Market-oriented Grids and Utility Computing: The state-of-the-art and future directions

James Broberg, Srikumar Venugopal and Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Information Technology
The University of Melbourne, Australia
{brobergj,srikumar,raj}@csse.unimelb.edu.au

August 14, 2007

Abstract

Traditional resource management techniques (resource allocation, admission control and scheduling) have been found to be inadequate for many shared Grid and distributed systems that face unpredictable and bursty workloads. They provide no incentive for users to request resources judiciously and appropriately, and they do not capture the true value and importance (the *utility*) of user jobs. Consequently, researchers and practitioners have been examining the appropriateness of 'market-inspired' resource management techniques in ensuring that users are treated fairly, without unduly favouring one set of users over another. Such techniques aim to smooth out access patterns and reduce the chance of transient overload, by providing incentives for users to be flexible about their resource requirements and job deadlines. We examine the recent evolution of these systems, looking at the state of the art in price setting and negotiation, grid economy management and utility-driven scheduling and resource allocation, and identify the advantages and limitations of these systems. We then look to the future of these systems, examining the emerging 'Catallaxy' market paradigm and present Mercato, a decentralised, Catallaxy inspired architecture that encapsulates the future directions that need to be pursued to address the limitations of current generation of market oriented Grids and Utility Computing systems.

1 Introduction

The rise of Grid Computing [14] has led to knowledge breakthroughs in fields as diverse as climate modelling, drug design and protein analysis, though the harnessing of computing, network, sensor and storage resources owned and administered

by many different organisations. These fields (and other so-called Grand Challenges) have benefited from the economies of scales that Grid Computing brings, tackling difficult problems that would be impossible to feasibly solve using the computing resources of a single organisation.

Despite the obvious benefits of Grid Computing, there are still many issues to be resolved. As the Grid Computing paradigm has become more popular, large Grid Computing organisations are facing new problems, such as excessive spikes in demand for resources coupled with strategic and adversarial behaviour by users. Such conditions have been observed on PlanetLab [29, 7], which whilst not specifically considered a Grid is one of the largest open-access distributed systems of it's kind, and Mirage [10], a shared sensornet testbed, among other examples.

In both of these systems, bursty periods of high contention for available resources have been observed, where demand frequently exceeds supply. PlanetLab load often corresponds closely to deadlines for major computing and networking conferences, where disparate research teams are competing for resources to run their experiments on [29]. Mirage is one of very few real sensor network testbeds available, and is of prime interest to commercial and academic researchers across the globe wishing to explore the behaviour of their algorithms on a real system [26]. As a result, it can be difficult to get access to this system at peak times.

In these situations, traditional resource management techniques (resource allocation, admission control and scheduling) have been found by many researchers to be lacking in ensuring fair and equitable access to resources [21, 35]. Traditional resource management techniques for clusters focus on metrics such as maximising throughput, and minimising mean waiting time and slowdown. These metrics fail to capture more subtle requirements of users, such as quality of service constraints. Consequently, researchers have been examining the appropriateness of 'market-inspired' resource management techniques in ensuring users are treated fairly, without unduly favouring one set of users over another. To achieve this, there needs to be incentives for users to be flexible about their resource requirements and job deadlines, and utilise these systems outside of peak time. Similarly, there needs to be provisions for a user with urgent work to be satisfied rapidly, where possible. These aims are achieved by users assigning a *utility* to their jobs - effectively a fixed or time-varying valuation that captures various quality of service constraints (deadline, importance, and satisfaction) associated with a users job. This utility typically represents the amount they are willing to compensate a service provider to satisfy their job demands. Shared Grid systems are then viewed as a marketplace, where users compete for resources based on the perceived utility or value of their jobs.

The notion of utility is not restricted to end-users alone, especially in commercial grid and cluster systems. As is the case in free markets, all participants

(described in Section 2.1) are self-interested entities that attempt to maximise their own gain. Service providers in commercial systems will attempt to maximise their own utility. In this instance, the utility they receive may directly correlate to the profit (i.e. the difference between the cost of offering the service and the compensation they receive) they make. One or many brokers can operate in grid and cluster systems, acting as a middleman between end-users and service providers. They can perform a number of important functions, such as aggregating resources from many providers and negotiate and enforce quality of service targets, reducing the complexity of access for end-users. As compensation for performing these value-added functions, brokers generate utility for itself by selling access to resources at a higher cost than what they pay a service provider, generating a profit.

In this paper we examine many recent advances in the field of utility-oriented grid computing markets. There have been prior surveys of market-based resource management [39], utility computing [40] and Grid Economies [9]. This survey is intended to complement these prior works, by covering the most recent research in Grid Computing Markets. In Section 2 we provide an overview of market-based approaches for utility-driven distributed systems, highlighting the benefit of utility-driven computing and explaining the role of the market in Grid computing. In Section 3 we examine the state of the art in market-based utility computing, examining the benefits of utility-driven pricing, resource allocation, scheduling, admission control. Section 4 describes the so-called 'Catallaxy' paradigm, which has moved the research focus from stand-alone grid marketplaces to multiple, linked, decentralised and autonomic 'free market'. In Section 5 we present our Catallaxy-inspired ad-hoc Grid Market architecture, called Mercato.

2 Overview of market-based approaches for utility-driven distributed systems

As networked resources such as grids, clusters and sensors are being aggregated and shared amongst multiple stake-holders with often competing goals, there has been a rising consensus that traditional scheduling techniques are insufficient. Indeed, simply aiming to maximise utilisation for service providers and minimise waiting time and slow down for end-users does not always capture the diverse valuations that participants in these systems place on the successful execution of jobs and services. Instead, the notion of maximising the *utility* of each participant in these system is becoming a priority.

