KTH Computer Science
and Communication

Managing Service Levels in Grid Computing Systems

Quota Policy and Computational Market Approaches

THOMAS SANDHOLM

Licentiate Thesis
Stockholm, Sweden 2007

TRITA CSC-A 2007:6
ISSN 1653-5723
ISRN KTH/CSC/A--07/06--SE
ISBN 978-91-7178-658-6

KTH School of Computer Science and Communication
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av filosofie licensiatsexamen i datalogi måndagen den 14 maj 2007 klockan 10.00 i Sal 304, Parallelldatorcentrum, Teknikringen 14, Kungl Tekniska högskolan, Stockholm.

Tryck: Universitetsservice US AB

Abstract

We study techniques to enforce and provision differentiated service levels in *Computational Grid* systems. The Grid offers simplified provisioning of peak-capacity for applications with computational requirements beyond local machines and clusters, by sharing resources across organizational boundaries. Current systems have focussed on access control, i.e., managing who is allowed to run applications on remote sites. Very little work has been done on providing differentiated service levels for those applications that are admitted. This leads to a number of problems when scheduling jobs in a fair and efficient way. For example, users with a large number of long-running jobs could starve out others, both intentionally and non-intentionally.

We investigate the requirements of High Performance Computing (HPC) applications that run in academic Grid systems, and propose two models of service-level management. Our first model is based on global real-time quota enforcement, where projects are granted resource quota, such as CPU hours, across the Grid by a centralized allocation authority. We implement the SweGrid Accounting System to enforce quota allocated by the Swedish National Allocations Committee in the SweGrid production Grid, which connects six Swedish HPC centers. A flexible authorization policy framework allows provisioning and enforcement of two different service levels across the SweGrid clusters; high-priority and low-priority jobs. As a solution to more fine-grained control over service levels we propose and implement a *Grid Market* system, using a market-based resource allocator called Tycoon.

The conclusion of our research is that although the Grid accounting solution offers better service level enforcement support than state-of-the-art production Grid systems, it turned out to be complex to set the resource price and other policies manually, while ensuring fairness and efficiency of the system. Our Grid Market on the other hand sets the price according to the dynamic demand, and it is further incentive compatible, in that the overall system state remains healthy even in the presence of strategic users.

Keywords: Grid Market, Computational Grid, Service Level Management, QoS, HPC, Grid Middleware

Sammanfattning

Vi studerar metoder för att tillhandahålla och upprätthålla olika servicenivåer i Grid system för storskaliga beräkningar. Grid modellen gör det enklare att tillgodose den maxkapacitet som storskaliga beräkningar kräver genom att möjliggöra ett dynamiskt och automatiserat utbyte av datorkraft mellan olika organisationer. Dagens Grid system fokuserar på behörighetskontroll, dvs hanterande av vem som tillåts köra applikationer på främmande system. Väldigt lite arbete har ägnats åt att erbjuda olika servicenivåer till de som har behörighet. Detta leder till åtskilliga problem när jobb ska distribueras och köras på ett effektivt och rättvist sätt. Användare som kör många långa jobb kan, t.ex. blockera andra körningar, både medvetet och omedvetet.

Vi undersöker kraven som storskaliga beräkningsapplikationer ställer på infrastrukturen i akademiska Grid system, och föreslår två modeller för att hantera servicenivåer. Vår första modell baserar sig på global kvotakontroll i realtid, där forskningsprojekt tilldelas en kvota datorkraft, som t.ex. CPU timmar, av en centraliserad allokeringsenhet. Vi implementerar SweGrid Accounting System, ett system för att se till att resurs kvota som tilldelats forskare av Swedish National Allocations Committee, levereras av ett nätverk av datorer, SweGrid, som sammanbinder sex super- och parallelldatorcentra i Sverige. Ett enkelt konfigurerbart policystyrt auktoriseringsramverk tillåter tillhandahållande och upprätthållande av två olika servicenivåer ; högprioritets- och lågprioritetsjobb. För att få ytterligare och bättre kontroll över servicenivå föreslår och implementerar vi en marknad för Grid resurser som avänder sig av Tycoon, ett marknadsbaserat allokeringssystem för datorresurser.

