluate: $p_a > \frac{p_w + pd - 2q}{2(1-q)}$. If it evaluates to false, the provider might want to increase the spot-market partition compared to the reservation partition. Whether to use SAC or CAC guarantees largely depends on the level of control the admission broker has over the resource allocator, but also on the frequency of high contention periods (when CAC tends to outperform SAC). Furthermore, the provider could evaluate how much should be charged (q or c/v) for the capacity admission control service given a certain system state (represented by p_w, p_a and p_d).

6. Related Work

Related work fall into three broad categories, each discussed in turn below, market-based resource scheduling and

allocation, IP traffic engineering, and QoS enabled web servers.

Chun and Culler [4] present a performance analysis of three different scheduling algorithms, FirstPrice (priorities paid for on centralized auction market), SJF (short jobs have priority), and PrioFIFO (three priority queues with different prices set statically) based on aggregate user utility. FirstPrice outperforms both SJF and PrioFIFO significantly for highly parallel jobs. PrioFIFO was sensitive to changes in demand and deteriorated in performance if the wait time in the most expensive queue was long. Our work differs from this work in that our results are independent of which scheduling algorithm is used, our workload is constructed by carefully modeling real traces, and our underlying allocation mechanism is a continuously cleared decentralized spot market auction. These differences are also apparent in extensions of Chun's and Culler's work [14, 22, 1] that study more elaborate scheduling algorithms and utility functions that take resource price and provider profit into account. The combination of reservation and spot market pricing with statistical guarantees is novel and sets this work apart from other microeconomic systems that control job performance in shared clusters for parallel jobs, such as [27, 26, 29, 6, 28, 15, 5].

There is a substantial body of work on Internet Protocol quality-of-service enforcement or traffic engineering, represented by the two IETF specifications IntServ [3], and DiffServ [2]. The IntServ specification takes the approach of reserving paths for individual users, and thus does not scale as well as the DiffServ approach, which is based on marking individual packets with different *per-hop behaviors* in a stateless and decentralized architecture. We are facing the same issues and tradeoffs when allocating computational resources across large distributed systems. However, new virtualization technology and the fact that many of the resources are localized (e.g. CPU, memory, disk) makes it worth revisiting the reservation concepts. Knightly and Shroff [17] provide an evaluation of the different admission control algorithms available for IP traffic shaping. The dilemma of choosing between denying access to flows that might have been served and thereby cause underutilization and serving requests that might break existing QoS contracts makes it hard to use coarse statistical bounds and too simplified assumptions about traffic flow distributions. Put differently, both accuracy maximization and risk minimization are desired. Again, our admission control decision differs from the IP flow one, in that we can, through virtualization, more directly enforce that an admitted request stays within its bounds. Our decision is thus more about making sure that the provider does not lose out on utilization or profit by admitting low priority tasks prematurely.

Admission control as a means to avoid service degradation of high priority tasks during overload has also been extensively studied in the context of Web servers, as exemplified in [10, 7, 19]. Priorities of individual requests are either set explicitly in the server configuration or inferred implicitly by the admission algorithm. Our admission controller, on the other hand, gives users an incentive to specify the priority truthfully themselves. Another key difference is that, in a Web server context, the focus is on optimizing throughput and response time by applying queuing and control theory, and estimating expected service time. Our system does not use centralized queues and the service times are not estimated but explicitly requested by the users, as they can vary greatly. In general, our approach of enforcing contracts by means of resource virtualization provides much finer-grained control over service-levels than purely statistical load estimates.

7. Conclusions

Using a statistically accurate, representative and realistic workload model, we have experimentally shown how a simple statistical admission control mechanism can improve contract fulfillment without causing underutilizition during times with high resource contention. Based on two intuitive utility functions, elastic and inelastic, we have also shown that the system remains efficient, i.e. exhibits high social welfare or aggregate utility, even under heavy contention and with various job valuation distributions, such as Equal, Pareto and Normal. The efficiency results were analytically verified and constraints on resource cost, admission ratio, and contract violation ratio were derived to inform which admission policy is best for different utility functions given a certain system state.