The behaviour, roles and responsibilities of each of these participants as well as how they each measure utility are explored in Section 2.1. An overview of the motivation behind market-drive utility computing and the obstacles that need to be

overcome is presented in Section 2.2. In Section 2.3 we look at some of the emerging technology trends that are allowing service providers unprecedented flexibility in partitioning and allocating their resources for computing marketplaces.

2.1 Participants

Participants, or 'actors' in a utility-oriented Grid Computing markets can be generally classified as belonging to one of three categories: users, brokers and service providers. In some instances, participants may actually perform the functions from more than one category - a users may also offer some of its own resources to other participants, (acting as a service provider), or a service provider may aggregate other resources along with its own (acting as a broker). Each actor in the system is a self-interested, utility maximising entity. How each actor measures and maximises its utility depends on the system it operates in - the behaviour exhibited in a shared system where market-driven techniques are used simply to regulate access differs greatly from a profit-driven commercial system. Many of these differences are highlighted from out study of existing systems in Section 3.

A user requires access to more resources than it has available, for requirements that can range from jobs to be processed, to web services that need to be run on networked computers. These users are willing to proportionally compensate a provider to satisfy their requirements depending on the utility they receive. The sophistication of this compensation depends on the system being used, as outlined in Section 3.1.

Rather than dealing with multiple heterogeneous service providers directly, users obtain access to resources via one or many brokers. These brokers act as 'middlemen' by virtualising and making available the resources of multiple service providers, removing the complexity from the user of multiple architectures, operating systems and middleware. This aggregation allows greater economies of scale, improving throughput and reducing cost. Brokers can have more specific responsibilities, such as reserving resource slots, scheduling jobs and services on resources held by service providers, performing admission control to avoid overload and ensuring that a user's quality of service targets can be met.

Service providers satisfy the needs of end-users by providing the resources (i.e. disk, memory, CPU, access to sensors, etc.) that is requested by end users and arbitrated by brokers. A service provider may offer its services through multiple brokers, and even offer more resources than it has available to give, in an effort to improve utilisation via statistical multiplexing. A service provider is ultimately responsible for ensuring that all its commitments are met. It must ensure appropriate performance isolation for jobs and services running on its resources, to ensure that users quality of service targets are satisfied. This can be achieved by appropriate

partitioning or scheduling of its resources.

2.2 Utility computing and the role of market-based techniques

Lai attempts to motivate the use of market-based techniques for resource assignment, whilst exploring the pitfalls of such approaches [21]. Over a long enough time frame, Lai notes that Proportional Share is not sufficient to reach economic efficiency, as there is no incentive for users to shift their usage from high demand periods to low demand periods. Simple fixed price schemes are not as efficient as variable price in addressing this transient peak load, assuming a variable demand. Indeed, the more variable the demand the greater the efficiency loss of a fixed price scheme. When this is combined with a controlled pool of funds, users have an incentive to truthfully reveal their valuation of tasks. Finer grained bidding for resources can occur, where users can choose between reserved resources or best effort scheduling, depending on necessity.

Shneidman et al. have noted important emerging properties of these systems, where users are self-interested parties that both consume and supply resources, demand often exceeds resource supply, and centralised (global) approaches cannot be used due to the sheer scale of these systems [35]. The authors claim the goal is no longer maximising utilisation, especially when demand exceeds supply, and more intelligent allocation techniques are needed than just best effort or randomised allocation. They propose that a efficient allocation mechanism (i.e. social policy) that can allocate resources is needed that allocates resources to users who have the highest utility for them, favours small experiments (a common practice in scheduling) and underrepresented stake-holders (e.g. those that have been denied or refrained from receiving service in the past) and maximises revenue.

The authors note that many recent developments make market-based techniques appropriate and timely, as they could be immediately deployed in many commercial scenarios (such as test-beds and shared grids) to solve real world problems. Developments in Virtual Machine technology (Xen [6], VMWARE, etc.) and other resource isolation techniques (Linux Class-based Kernel Resource Management, BSD Jails [32]) can be used to partition, isolate and share resources. Combinatorial bidding languages are emerging that are sufficiently expressive, and can be often solved as mixed integer optimisation problems in modern solver packages like CPLEX [27].

Shneidman et al. state that there are a number of problems that need to be solved to make effective market-based, utility driven approaches a reality. Resource allocation policies must be explicitly defined, particularly, what are the social policy goals that need to be met? However, we note that it is unclear at this point who (i.e. users, administrators, service providers) should mandate these

goals. Sellers of resources must be able to divide resources efficiently both explicitly and implicitly (e.g. memory is useless without at least some CPU) into resource bundles. However, whilst tools to achieve this (e.g. Xen, CKRM) are maturing they still have some way to go. Buyers needs must be predicted so required resources are not misestimated, leading to waste (e.g excess resources that have been reserved and not used, or wasted due to unfinished experiments that may be invalid).

Forecasting resource requirements is challenging and new for users. It is unclear whether market can resolve this, or whether 'practice' runs in a best effort staging grounds are needed so users can gain experience predicting their needs. However, such staging grounds may be impractical in reality, and is in opposition to the free market approach of 'letting the market work it out' by allowing users to suffer the consequences of poor choices. Users need to find the true value of resources they wish to reserve. The currency system needs to be well defined (either virtual or real), and receive ongoing care so that it functions correctly to maximise the utility of the system, avoiding the typical problems of starvation (users run out of money), depletion (users hoard currency or leave the system) and inflation (currency is injected into the system without limit). The use of real money is favoured by many researchers to encourage more honest valuations of resources in the system. An effective means to calculate and express valuation of resources is needed for efficient operation of these systems. Bidding languages are becoming more sophisticated and are well suited to this task. However, intuitive interfaces are needed to construct these bids, allowing users to easily and accurately express their preferences.