Slutsatsen av vår forskning är att trots att SweGrid Accounting lösningen erbjuder bättre servicenivåstöd än dagens Grid system, visade det sig vara komplicerat att konfigurera resurspris och andra policyvärden manuellt, och samtidigt tillförsäkra en rättvis och effektiv allokering. Vår lösning med en Grid-marknad å andra sidan sätter priser utefter efterfrågan dynamiskt, och den är *incitamentkompatibel*, dvs systemet som helhet förblir effektivt och rättvist trots att det finns strategiska användare som försöker utnyttja det.

Contents

Chapter 1

Introduction

Large-scale networks are evolving rapidly to become faster, more reliable, and more accessible, which is exemplified by the enormous technical as well as social impact of the Internet. This trend is a result of advances in computer science and engineering, such as more efficient hardware and network protocols, but it is also an indirect result of the advances in physical and social sciences, such as Bioinformatics, High-Energy Physics, and Economics demanding increased capacity for data processing, storage, and transfer. These demands are typically fluctuating over time, making it impractical to purchase dedicated hardware that is unutilized most of the time, and furthermore quickly becomes obsolete. As hardware, operating systems, and networking software are commoditized, it becomes more feasible to share these resources. A new array of computing systems thus evolved to govern the sharing of resources in large-scale networks.

The power-grid utility paradigm is often used to describe such systems. It should be as easy to upgrade your computing capacity as plugging in your appliance into a power socket and turning a knob to get more electricity. One set of problems that need to be tackled to achieve this involves agreeing on standards for communication interfaces and protocols, another is related to ensuring that the shared resources are correctly used in a secure way and that the usage is accounted and charged for regardless of where it was provisioned. A final set of problems involves offering a variety of service levels for customers with different needs and preferences in an economically fair and efficient manner.

The state-of-the-art Grid systems have made great progress in interface standardization recently and have also tackled many of the security related problems involved in executing applications remotely. Grid Accounting systems are in development but not yet widely deployed, and not yet standardized, which complicates charging for compute resource usage. However, the most apparent shortcoming of today's Grid systems is the lack of provisioning and enforcement of service levels. The problem has been addressed from a linguistic perspective by inventing new service level agreement languages and negotiation protocols, but very little has been done to facilitate provisioning and enforcement of these agreements. As a result it is hard to make the current Grid deployments economically sustainable and thereby offered in a commercial setting as opposed to in a government-funded research project.

1.1 Problem Statement

In this thesis we[1] investigate what infrastructure can be added to existing HPC Grid systems to automatically provision and enforce service levels more accurately and easily.

Provisioning of service levels is the process when resource providers offer and advertise different levels of service performance to users, whereas enforcement, a.k.a. policing, of service levels involves making sure that the promised levels are indeed delivered. These two activities are deeply interrelated, and we thus consider the combination, referred to as service-level management here.

1.2 Scope

We examine service-level management from a middleware perspective. That is, we study what tools can be developed to help Grid application programmers take better advantage of the shared resources while still assuring that the overall state of the system is healthy. Our focus is not on advancing research in the graphical end-user interface design nor the design of the networking fabric, but rather to improve the technology that bridges the two.

1.3 Approach

We have investigated two different approaches to service-level provisioning and enforcement in Grids. The first approach relies on a Grid accounting system, which we developed, that allows centrally set project quota policies as well as locally configured resource provisioning policies determine the service-level for users across HPC clusters. The second approach is based on resource virtualization and slicing in a proportional share, market based resource allocator.

Simulations, benchmarks, experiments, and analyzes of production system deployments with real users are all methods used to verify the results and the feasibility of our models and their implementation in different settings and against alternative solutions.

[1] The term *we* is used throughout this thesis to denote the work lead and performed by the author while collaborating with other researchers. Where there are joint contributions, the parts done by the author are explicitly stated.

1.4 Organization

The thesis is organized as follows. In the first part we summarize the background and results of our work. Chapter 2 presents the problem domain and the underlying technology and theory. The software that was developed as part of the thesis research is described in some more detail in Chapter 3 and then the contributions and the thesis papers are summarized with future work in Chapter 4.

In the second part we include the thesis papers previously published in a journal, conference proceedings, a technical report, and a research manuscript.

Part I

Background and Results

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. Second, we review the underlying theory of markets including game theory, and fundamental micro-economic theory.