Acknowledgments

We would like to thank our colleagues and collaborators, Bernardo Huberman, Tad Hogg, Li Zhang, Lars Rasmusson and John Wilkes for lucid discussions. Thanks also to Cynthia Bailey Lee for clarifying and explaining patterns seen in the SDSC workload trace.

References

[1] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes. Service contracts and aggregate utility functions. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC)*, June 2006.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, IETF, December 1998.

[3] R. Braden, S. Clark, and S. Shenker. Integrated services in the internet architecture. RFC 1633, IETF, June 1994.

[4] Brent N. Chun and David E. Culler. User-centric Performance Analysis of Market-based Cluster Batch Schedulers. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, 2002.

[5] Brent N. Chun and Philip Buonadonna and Alvin AuYoung and Chaki Ng and David C. Parkes and Jeffrey Shneidman and Alex C. Snoeren and Amin Vahdat. Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, 2005.

[6] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. *Software: Practice and Experience (SPE) Journal*, 35(5):491–512, April 2005.

[7] X. Chen, P. Mohapatra, and H. Chen. An admission control scheme for predictable server response time for web accesses. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 545–554, New York, NY, USA, 2001. ACM Press.

[8] David C. Parkes and Ruggiero Cavallo and Nick Elprin and Adam Juda and Sebastien Lahaie and Benjamin Lubin and Loizos Michael and Jeffrey Shneidman and Hassan Sultan. ICE: An Iterative Combinatorial Exchange. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[9] B. Efron and R. J. Tibshirani. *An Introductin to the Bootstrap*. Chapman and Hall, 1993.

[10] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. A method for transparent admission control and request scheduling in e-commerce web sites. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 276–286, New York, NY, USA, 2004. ACM Press.

[11] D. G. Feitelson. Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, 2007.

[12] D. Ferguson, Y. Yemimi, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In *International Conference on Distributed Computer Systems*, pages 491–499, 1988.

[13] H. Hurst. Long term storage capacity of reservoirs. *Proc. American Society of Civil Engineers*, 76(11), 1950.

[14] D. Irwin, J. Chase, and L. Grit. Balancing Risk and Reward in Market-Based Task Scheduling. In *International Symposium on High Performance Distributed Computing*, 2004.

[15] L. V. Kale, S. Kumar, M. Potnuru, J. DeSouza, and S. Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society.

[16] S. D. Kleban and S. H. Clearwater. Quelling queue storms. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 162, Washington, DC, USA, 2003. IEEE Computer Society.

[17] E. W. Knightly and N. Shroff. Admission control for statistical qos: Theory and practice. *ieeenet*, 13(2):20–29, March/April 1999.

[18] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. *Multiagent and Grid Systems*, 1(3):169–182, Aug. 2005.

[19] S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. Admission control and dynamic adaptation for a proportional-delay diffserv-enabled web server. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 172–182, New York, NY, USA, 2002. ACM Press.

[20] B. Mandelbrot and R. L. Hudson. *The (Mis)behavior of Markets: A Fractal View of Risk, Ruin, and Reward*. Basic Books, New York, NY, USA, 2004.

[21] M. E. Paté-Cornell. Uncertainties in risk analysis: Six levels of treatment. *Reliability Engineering and System Safety*, 54(2):95–111, 1996.

[22] F. I. Popovici and J. Wilkes. Profitable services in an uncertain world. In *SC05: Proceedings of Supercomputing*, 2005.

[23] O. Regev and N. Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of 1st International Conference on Information and Computation Economies*, pages 148–157, 1998.

[24] T. Sandholm and K. Lai. A Statistical Approach to Risk Mitigation in Computational Markets. In *Proceedings of the ACM International Symposium on High Performance Distributed Computing (HPDC)*, June 2007.