Linked market-based mechanisms have been proposed as an interesting possibility, quantifying the value of cluster time at one network versus another (in a similar fashion to currency exchanges). This is complementary with the move from centralised approaches to catallaxy inspired [13, 3] distributed and autonomic systems, described in Section 4.

2.3 Emerging technologies for computing marketplaces

The more recent growth in popularity of broadly accessible virtualisation solutions (such as Xen [6], VMWARE [1]) and services (Amazon Cloud [2], PlanetLab [7]) provides an interesting framework where a lightweight virtual machine (VM) image can be a unit of execution (i.e. instead of a 'task' or process) and migration [25]. Whilst virtual machines and containers are hardly new technology, they have only recently become ubiquitous, and now run with hardware assistance (using paravirtualisation on recent Intel VT-x and AMD Pacifica enabled processors) on commodity computers and operating systems, rather than expensive

'big-iron' Unix machines. Recent developments, described by Nelson et al., have allowed migration of unmodified applications encapsulated in virtual machines, even if these applications to be unaware of the mechanics behind it. This migration can be achieved in a work-conserving fashion without the running application nor any dependant clients or external resources being aware that it has occurred. The VMWARE management capabilities makes this possible encapsulating the state of the VM, such CPU, networking, memory and I/O while the virtual machine is still running. It can transfer open network connections due to the layer of indirection provided by the VMWARE Virtual Machine layer. Physical memory is often the largest overhead in terms of state that must be migrated. Stopping a VM to save and transfer this state can cause a lengthy downtime, making the migration process far from transparent. As such, this is handled *in situ* by a virtual memory layer [37], allowing memory state to be transferred whilst a VM is still running by iteratively pre-copying memory to the destination node and marking it as temporarily inaccessible to the source.

Nelson et al. found that for a variety of CPU, I/O and memory bound workloads, VM migration in a local cluster can be fast and transparent for the applications themselves and any dependent clients and external resources. In most causes downtime was kept under a second, short enough to avoid a noticeable lapse in service by clients. If the VMs are simply batch workloads, this is even less of an issue. Even if migration takes several seconds, this must be contrasted with the probable runtime of a virtual machine, which could be in the order of minutes or hours. As such, the benefits of migration would depend on the nature of the virtual machine workload.

3 The state-of-the-art in market-based utility computing

Utility-driven distributed computing marketplaces are typically made up of several key components and processes. Users can have jobs that need to be processed, for which they are willing to compensate an entity to perform. The level of compensation depends on a number of factors, including the currency available to the client, the contention for resources and the urgency of the job. These factors are considered in the price setting and negotiation phase, described in Section 3.1.

As described previously, one or many brokers can exist in a grid marketplace, acting as a 'middleman' by aggregating and virtualising access to disparate Grid resources offered by service providers. In the face of competition from many clients for grid resources, brokers and service providers must manage these resources effectively, meeting demand where possible but ultimately maximising the utility (i.e. revenue) gained. This is achieved via utility-driven resource allocation, scheduling

and admission control, described in Section 3.2.

For a grid marketplace to be sustainable, the grid economy must be managed effectively. If users have an unlimited supply of currency, there is no rewards or disincentives for users to manage their currency and use it prudently and truthfully, revealing their true valuation for jobs. As such, the currency (whether real or virtual) must be carefully managed to ensure equity for all participants. This is discussed in further detail in Section 3.3.

Some systems address many or all of the above aspects of market-based, utility-driven distributed systems. As such, we will first introduce these systems below:

Buyya et al. proposed an economy driven resource management architecture for global computational grids [8, 9]. This consists of a generic framework, called **GRACE**, for negotiation and trading resources dynamically in conjunction with existing grid components, such as local schedulers and grid or meta-schedulers. The function of GRACE is to enable supply and demand-driven pricing of resources to regulate and control access to computational resources in a grid.

The **Bellagio** [4] is a system that seeks to allocate resources for distributed computing infrastructures in an economically efficient fashion to maximise aggregate end-user utility. In this system, users identify resources of interest via a resource discovery mechanism (such as SWORD [28]) and register their preference (via a constrained supply of virtual currency) for said resources over time and space using combinatorial auction bids [27]. Unlike existing work that focuses on contention for a single resource (CPU cycles), they are motivated by scenarios where users express interest in 'slices' of heterogeneous goods (e.g. disk space, memory, bandwidth).

Chun et al. [10] propose a so-called microeconomic resource allocation scheme (using combinatorial auctions within a closed virtual currency environment) for sensornet testbed resource management, called **Mirage**. The operation of Mirage is very similar to that of Bellagio however it is a real-world deployed system, and observations of this system have revealed many users behaving strategically to exploit the system.

The **Tycoon** [20, 22] system is a market-based resource allocation system that utilises an Auction Share scheduling algorithm where users trade off their preferences for low latency (e.g. for a web server), high utilisation (e.g. batch computing) and risk.

Libra [34] is a computational economy-based scheduling system for clusters that focuses on improving the utility, and consequently the Quality of Service, delivered to users. Libra is intended to be implemented in the resource management and scheduling (RMS) logic of cluster computing systems, such as PBS [16].