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) [28, 27, 5]. 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. One of the first super computing projects to span multiple organizations and utilizing a cross-Atlantic Grid was the I-WAY project [17], which paved the way for Grid computing as a scientific field. This community is known as a *Virtual Organization* (VO) [26]. An example VO architecture is shown in Figure 2.1.

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, 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

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

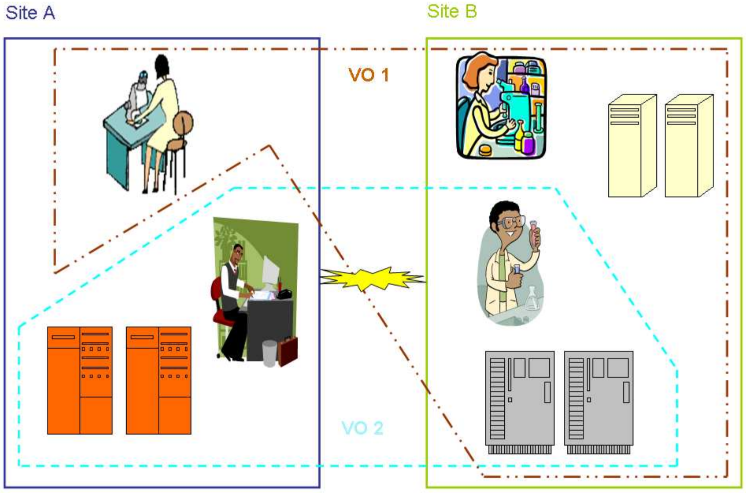

Figure 2.1: Virtual Organization Example.

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) [36], with extensions to allow delegation of rights and single sign-on using self-signed *proxy certificates* [65, 69]. 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 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.

Policy Management

In cross-organizational wide-area networks with community overlays, such as Virtual Organizations in Grid systems, managing policies for all stakeholders becomes a challenging task. The policies of resource providers, funding agencies, virtual organizations, and users must be combined in order to make accurate authorization and service-level decisions. Policies may either be *pulled* in from 3rd parties, by intercepting the message flow and making call-outs, or *pushed* to decision makers by attaching capability assertions to the message payload [69]. Combinations of the push- and the pull-models are also common and the policy decisions may be made both on the client and on the server side in a client-server interaction [47]. Policy-based systems aim to manage resources by enforcing, evaluating, retrieving, administering and combining policies with standard protocols. By communicating with all the heterogeneous resources in the same way, generic tools can automatically manage arbitrarily complex networks of resources and stakeholders by means of control feedback-loops, a.k.a. the MAPE (Measurement, Analysis, Planning, and Execution) or autonomic self-management model [41]. Policy-based management systems can dynamically change the configuration of resources in response to events that in turn were triggered by various system states occurring. A key to designing efficient policy-aware systems is to separate the policy-related tasks into different layers and making them agnostic to the application code. These layers, often referred to as *policy points*, can be stacked and combined in arbitrarily complex policy-decision trees. The different layers and their responsibilities, summarized below, comprise a common architecture and nomenclature for policy-based systems [1].

- A Policy Enforcement Point (PEP) intercepts the execution path and calling decision points, i.e. a PEP integrates the application with the policy mechanisms.

- A Policy Decision Point (PDP) combines and evaluates local as well as external policies, and may call information points to retrieve policies to base its decision on. The result from an evaluation is typically *permit*, *deny*, or *not applicable*. The result may also contain obligations that must be met for the result to take effect.

- A Policy Information Point (PIP) stores and retrieves policies, e.g. returns roles that an authenticated user has in an RBAC (Role-Based Access Control) system [21]. The additional information can then be used by the PDP and is sometimes called evidence or context credentials.

- A Policy Administration Point (PAP) sets and configures policies used by the PDP and PIP layers.

Accounting and Systems Integration

Grid middleware services include secure remote execution management, remote data storage and replication, monitoring services, and high-volume file-transfer services. What distinguishes these services from other network services such as FTP, WWW, and LDAP, is that a Grid needs to handle higher volumes of data to transfer and store; and allow VO-enabled access control, and execution of arbitrary applications. Furthermore, all users and Virtual Organizations are accountable for their resource usage in a Grid, in order to promote fair sharing.