[25] T. Sandholm and K. Lai. Prediction-based enforcement of performance contracts. In *GECON '07: Proceedings of the 4th International Workshop on Grid Economics and Business Models*, 2007.

[26] I. Stoica, H. Abdel-Wahab, and A. Pothen. A Microeconomic Scheduler for Parallel Computers. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–135, April 1995.

[27] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A Distributed Computational Economy. *Software Engineering*, 18(2):103–117, 1992.

[28] M. P. Wellman, D. M. Reeves, K. M. Lochner, and Y. Vorobeychik. Price prediction in a trading agent competition. *J. Artif. Intell. Res. (JAIR)*, 21:19–36, 2004.

[29] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, page 10046.2, Washington, DC, USA, 2001. IEEE Computer Society.

DEPARTMENT OF COMPUTER AND SYSTEMS SCIENCES
Stockholm University/KTH
www.dsv.su.se/eng/publikationer/index.html
Ph.D. theses:

No 91-004 **Olsson, Jan**
An Architecture for Diagnostic Reasoning Based on Causal Models
No 93-008 **Orci, Terttu**
Temporal Reasoning and Data Bases
No 93-009 **Eriksson, Lars-Henrik**
Finitary Partial Definitions and General Logic
No 93-010 **Johannesson, Paul**
Schema Integration Schema Translation, and Interoperability in Federated Information Systems
No 93-018 **Wangler, Benkt**
Contributions to Functional Requirements Modelling
No 93-019 **Boman, Magnus**
A Logical Specification for Federated Information Systems
No 93-024 **Rayner, Manny**
Abductive Equivalential Translation and its Application to Natural-Language Database Interfacing
No 93-025 **Idestam-Almquist, Peter**
Generalization of Clauses
No 93-026 **Aronsson, Martin**
GCLA: The Design, Use, and Implementation of a Program Development
No 93-029 **Boström, Henrik**
Explanation-Based Transformation of Logic programs
No 94-001 **Samuelsson, Christer**
Fast Natural Language Parsing Using Explanation-Based Learning
No 94-003 **Ekenberg, Love**
Decision Support in Numerically Imprecise Domains
No 94-004 **Kowalski, Stewart**
IT Insecurity: A Multi-disciplinary Inquiry
No 94-007 **Asker, Lars**
Partial Explanations as a Basis for Learning
No 94-009 **Kjellin, Harald**
A Method for Acquiring and Refining Knowledge in Weak Theory Domains
No 94-011 **Britts, Stefan**
Object Database Design
No 94-014 **Kilander, Fredrik**
Incremental Conceptual Clustering in an On-Line Application
No 95-019 **Song, Wei**
Schema Integration: - Principles, Methods and Applications
No 95-050 **Johansson, Anna-Lena**
Logic Program Synthesis Using Schema Instantiation in an Interactive Environment
No 95-054 **Stensmo, Magnus**
Adaptive Automated Diagnosis
No 96-004 **Wærn, Annika**
Recognising Human Plans: Issues for Plan Recognition in Human - Computer Interaction
No 96-006 **Orsvärn, Klas**
Knowledge Modelling with Libraries of Task Decomposition Methods
No 96-008 **Dalianis, Hercules**
Concise Natural Language Generation from Formal Specifications
No 96-009 **Holm, Peter**
On the Design and Usage of Information Technology and the Structuring of Communication and Work
No 96-018 **Höök, Kristina**
A Glass Box Approach to Adaptive Hypermedia
No 96-021 **Yngström, Louise**
A Systemic-Holistic Approach to Academic Programmes in IT Security