FirstPrice and **FirstReward** are two utility-driven scheduling heuristics that balance the risk of future costs against the potential gains (reward) for accepting

Table 1: Summary of price setting and negotiation

System Name	Price Setting	Acceptance Criteria	Penalty
Bellagio [4]	Fixed	'Threshold'-based	N/A
Mirage [10]	Fixed	'Winner Determination Problem'	N/A
Tycoon [20, 22]	Fixed	First/Second Price	No
Libra	Fixed	Minimum Cost	No
Li [23]	Fixed	nth Price	No
FirstPrice [11]	Variable	None	No
FirstReward [19]	Variable	Risk versus reward	Yes
FirstProfit [31]	Variable	Risk versus per-job reward	Yes
FirstOpportunity [31]	Variable	Affect on profit	Yes
Aggregate Utility [5]	Variable	Contract feasibility / profitability	Yes

tasks [19]. The importance of using admission control in such schemes is also illustrated. Tasks in such systems are batch jobs that utilise resources (that are predicted accurately by the client) but provide no value until completion. Each job is characterised by a utility function that gives the task a value as a function of its completion time. This value reduces over time, and if unbounded can be a negative value (i.e. penalty) placed on the service provider. These systems do not model the currency economy or injection (unlike Bellagio and Mirage). Sealed bids are used and no price signals to buyers are assumed.

Popovici and Wilkes examine profit-based scheduling and admission control algorithms called **FirstProfit** and **FirstOpportunity** that consider a scenario where (grid) service providers rent resources from resource providers (rather than running and administering them), acting as a "middleman" [31]. Clients have jobs that need processing with price values (specifically a utility function) associated to them. The service provider rents resources from the resource provider at a cost, and price differential is the jobs profit that goes to the service provider. Complicating matters is the fact that resource uncertainty exists, as resources may be over-provisioned. If the service provider promises resources they cannot deliver, the clients QoS targets will not be met and the price they will pay will decline, as defined by the clients utility function. It is assumed that service providers have some domain expertise and can reasonably predict running times of jobs in advance. The accuracy of this estimator was modelled using a normal distribution.

3.1 Price setting and negotiation

In a market-driven distributed system, users express the value or 'utility' they place on successful and timely processing of their jobs through the process of price setting and negotiation. Users can attempt to have their job prioritised and allocated a greater proportion of resources by increasing the price they are willing to pay to a service provider for the privilege. Conversely, users can endeavour to save their currency when processing low priority tasks by specifying flexible deadlines. Such preferences are captured by the price setting and negotiation mechanisms of a given grid computing market. These price setting and negotiation approaches are presented below, and summarised in Table 1.

3.1.1 Fixed price models

The GRACE [8, 9] system does not specifically prescribe a particular price model, claiming it should be left up to each participant to define their own utility-maximising strategies when negotiating for access to resources. However, the interactions described in preliminary work on GRACE closely resemble a double auction [8], where 'asks' and 'bids' are traded that can incorporate costs and flexible deadlines (to trade-off against cost) until a fixed agreement is struck, or the negotiation is abandoned. Such double auctions were subsequently been found to be highly efficient by Placek and Buyya, and were integrated into Storage Exchange, a platform that allows organisations to treat storage as a tradeable resource [30].

Market inspired systems such as Bellagio [4] and Mirage [10] utilise second-price and first-price auctions respectively to schedule and allocate resources. In each case, users specify a fixed value bid for a set of resources over time and space. Auctions are held at fixed intervals (e.g. every hour for Bellagio), and users are either notified if they are successful (and are allocated the requested resources), or are denied access if unsuccessful. Users are encouraged to bid their true value in this approach, as they must wait until the next auction to try again, potentially with a modified bid to improve their chances of success. Users only have limited means to express the utility of their job, via the fixed price they are willing to pay, the duration of time they need the resources for and the deadline it must be completed by.

The Tycoon [20, 22] system similarly uses auctions, but removes the onus on the user to adjust their bids based on their success at auction. So called 'parent-agents' are controlled for each application a user wants to run, which manage the budget and the associated child-agents. A user specifies high level parameters (such as the number of credits available to that parent, the deadline and the number of hosts needed). A child agent initiates the bidding process, potentially liaising with

multiple providers. It monitors the progress of these negotiations and reports it up the chain. An auctioneer computes efficient first or second price bids for CPU slices based on the funds and information presented by the child nodes interacting with it. Funds can be received upon entry into the system, and also at regular intervals, but this element or its effect is not explored further.

Li et al. propose an iterative gradient climbing price setting approach to balance supply and demand in agent-driven grid marketplaces [23]. In this approach, the participants iteratively establish prices using a gradient climbing adaptation such that supply meets demand, using an nth price auction scheme. While each bid uses a fixed price, future bids are automatically modified based on current market conditions. If a price is rejected, an agent subsequently increases its bid; if it is accepted it reduces its bid. In this manner, Agents in the market act independently to maximise their own individual profits.

3.1.2 Variable price models

In the FirstPrice/FirstReward [19] and FirstProfit/FirstOpportunity [31] job scheduling systems, each users job is characterised by a utility function that gives the task a value as a function of its completion time. This value reduces over time, and if unbounded can be a negative value (i.e. penalty) placed on the service provider. Jobs provide no value to the user until they are completed. Each jobs deadline is equal to its minimum runtime, so any delay incurs an immediate cost. After this point, the compensation that the provider receives decays linearly as time progresses past the deadline.