Accounting services thus play a fundamental role in Grids. Grid accounting systems must be able to handle VO-scoped accounting of the usage of heterogeneous resources. Standard accounting records that translate and coalesces resource and site specific usage need to be exchanged and coordinated across the Grid. This process is complicated by the diversity of resource management technology and policies (e.g., security, accounting, auditing) in HPC centers offering Grid resources [57, 19, 58].

Resource heterogeneity in Grid systems was first addressed by the Open Grid Services Infrastructure protocol [66], which specifies a standard protocol interface for managing the state of a Grid resource [25]. It was founded on state-of-the-art systems integration technology of the time, including the XML Web services stack with extensions, and it was adopted by the Global Grid Forum (GGF) standardization body. This work later evolved into the Web Services Resource Framework [30], within the OASIS standards group, which now makes up the backbone infrastructure of various distributed management standards, such as OASIS WSDM [10] and GGF WS-Agreement [3].

Resource Allocation

Service level and QoS enforcement was addressed in a Grid context in the Grid Advanced Reservation and Allocation (GARA) [22, 23] 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 [6, 8], and DSRT CPU scheduling control [49]. This work evolved into the SNAP protocol [15] 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 [16] and SLAng [46].

Complimentary to protocol standardization, heterogeneity can also be addressed by resource virtualization. For example, virtualization of a host operating system [18] 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 [40].

As the Grid deployments extend beyond academic projects, such as EDG [2] [7], 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 computing strategy [31, 12, 35]. 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.

2.2 Market 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 economic 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* [33]. 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 compute power is offered as a common good without providing some incentive for users to constrain their usage.

[2] European Data Grid, http://www.edg.org

[3] Enabling Grids for ESciencE, 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

Game Theory

In *Game Theory* [52, 51] 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

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

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

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

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

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

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* [53] 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

$$U(\omega) = \sum\nolimits_{i=1}^{k} u_i(r_i)$$

where 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* [67] which can be calculated as

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

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 an 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 and commensurable 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 [20, 72]. Given a fixed budget and a pool of *divisable* resources allocated according to the proportional share mechanism described above, 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 [72] 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 efficiency bound of $\Theta(\frac{1}{\sqrt(m)})$ and an envy-freeness of $2\sqrt(2) - 2$ or approximately 0.828 in Nash equilibria with m players are theoretically deduced.

Chapter 3

Software

The research results of this thesis were obtained by implementing service-level management support for two Grid and cluster middleware toolkits. Three types of experiments were then performed. First, local simulations were run to test the algorithms against theoretical models, where both resource users and providers were simulated. Second, simulated users were run against providers deployed in the real Grid. Finally, real users and applications were run against the real Grid.

The SweGrid Accounting System (SGAS) was implemented as a Grid accounting system on top of the Globus Toolkit, and the Tycoon Grid Resource Allocator was implemented as a Grid market broker on top of the Tycoon market-based resource allocator. The general design of the two systems will be discussed below.

3.1 SweGrid Accounting System

We developed the SweGrid accounting system[1] to meet the quota enforcement needs of SweGrid, a national compute resource integrating 600 nodes at six HPC Centers across Sweden [57, 19, 58]. Resource quota is granted to research projects after peer review by the Swedish National Allocation Committee (SNAC). The quota can then be consumed by running jobs on any of the six participating sites. The main challenge was to consolidate the heterogeneous local accounting and security policies into one uniform accounting system capable of charging and enforcing allocations globally and in real-time. Our solution was to develop a Web services architecture based on a generic authorization policy framework capable of administering, storing, enforcing, and validating stakeholder policies at runtime. The stakeholders in SweGrid are the Grid application users, the resource providers, and the allocation authorities. Service-level management is carried out jointly by three services: the *Bank*, the *Logging and Usage Tracking Service* (LUTS), and the *Job Account Reservation Manager* (JARM). An overview of these services is shown in Figure 3.1.