No 97-005 **Wohed, Rolf**
A Language for Enterprise and Information System Modelling
No 97-008 **Gambäck, Björn**
Processing Swedish Sentences: A Unification-Based Grammar and Some Applications
No 97-010 **Kapidzic Cicovic, Nada**
Extended Certificate Management System: Design and Protocols
No 97-011 **Danielson, Mats**
Computational Decision Analysis
No 97-012 **Wijkman, Pierre**
Contributions to Evolutionary Computation
No 97-017 **Zhang, Ying**
Multi-Temporal Database Management with a Visual Query Interface
No 98-001 **Essler, Ulf**
Analyzing Groupware Adoption: A Framework and Three Case Studies in Lotus Notes Deployment
No 98-008 **Koistinen, Jari**
Contributions in Distributed Object Systems Engineering
No 99-009 **Hakkarainen, Sari**
Dynamic Aspects and Semantic Enrichment in Schema Comparison
No 99-015 **Magnusson, Christer**
Hedging Shareholder Value in an IT dependent Business society - the Framework BRITS
No 00-004 **Verhagen, Henricus**
Norm Autonomous Agents
No 00-006 **Wohed, Petia**
Schema Quality, Schema Enrichment, and Reuse in Information Systems Analysis
No 01-001 **Hökenhammar, Peter**
Integrerad Beställningsprocess vid Datasystemutveckling
No 01-008 **von Schéele, Fabian**
Controlling Time and Communication in Service Economy
No 01-015 **Kajko-Mattsson, Mira**
Corrective Maintenance Maturity Model: Problem Management
No 01-019 **Stirna, Janis**
The Influence of Intentional and Situational Factors on Enterprise Modelling Tool Acquisition in Organisations
No 01-020 **Persson, Anne**
Enterprise Modelling in Practice: Situational Factors and their Influence on Adopting a Participative Approach
No 02-003 **Sneiders, Eriks**
Automated Question Answering: Template-Based Approach
No 02-005 **Eineborg, Martin**
Inductive Logic Programming for Part-of-Speech Tagging
No 02-006 **Bider, Ilia**
State-Oriented Business Process Modelling: Principles, Theory and Practice
No 02-007 **Malmberg, Åke**
Notations Supporting Knowledge Acquisition from Multiple Sources
No 02-012 **Männikkö-Barbutiu, Sirkku**
SENIOR CYBORGS- About Appropriation of Personal Computers Among Some Swedish Elderly People
No 02-028 **Brash, Danny**
Reuse in Information Systems Development: A Qualitative Inquiry
No 03-001 **Svensson, Martin**
Designing, Defining and Evaluating Social Navigation
No 03-002 **Espinoza, Fredrik**
Individual Service Provisioning
No 03-004 **Eriksson-Granskog, Agneta**
General Metarules for Interactive Modular Construction of Natural Deduction Proofs
No 03-005 **De Zoysa, T. Nandika Kasun**
A Model of Security Architecture for Multi-Party Transactions
No 03-008 **Tholander, Jakob**
Constructing to Learn, Learning to Construct - Studies on Computational Tools for Learning