AuYoung et al. proposes using aggregate utility functions in conjunction with individual utility functions for sets of jobs, described as a 'contract' [5]. An aggregate utility function is proposed in the form of $aggregate\_utility = \alpha x^{\beta}$. For a contract the overall payment is the sum of the per-job prices multiplied by the value of the aggregate utility function. The parameters can be changed to reflect the users preference in having a full set of tasks processed. When α is 1 and β is 0, the user is indifferent as to whether a partial of compete set is processed. α is 1 and β is 1 there is a linear relationship between the level of completeness in processing the set and the aggregate utility function. When α is 1 and β increases, the user places a higher dependency on the majority of the jobs completing in a set, penalising the provider if this is not the case. β controls the clients sensitivity to the aggregate metric a high β and α value would result in high risk for the service provider, but high reward if they can satisfy as much of a contract (i.e. set) as possible getting a "bonus".

Table 2: Summary of Utility-driven Resource Management Systems

System Name	Allocation	Scheduler	Adm. Control
Bellagio [4]	SHARE [12]	Proportional Share	N/A
Mirage [10]	Auction-based	Proportional Share	N/A
Tycoon [20, 22]	Auction-based	Auction Share	No
Libra	N/A	Proportional Share	Yes (basic)
Li [23]	Double Auction	Proportional Share	N/A
FirstPrice [11]	N/A	Highest Unit Value First	No
FirstReward [19]	N/A	Highest Unit Value First[1]	Yes
FirstProfit [31]	N/A	Highest Per-job Profit First	Yes
FirstOpportunity [31]	N/A	Highest Total Profit First	Yes
Aggregate Utility [5]	N/A	Highest Contract Profit First	Yes

3.2 Utility-driven Resource Management Systems (RMS)

Utility-driven resource management systems (RMS) actively consider the utility of participants when performing resource allocation, scheduling and admission control. Some well known utility-aware resource management systems are presented below, and summarised in Table 2.

3.2.1 Auction-based resource allocation

Bellagio: A 'second-price' style auction is employed in Bellagio [4] to encourage users to reveal the true value they place on these resources. The auctions are then held every hour using using SHARE [12] which allocates resources by clearing a combinatorial auction. Given that clearing such combinatorial auctions is known to be NP-Complete, SHARE utilises approximation algorithms to determine the winner. The Bellagio system assumes that an authentication entity exists that authenticates bids, resource capabilities (reservations) and account balances, such as SHARP [15]. The virtual currency economy is managed such that one user cannot dominate the available resources for lengthy periods of time. This currency system is explained in further detail in Section 3.3.

Experimental results show that Bellagio scales well to clearing auctions involving thousands of bids and resources. Bellagio generates more utility for its users than a standard proportional share resource allocation under a variety of load con-

[1] Risk-aware

ditions. Bellagio is especially effective under higher system load, spreading out resource requests in order to maximise overall utility. The fairness of the Bellagio policy (with regards to protecting light users against starvation from heavy users) can be ensured by using an appropriate currency distribution policy.

Mirage: The resource allocation problem for sensornet testbeds (such as Mirage [10]) is well suited to combinatorial auctions, as resources are both substitutes (specific machine allocation is often not important) and complimentary (partial request allocation is not useful, but full allocation are complementary to each other). Mirage can locate and reserve resources based on per-node attributes (e.g. a certain feature needed at each node) but not inter-node attributes (e.g. node is 10 metres from other node). The bid format language used by Mirage for users to express their preferences is based on XOR [27], allowing them to request resources over both space and time, with the valuation and deadline flexibility of their bid contributing to the probability of a successful bid.

While observing the system over a period of 4 months, the authors observed that the values users placed on resources varied over four orders of magnitude, validating the auction-based approach. Users also requested a range of resource allocations and duration's, highlighting the flexibility of the bidding language by allowing users to share access to the system where appropriate and improving utilisation. However, as the authors observed a live system, it was not compared against any baseline scheduler, so the increase in overall utility was not quantified.

Tycoon: The Tycoon [20, 22] system utilises an Auction Share scheduling algorithm where users trade off their preferences for low latency, high utilisation and risk - depending on their specific requirements. Many schedulers in similar systems utilise proportional share, where processes are allocated an interval, which is weighted by the importance of that process. As the authors note this method can be abused, as it depends on users truthfully valuing their process, and as the sum of weights increases each processors share approaches zero. The Tycoon approach encourages users to devise specialised strategies for each specific application they need to run, based on their needs. Tycoon utilises an agent-driven approach, which removes much of the complexity (and, we note, the control) in devising effective market-driven strategies for users processes.

When a market strategy is adapted to proportional scheduling, it performs well under high utilisation - equivalent to a best case proportional scheduling scenario where users value their jobs truthfully. When no market principles are used and users value their tasks strategically, utility converges to zero as the load increases.

An auction strategy attempts to provide the best elements of many approaches

high utilisation (proportional share), low latency (borrowed virtual time share) and low risk (resource reservation). Users specify their preference based on application-specific needs. A proportional share scheduler can only hope to achieve high utilisation or low latency / fairness, but not both, and users are not encouraged to value their tasks truthfully. The limited results demonstrate that an Auction share approach can be effective in ensuring predictable latency and also be fair by yielding the CPU when not needed.