[1] http://www.sgas.se

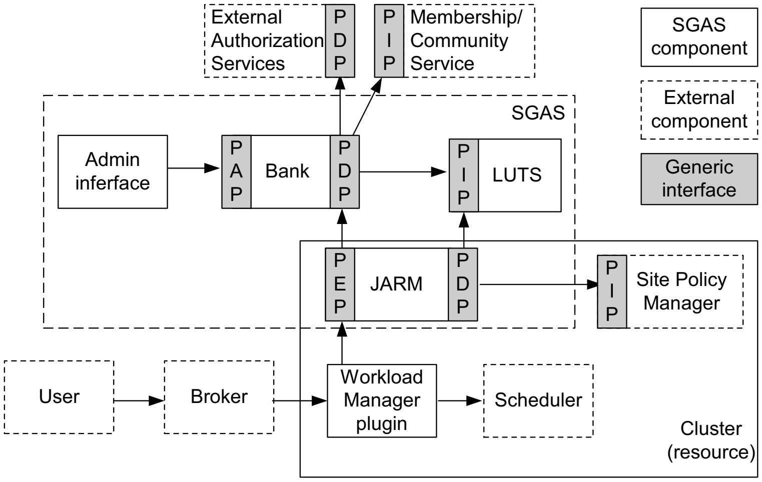

Figure 3.1: SGAS Components Overview.

Bank

The Bank service manages resource quota on a project and user basis. An account in the bank is created for each research project and then the principal investigator of the project can add all the users who should be allowed to submit jobs that are allowed to consume the project quota. The Bank service can be queried for available funds in an account, and *holds* of parts of the funds can be issued and then charged. The Bank thus represents the allocation authority stakeholder.

LUTS

The Logging and Usage Tracking Service allows off-line accounting after the jobs have run, and off-loads the bank from storing detailed logging and auditing records. The LUTS can be queried by all stakeholders as a means to making allocation and authorization decisions based on previous history.

JARM

The Job Account Reservation Manager component integrates the local accounting system and job manager infrastructure with the SGAS services. JARM implements the resource provider policies to enforce service levels. Currently only a binary service-level model is implemented where the job either runs as a full-priority task if enough quota is available in the Bank, and as a low-priority task if the quota has been exceeded.

3.2 Tycoon Grid Resource Allocator

Tycoon

Tycoon is a market-based resource allocation system allowing resource shares to be auctioned out proportionally to users' bids [43, 44, 42]. 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

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

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

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

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

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 [20] that solves the following optimization problem for a user: from a set of n resources pick the set $\{x_{i,j}...x_{i,n}\}$ that

$$\text{maximizes } U_i = \textstyle\sum_{j=1}^{n} w_{i,j} \frac{x_{ij}}{x_{ij}+y_j} \text{ subject to } \sum_{j=1}^{n} x_{ij} = X_i, \text{ and } x_{ij} \geq 0. \quad (3.1)$$

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 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, which they can do at any time.

Grid Market

As part of our investigation of service-level management in Grid systems we developed a Grid broker on top of Tycoon (see Figure 3.2), 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 3.1. 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. An important addition to Tycoon that we also developed was to provide 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.

The user interface of the broker uses the Nordugrid ARC *meta-scheduler* [62] which in turn is based on the Globus Toolkit [24], both extensively deployed in production Grid systems worldwide.

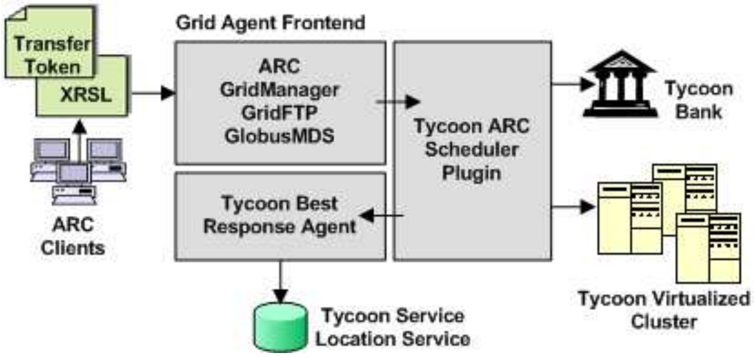
Transfer Token
XRSL
ARC Clients
Grid Agent Frontend
ARC GridManager GridFTP GlobusMDS
Tycoon ARC Scheduler Plugin
Tycoon Best Response Agent
Tycoon Bank
Tycoon Virtualized Cluster
Tycoon Service Location Service

Figure 3.2: Tycoon Grid Market Architecture.