No 03-009 **Karlgren, Klas**
Mastering the Use of Gobbledygook - Studies on the Development of Expertise Through Exposure to Experienced Practitioners' Deliberation on Authentic Problems
No 03-014 **Kjellman, Arne**
Constructive Systems Science - The Only Remaining Alternative?
No 03-015 **Rydberg Fåhræus, Eva**
A Triple Helix of Learning Processes - How to cultivate learning, communication and collaboration among distance-education learners
No 03-016 **Zemke, Stefan**
Data Mining for Prediction - Financial Series Case
No 04-002 **Hulth, Anette**
Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction
No 04-011 **Jayaweera, Prasad M**.
A Unified Framework for e-Commerce Systems Development: *Business Process Patterns Perspective*
No 04-013 **Söderström, Eva**
B2B Standards Implementation: Issues and Solutions
No 04-014 **Backlund, Per**
Development Process Knowledge Transfer through Method Adaptation, Implementation, and Use
No 05-003 **Davies, Guy**
Mapping and Integration of Schema Representations of Component Specifications
No 05-004 **Jansson, Eva**
Working Together when Being Apart – An Analysis of Distributed Collaborative Work through ICT from an Organizational and Psychosocial Perspective
No 05-007 **Cöster, Rickard**
Algorithms and Representations for Personalised Information Access
No 05-009 **Ciobanu Morogan, Matei**
Security System for Ad-hoc Wireless Networks based on Generic Secure Objects
No 05-010 **Björck, Fredrik**
Discovering Information Security Management
No 05-012 **Brouwers, Lisa**
Microsimulation Models for Disaster Policy Making
No 05-014 **Näckros, Kjell**
Visualising Security through Computer Games
Investigating Game-Based Instruction in ICT Security: an Experimental approach
No 05-015 **Bylund, Markus**
A Design Rationale for Pervasive Computing
No 05-016 **Strand, Mattias**
External Data Incorporation into Data Warehouses
No 05-020 **Casmir, Respickius**
A Dynamic and Adaptive Information Security Awareness (DAISA) approach
No 05-021 **Svensson, Harald**
Developing Support for Agile and Plan-Driven Methods
No 05-022 **Rudström, Åsa**
Co-Construction of Hybrid Spaces
No 06-005 **Lindgren, Tony**
Methods of Solving Conflicts among Induced Rules
No 06-009 **Wrigstad, Tobias**
Owner-Based Alias Management
No 06-011 **Skoglund, Mats**
Curbing Dependencies in Software Evolution
No 06-012 **Zdravkovic, Jelena**
Process Integration for the Extended Enterprise
No 06-013 **Olsson Neve, Theresia**
Capturing and Analysing Emotions to Support Organisational Learning:
The Affect Based Learning Matrix
No 06-016 **Chaula, Job Asheri**
A Socio-Technical Analysis of Information Systems Security Assurance
A Case Study for Effective Assurance

No 06-017 **Tarimo, Charles N.**
ICT Security Readiness Checklist for Developing Countries:
A Social-Technical Approac**h**
No 06-020 **Kifle Gelan, Mengistu**
A Theoretical Model for Telemedicine
- Social and Value Outcomes in Sub-Saharan Africa
No 07-001 **Fernaeus, Ylva**
Let's Make a Digital Patchwork
Designing for Children's Creative Play with Programming Materials
No 07-003 **Bakari, Jabiri Kuwe**
A Holistic Approach for Managing ICT Security in Non-Commercial Organisations
A Case Study in a Developing Country
No 07-004 **Sundholm, Hillevi**
Spaces within Spaces: The Construction of a Collaborative Reality
No 07-005 **Hansson, Karin**
A Framework for Evaluation of Flood Management Strategies
No 07-007 **Aidemark, Jan**
Strategic Planning of Knowledge Management Systems
- A Problem Exploration Approach
No 07-009 **Jonsson, Martin**
Sensing and Making Sense
Designing Middleware for Context Aware Computing
No 07-013 **Kabilan, Vandana**
Ontology for Information Systems (O4IS) Design Methodology:
Conceptualizing, Designing and Representing Domain Ontologies
No 07-014 **Mattsson, Johan**
Pointing, Placing, Touching
- Physical Manipulation and Coordination Techniques for Interactive Meeting Spaces
No 07-015 **Kessler, Anna-Maria**
A Systemic Approach Framework for Operational Risk
- SAFOR
No 08-001 **Laaksolahti, Jarmo**
Plot, Spectacle and Experience: Contributions to the design and evaluation of Interactive Storytelling
No 08-002 **Van Nguyen Hong**
Mobile Agent Approach to Congestion Control in Heterogeneous Networks
No 08-003 **Rose-Mharie Åhlfeldt**
Information Security in Distributed Healthcare
- Exploring the Needs for Achieving Patient Safety and Patient Privacy
No 08-004 **Sara Ljungblad**
Beyond users:
Grounding technology in experience
No 08-005 **Eva Sjöqvist**
Electronic Mail and its Possible Negative Aspects in Organizational Contexts