Combinatorial Exchange: Schnizler et al. propose to formulate the resource trading and allocation problem as a multi-attribute combinatorial exchange model [33]. Like many other grid economy researchers, the authors claim that traditional (e.g. FCFS) allocation and scheduling are not sufficient in market driven computational grids, as they fail to capture the utility (i.e. true) valuation that users place on the successful processing of their jobs. Flat rate pricing schemes for grid economies are also insufficient, as they do not capture the variability in demand and user utility that exists. The authors state certain desirable properties that any pricing and allocation scheme should have in an economy driven grid:

- Allocative efficiency: Pareto efficiency dictates that any allocation mechanism must ensure that one agent is made better off without making at least one agent worse off. If user utility is equivalent and transferable, then a mechanism should maximise the sum of individual utilities.
- Incentive compatible: All participants must report their preferences (i.e. valuations) truthfully.
- Individual Rationality: Requires that users participating in a mechanism have a higher utility than before they joined - otherwise they have no incentive to participate.
- Budget balance: A mechanism is budget balanced if the amount of prices sum to zero over all participating agents. This can be achieved by a closed loop currency system where payments are redistributed among the agents, with no funds removed nor injected from outside. Weak balanced budget occurs when participants make payments to the mechanism, but not vice versa.
- Computational / Communication tractability: The cost in computing the outcome of allocating resources in an optimal or near optimal fashion must be considered, as well as the communication effort that is needed to converge on these goals.

The underlying environment must also consider the following domain specific requirements:

- Simultaneous trading: Support of multiple buyers and multiple sellers.
- Trading dependant resources: Buyers demand combinations ('bundles') of resources, and may bid on many such bundles at once though XOR bids.
- Support for multi-attribute resource: A single resource may have multiple resources that are of interest to a buyer (e.g. a HDD has capacity and access time)

.

Existing market-based approaches do not account for time nor quality constraints in the bidding and auction process. As such, the authors introduce a bidding language to express the auction process for a computation grid, capturing the pertinent elements of a users bid - such as the makeup and quality requested for a specific resource bundle, and the valuation they place on it. The problem is formulated as a generalisation of the combinatorial allocation problem, which is known to be NP-complete. The objective function attempts to maximise the surplus $V*$, which is the sum of the difference between the buyers valuations and the sellers reservation prices. A Vickrey-Clarke-Groves pricing mechanism is assumed, but is not the focus of the paper. The proposed approach is simulated, and its computational tractability is shown (by solving the auctions using the well known CPLEX solver), clearly demonstrating the optimisation problem does not scale. The simulation shows an exponential relationship between the numbers of orders and the CPU time needed to compute the results of the auction. A scenario with 140 bundles comprising 303 bids took nearly a minute to compute, highlighting the critical need for a faster approximate solution to clear auctions.

3.2.2 Scheduling

Libra: As well as submitting typical parameters required by batch computing job submission systems such as estimated runtime E, location of data sets and expected output, the Libra [34] system allows users to express more descriptive requirements by parameters such as deadline D and budget B. A deadline denotes when the user needs the results of the job by, and the budget denotes how much they are willing to pay to have the job completed by the deadline. However, the operation of Libra depends on several limiting assumptions; that only one centralised gateway is used to submit jobs to the cluster, that there must be no background jobs (the cluster nodes are dedicated), that CPU time is divisible and reservable,

and that the estimated runtime E is accurate. The RMS makes a simple minimum cost computation for a submitted job, to see if a user's budget B is sufficient to cover the cost, $C = \alpha * E + \beta * E/D$, where α and β are the coefficients. The α parameter captures the raw cost of the resources required and the duration they are needed for, whilst β denotes the incentive offered to the user for specifying an actual deadline. This ensures that a user is charged for the cluster hours it uses, regardless of deadline, whilst considering the user's flexibility regarding the processing deadline and compensating or penalising them accordingly. This provides a good mix of user and provider satisfaction. If the budget is sufficient, the provider will check whether a cluster node can feasibly meet the prescribed deadline, given its existing commitments. If no node can meet the deadline, the job is rejected. In this case, a user should try again later or try a more relaxed deadline. By doing this, Libra can guarantee an accepted job will be processed before its deadline, provided the runtime estimate E is accurate.

Libra's proportional share approach allowed more jobs to be completed by their deadline (and consequently less jobs rejected) for a variety of cluster and workload combinations. However, such proportion share scheduling is not appropriate for all workloads - specifically, memory intensive jobs would suffer due to excessive context switching. Furthermore, Libra's approach critically depends on the quality of the runtime estimation to schedule jobs and resources effectively. However, accurate runtime estimates cannot be guaranteed for many batch or cluster workloads [24, 36].

FirstPrice and FirstReward: FirstPrice [19] sorts and greedily schedules jobs based on a schedule that will maximise (per unit) return. FirstReward [19] considers both the per unit return for accepting a task and the risk of losing gains in the future. A Present Value (PV) approach is utilised that considers the yield countered by a discount_rate on future gains that could be made. A high discount_rate causes the system to discount (avoid) future gains and focus on short running quick profit jobs - a risk adverse strategy. Experiments demonstrated that a high discount_rate is appropriate when there is a large skew between job values in the job mix. The integration of opportunity costs for the value of the potential loss incurred by taking one job over another. Ideally, jobs should only be deferred (to chase extra profit) if they have have a low decay rate, even if they have a high unit gain. The reward metric is a tuneable metric that considers potential gain with opportunity costs, that can be weighted to control the degree of which the system considers expected gain. When penalties are bounded, it is beneficial to take some risk in order to gain more profit the heuristic biases against low value jobs, and the penalty is capped. When the penalty is uncapped, taking considerable risks on future gain begins to become

a poor strategy.