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:

- **Job Payments.** A Grid user can pay for her jobs by attaching a transfer token to the job submission. The transfer token is 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 3rd party to clients who don't 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.
- **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 allowing 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[2] installer to automatically provide a wide range of installation packages that may optionally be installed on demand before the job is run, and thus customizing 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 itself, 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

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

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 don't 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 submission infrastructure we provides the same command line interface as OpenPBS [3], one of the most common cluster job submission toolkits.

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

Chapter 4

Results

In this chapter, the paper contributions attached to the end of this thesis are summarized. The papers represent the evolution of approaches used to solve the service-level provisioning and enforcement problem discussed in Section 1.1. In Paper 3, the contribution from the work conducted as part of this thesis is limited to the results section and to performing the experiments. All other papers were authored as full parts of this thesis.

We also summarize our contribution to other publications, which were only co-authored or only indirectly related to the service-level problem addressed here. Finally we summarize related work and conclusions.

Various research projects funded parts of this work, including the Swegrid accounting project (Swedish Research Council), Enabling Grids for EScienceE (European Union), NextGrid (European Union), Globus (Globus Alliance), and Tycoon (Hewlett-Packard and Intel JIP).

The thesis author's contribution level is given within parenthesis in each paper headline.

4.1 Thesis Papers

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

In this journal article[1] [58] 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

[1] 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.

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 resolved.

Paper 2: Service Level Agreement Requirements of an Accounting-Driven Grid (100%)

In this technical report[2] [56] we discuss the requirements obtained after studying the first production deployment of the accounting system presented in [58]. 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.

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 to the research presented in this thesis is the mapping of typical Grid user requirements to an agent-based, contract-driven architecture. The first insight gained from the SweGrid accounting system [58], 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.

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

Agents could thus use contracts embodying user and provider preferences to optimize user utility, or provider profit and utilization by automatically setting these policies.

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

In this manuscript[3] [45] 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 as 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.

Paper 4: Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications (90%)

In this conference paper[4] [59] we combine the results from the previous three papers by providing 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

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

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

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 as a lightweight contract simulating a 'gift certificate' to purchase resource shares, b) a broker receiving the transfer token attached to the jobs to be submitted, which 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. It was shown that a continuous service level (as opposed to the binary one in SweGrid) could be offered proportional to the funding of the job. The account management is also simplified in our Grid Market, as the local accounts are created on demand and 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 that in turn can be used to submit jobs. Therefore, our Grid Market has the desirable property of offering a closed-loop sharing of resources among peers, true to the foundational idea of the Grid.

4.2 Additional Publication Contributions

Contribution 1 to 5 are co-authored papers and 6 is a lead-authored technical report.

Contribution 1 (10%)

The Global Grid Forum Open Grid Services Infrastructure (OGSI) specification [66] introduces many of the fundamental integration concepts that the SGAS work is based on. We contributed the XML rendering of that specification.

Contribution 2 (90%)

A conference version of Paper 1 was presented in [57]. It contains some additional Fuzzy Logic experiments and it is based on an earlier Web service integration platform. Paper 1 also contains some lessons learned from deploying the solution presented in [57] in SweGrid.

Contribution 3 (50%)

The Bank service of SGAS is presented in some more detail in the conference paper [19]. The Bank was implemented by a collaborator, but the core Web services infrastructure, and the access control and policy framework was contributed as part of this thesis. The overall design of the Bank was also a collaborative effort.

Contribution 4 (10%)

The SGAS authorization framework was contributed to the Globus Toolkit, and it is the foundation for extended work presented in the workshop publication [61]. Our authorization framework, in turn, borrows many concepts from the XACML architecture [1] and the GGF Authorization Working Group model [47].

Contribution 5 (10%)

SGAS provides a testbed for authorization management rights delegation, in the conference paper [60]. This work is also based on the authorization policy framework developed as part of SGAS, and extends it by integrating a 3rd-party authorization engine as a policy administration and decision point.

Contribution 6 (100%)

In the technical report [55] 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.

4.3 Related Work

Related work fall into three categories; first, systems that focus on the accounting aspect of the problem; second, general purpose computational economies; and finally Grid market systems. These categories can be related to our work with SGAS, Tycoon and the Tycoon Grid market respectively.

The DataGrid Accounting System (DGAS) [32] was an early approach to create a closed-loop accounting system for the LHC Grid at CERN, capable of exchanging virtual Grid credits for computational resource time. The project focussed mostly on providing an economic infrastructure for exchanging credits, but did not provide any price setting mechanism, like the one implemented in Tycoon. Furthermore, it did not take the integration approach used by SGAS, which made it difficult to deploy in Swegrid, without completely replacing the existing accounting and job submission infrastructure used by the different HPC sites. The GridBank project [4] took a similar approach to SGAS in that only a single call-out to a bank is necessary to verify the availability of funds to execute the job on the requested resource. They also took a similar approach to our Grid market by attaching a cheque-like token to the job-submission request to pay for the job. It, however, lacks the policy customization infrastructure of SGAS, allowing different resources to easily implement different policies for running and charging for external jobs. SGAS also implements account holds which can be seen as soft reservations of a portion of the account balance, where jobs are only charged for the amount of resources actually consumed. A similar hold approach is implemented in the Gold accounting system [38], which also has expiring account quotas similar to SGAS. Gold did, however, not take the standards-based Web services architecture approach central to the design of SGAS, which also made it harder to integrate with an arbitrary local HPC site accounting system. Neither GridBank nor Gold have any price-setting mechanisms nor the same flexible authorization framework implemented in our work. Additional related accounting approaches can be found in [64, 37, 34].

Spawn [68], 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. However, the general, continuous bid and proportional share auction architecture is largely the same. Bellagio [50] 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 decentralized, i.e. runs an auction at every single host, and does not explicitly operate on complementarities. The efficiency in Tycoon may thus not be as high but all the overhead and computational complexities of combinatorial auctions, as well as the issues with strategic users gaiming the mechanism is avoided [45]. Related computational economy approaches are described in [54, 48, 29, 9, 63, 13].

Faucets [39] is a 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 Faucet 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 [70] 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 Tycoon the best response algorithm ensures fair and efficient allocations across resources. 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 our SGAS work) 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. Additional Grid Market models are described in [14, 71, 11].

4.4 Conclusions

Our work with the SweGrid Accounting System advances the state-of-the art of academic production Grid systems for High Performance Computing tasks by providing real-time quota enforcement across the Grid governed by a flexible policy framework. It thereby improves the overall fairness in the system. However, it only enforces two levels of service, and it does not provide any price-setting mechanisms. Furhtermore, it is very complex to manage all the policies manually without some broker or agent layer between users or providers, and the accounting system. The quota allocation model fits the SweGrid SNAC, and US NRAC periodic central allocation schemes, but it does not promote fast low-burden entry for new users.

Tycoon, addresses all of these problems by implementing a market for virtualized computational resources, allowing any service levels to be configured proportional to a user's bid and inversely proportional to the demand of the resources. Our main contribution in this thesis is hence the merge of the Tycoon market mechanisms with the Grid, thus creating a market appropriate for hosting both academic and commercial Grid applications. Dynamic host account creation and configuration according to purchased service levels, transfer of incentive compatible job tokens, and a combined identity and capability-based authorization model were all important parts of our solution.

4.5 Future Work

Tycoon implements a spot market, in order to quickly adapt the prices to the demand, and to allow important tasks to preempt currently running lower-priority tasks. However, these features come at the cost of less predictability and reduced guarantees of service levels. To address this issue we are working on enhanced prediction techniques to estimate future demand and give users tools to budget their future resource requirements more efficiently.

We would also like to investigate the combination of spot and reservation markets (such as derivative markets, e.g. options) for Tycoon, as well as contract brokers guaranteeing service levels, and offering discounts (paying penalties) if the promised level of service was not delivered.

4.6 Acknowledgments

First and foremost I would like to thank Bernardo Huberman and Kevin Lai, at Hewlett-Packard Laboratories in Palo Alto, both for their invaluable technical insights and feedback on my work, and for their continuous support to allow me to extend my stay at HP Labs to complete my work. I am also very grateful to all the technical support and contributions from Olle Mulmo at the Royal Institute of Technology in Stockholm related to the Grid security work in this thesis. Peter Gardfjell from Umeå University co-designed and co-authored the SGAS system with me and made great contributions to the Bank service. I would also like to thank his advisor Erik Elmroth for his help on finalizing the SGAS publications. Finally, I would like to thank my own advisor Lennart Johnsson, and Lars Rasmusson for their feedback.