FirstOpportunity and FirstOpportunityRate: Popovici and Wilkes propose two profit-based scheduling algorithms [31]. The first of these is FirstOpportunity, which examines the effect of running a job on others in the queue. Builds a new schedule for pending workload by selecting each job in turn (using only the most profitable shape for that job) and uses FirstProfit to generate a schedule for the remaining jobs. Finally, the job is chosen that generates the schedule with the highest total profit. The second algorithm is FirstOpportunityRate; similar to FirstOpportunity but considers the job that produces the highest aggregate profit in proportion to the total schedule length.

When accurate resource information is available, FirstReward and FirstPrice have a higher acceptance rate and utilisation under low load, as they do not consider resource cost, only profit. FirstOpportunity and FirstOpportunityRate have the most accurate profit estimation (at admission control stage) under all shown load conditions as they consider existing jobs as well as the newly arriving job. Under more severe decay rates, FirstOpportunityRate has consistently better performance and accuracy, as it prioritises smaller, higher profit jobs. Under variable resource availability and price, FirstOpportunity and FirstOpportunityRate are the best policies as they consider resource costs as well as price, whereas other policies schedule on client utility or job size alone.

When uncertain resource information is available, FirstProfit was extended to consider profit reward versus probability of achieving that profit, the other policies are uncertainty-oblivious. FirstProfit admitted fewer jobs when uncertainty increased, reducing its risk exposure (i.e. accepting jobs then not meeting client targets). Profit rate was increased as FirstProfit took slightly more risk (20%)

Aggregate Utility: The use of aggregate utility functions in conjunction with traditional, single job utility functions is explored AuYoung et. al. [5]. These aggregate functions are used to reflect an end-user's preference for a "set" of jobs to be completed in or close to its entirety, and the resulting utility they place on such needs. These job sets or contracts are assumed to describe the individual jobs (number of jobs, size, arrival rates) and their utility accurately and are well behaved. A job's value is equal to its utility function at the moment it completes, or its maximum penalty if cancelled.

3.2.3 Admission control

FirstPrice and FirstReward: The use of admission control combined with FirstPrice and FirstReward [19] is valuable especially under conditions of high system

load. A decision can be made regarding whether it is beneficial to accept a task before the system accepts it by looking at how it integrates into the existing task mix. A slack metric is utilised that considers the amount of additional delay it can impose on a job before it becomes unprofitable to accept it. High slack jobs are desirable as they can be potentially rescheduled if a more profitable job comes along.

PositiveOpportunity: PositiveOpportunity [31] is the admission control component of FirstOpportunity and FirstOpportunityRate. PositiveOpportunity computes job schedule (based on matching scheduling policy) for all tasks including new job (with all possible job shapes for new job enumerated), then all tasks not including new job. The admission control algorithm then chooses most rewarding scheduling. If job is in said schedule, it is admitted

Aggregate Utility: Admission control can also be done for both contracts (sets of jobs) and individual jobs [5]. When new contracts arrive, feasibility and profitability tests are done to ensure it is worth a providers while. Admission control is also done at the per-job level with each jobs individual utility function. However, the utility function includes a bias that reflects the intrinsic value of the job when processed as part of the set it belongs to.

LibraRisk: The Libra RMS has some rudimentary admission control functionality [34]. If a job has insufficient budget to cover the cost of execution, it is rejected. If a job has a sufficient budget, but cannot be accommodated by a cluster node using its current deadline, it is also rejected. LibraRisk [38] addresses one of the key weaknesses of Libra by enhancing the admission control, considering delays caused by inaccurate runtime estimates. Each newly arriving job is examined to gauge the risk of that job causing a deadline to be exceeded (and a delay resulting) in the cluster. Specifically, the risk of delay is computed for each node in the cluster, and a job is only accepted if it can be accommodated on a node such that there is zero risk of delay. Experimental results showed in in the face of inaccurate user runtime estimates, LibraRisk accepted and completed more jobs than Libra, whilst maintaining better slowdown across a variety of workload conditions.

3.3 Managing the grid economy

Managing the grid economy effectively is crucial to the success of any market-driven grid computing system. Indeed, without some form of constraint and scarcity

in the economy there is little incentive for users to value and bid on their jobs truthfully. With unbounded currency, users would simply bid the largest amount possible with unrealistic deadlines regardless of the relative importance of their jobs, with no thought of rationing out their currency for later use. If all users bid in this fashion, the market-based approach becomes redundant and offers no benefit over traditional 'fair' scheduling.

There are two major considerations for managing the grid economy. The first is whether to use virtual or real currency as the means of incentive and compensation to participants in grid computing markets. The advantages and disadvantages of both approaches are discussed in Section 3.3.1. The second consideration is how the currency in question is managed - that is, how it is injected, taxed, controlled and dispersed. This is described in Section 3.3.2.

3.3.1 Virtual vs. Real Currency

A well-defined currency system is essential to ensure resources are efficiently shared and allocated in a grid computing marketplace. The use of virtual or real currencies each have their own advantages and disadvantages. Shneidman et al. [35] found that most deployed computational markets use virtual currency, due to its low risk and low stakes in case of mismanagement or abuse. However, they note that virtual currency has its drawbacks, due to requiring careful initial and ongoing management of the virtual currency economy, which is often ignored due to the low stakes involved. There is also a critical lack of liquidity and flexibility in virtual currencies, with users unable to 'cash-out' or transfer their credits to other systems easily, if at all.

These approaches tend to be most appropriate for single marketplaces, such as researchers from one organisation sharing a HPC cluster. In this instance, it is simply a means to regulate access, especially under high contention, and encourage socially responsible behaviour from users. Whilst they represent fairly simplistic marketplaces, currency still need to be managed appropriately for any benefit to be realised over traditional scheduling and resource allocation techniques.