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] 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.

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

[6] S. Blake, D. Black, M. Carlson, E. Davis, W. Zheng, and W. Weiss. Rfc 2475: An architecture for differentiated services. Technical report, IETF, 1998.

[7] 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.

[8] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Rfc 2205: Reservation protocol (rsvp) version 1 functional specification. Technical report, IETF, 1997.

[9] 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.

[10] Vaughn Bullard, Bryan Murray, and Kirk Wilson. An introduction to wsdm. Technical report, OASIS, 2006.

[11] 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.

[12] 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. URL `http://research.sun.com/techrep/`.

[13] Anthony Chavez, Alexandros Moukas, and Pattie Maes. Challenger: a multi-agent 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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. URL `citeseer.ist.psu.edu/dragovic03xen.html`.

[19] 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.

[20] 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.

[21] D. Ferraiolo and R. Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[22] 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.

[23] 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.

[24] 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.

[25] Ian Foster, Carl Kesselman, Jeffrey Nick, and Steven Tuecke. Grid services for distributed system integration. *Computer*, 7:37–46, March 2002.

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

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

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

[29] 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.

[30] S. Graham, A. Karmarkar, J. Mischkinksy, I. Robinson, and I. Sedukhin. Web services resource 1.2. Technical report, OASIS, 2005.

[31] 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. URL `http://www.hpl.com/techreports/2003`.

[32] A. Guarise, R. Piro, and A. Werbrouck. Datagrid accounting system - architecture - v1.0. Technical report, EU DataGrid, 2003.

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

[34] V. Hazelwood, R. Bean, and K. Yoshimoto. Snupi: A grid accounting and performance system employing portal services and rdbms back-end. 2001.

[35] 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.

[36] R. Housley, W. Ford, W. Polk, and D. Solo. Rfc 2459: Internet x.509 public key infrastructure and crl profile. Technical report, IETF, 1999.

[37] S. Jackson. Qbank: A resource management package for parallel computers. Technical report, Pacific Northwest National Laboratory, Washington, USA, 2000.

[38] S. Jackson. The gold accounting and allocation manager, 2004. http://sss.scl.ameslab.gov/gold.shtml.

[39] 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.

[40] 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.

[41] J. Kephart and D.M. Chess. The Vision of Autonomic Computing.

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

[43] Kevin Lai, Bernardo A. Huberman, and Leslie Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical report, arXiv, 2004. http://arxiv.org/abs/cs.DC/0404013.

[44] 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. Technical Report arXiv:cs.DC/0412038, HP Labs, Palo Alto, CA, USA, December 2004.

[45] 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.

[46] 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.

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

[48] 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.

[49] K. Nahrstedt, H. Chu, and S. Narayan. QoS-aware resource management for distributed multimedia applications. *Journal on High-Speed Networking* , December 1998.

[50] 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.

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

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

[53] Christos H. Papadimitriou. Algorithms, Games, and the Internet. In *Symposium on Theory of Computing*, 2001. URL `citeseer.ist.psu.edu/papadimitriou01algorithms.html`.

[54] 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.

[55] 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. http://www.pdc.kth.se/ sandholm/trita/SandholmOntologyV2.pdf.

[56] 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. http://www.pdc.kth.se/ sandholm/trita/TRITA-SLA.pdf.

[57] 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.

[58] 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. (to appear).http://www.worldscinet.com/ijcis/ijcis.shtml.

[59] 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.

[60] 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. http://pat.jpl.nasa.gov/public/grid2005/index.html.

[61] Frank Siebenlist, Takuya Mori, Rachana Ananthakrishnan, Liang Fang, Tim Freeman, Kate Keahey, Sam Meder, Olle Mulmo, and Thomas Sandholm. The globus authorization processing framework, April 2005. http://lotos.site.uottawa.ca/ncac05/index.html.

[62] 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.

[63] 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.

[64] W. Thigpen, J. Hacker, L. McGinnis, and B. Athey. Distributed accounting on the grid. Technical report, Global Grid Forum, 2001.

[65] 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. http://www.ietf.org/rfc/rfc3820.txt.

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

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

[68] 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. URL `citeseer.nj.nec.com/waldspurger91spawn.html`.

[69] 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.

[70] 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.

[71] 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.

[72] 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.