The use of real currency is appealing for a number of reasons. In some computational marketplaces, users must contribute their own resources (e.g. CPU, disk, etc.) in order to be granted credits to access the wider shared resources. This can be prohibitive for some, may want shorter term access and do not wish to be permanent participants in a grid. Allowing these users to buy access using real currency can lower the bar for entry and increase overall utilisation and revenue for the system overall. Real currency formats (e.g. USD, Euro, etc.) are universally recognised and are easily transferable and exchanged, and are managed outside the scope of a grid marketplace, by linked free markets and respective government

policy. However, many of the same problems that exist in real-world economies apply to computational marketplaces. It is entirely possible for one user to be denied access to a system (effectively priced out of the market) by another user with more money. The distribution of wealth held by users will have a critical effect on the a computational grid's accessibility, affordability and utilisation. As such, careful consideration must be given when choosing between virtual and real currency systems. For a shared system, where the aim is equality of access, a real currency system may be inappropriate. For a profit-driven commercial grid marketplace, an economy based on real currency would likely be the most appealing option.

3.3.2 Currency management and dispersal

The Bellagio [4] system offers some rudimentary controls on the distribution of wealth in the virtual economy - such as when to inject currency into system and how much. This subsequently controls the share of resources received by participating users. The total amount of currency that can be accumulated by a given user is bounded to reduce chance of abuse. This avoids situations such as a user hoarding currency for lengthy periods of time, then dominating the available resources causing starvation.

The Mirage [10] system offers more sophisticated currency management features. Some of its novel features (over Bellagio) include a proportional share profit sharing, where proceeds from cleared auctions are distributed proportionally to idle users to accumulate ad-hoc credit, and a savings tax to address unbalanced usage patterns and avoid excessive accumulation of credit. Indeed, over time a users credit regresses back to a baseline amount. The central bank and economy is more sophisticated than Bellagio as users have standard currency as well as currency 'shares', which affect the proportion of currency distributed back to idle users (after an auction clears). Mirage also features a proportional savings tax imposed on the currency held by users, known as the 'use it or lose it' policy.

Irwin, Chase et al. propose a self-recharging virtual currency system for shared computing infrastructure such as test-beds like PlanetLab, Intel SensorWeb [17]. They aim to address the so-called 'tragedy of the commons' where individual interests compete against the common good of all participants. They are primarily motivated by its use in a Cereus-based system for service-oriented computing. The authors propose recycling currency through the economy, while artificially capping the rate of spending by users, seeking to avoid currency hoarding and starvation that can occur in other market-based systems. The economy critically depends on an accountable, third party verifiable 'claim' system such as SHARP [15] to manage currency and resources.

In this system, credits recharge after a fixed recharge time from the time they

are spent (or committed to a bid). This is in an effort to avoid hoarding and starvation, but as the authors note if the recharge time is too short, this system resembles a simple random lottery. If the recharge time is too long, it resembles a typical money economy with all the endemic problems typically associated. Credit for a user is capped at a fixed budget of c users cannot hold nor bid more than c credits at any given time.

The fundamental issue in this proposal is when to recharge credits spent on winning bids, as this will significantly affect user behaviour. Existing work recharges credits when the purchased contract has expired, encouraging users to bid for short-term gratification rather than biding their time and bidding into the future. A Cereus system enforces a fixed interval recharge that occurs after a fixed time r after the user commits credit to a bid, encouraging early bidding for auctions in the future. Once committed, these credits are unavailable to users - they cannot bid on concurrent auctions with money they dont have.

However, without results it is unclear if this approach actually maximises end-user utility. Consider a group of users, each with c credits bidding on a single auction. All users bid c credits, and bid early. In a Cereus system, the broker accepts all bids, and returns a proportional amount of resources to each end-user, effectively negating the useful features of market-based scheduling. We propose that a bounded, randomly distributed recharge time could potentially diffuse this group behaviour. It is also unclear how the recharging currency system interacts with multiple concurrent auctions, or with the length of contracts awarded to end-users.

3.3.3 Trust models and accounting

SHARP is a framework for secure resource management (resource discovery and allocation) for wide-area, decentralised networked computing platforms [15]. SHARP provides participants with 'claims' on shared computing resources, which are probabilistic (i.e. not guaranteed) promises over resources for a specified time period. These claims can be utilised, subdivided or on-sold at the whim of the claim holder. These claims form the basis of a decentralised resource peering, trading and bartering system, with each claim being authorised by a chain of self-signed delegations which is anchored in the site authority of the resource provider of that claim.

SHARP is particularly suited to shared networks that can partition its resources into easily divisible units (slices), such as virtual machines. A local scheduler at each resource has the ultimate responsibility for enforcing any successful resource allocation claim. The resource claims themselves are split into two phases. In the first phase, a service manager (acting on behalf of a user who needs resources) obtains a 'ticket', representing a *soft* claim, that represents a probabilistic claim

on a specific resource for a period of time. In the second phase, the ticket must be converted into a concrete reservation by contracting the resources site authority and requesting a 'lease'. A 'lease' is a hard claim which is guaranteed valid, except in the instance of a resource failure. The separation of the claims process into two phases allows the site authority to consider current conditions (e.g. load) when determining when to redeem the claim, how to redeem the claim or even to reject the claim outright.

Claims in SHARP may be oversubscribed in that the agent acting on behalf of a resource has issues more more tickets than it can support. This can improve resource utilisation by statistical multiplexing, but also means that the claims themselves are probabilistic, and do not guarantee access. The probability of a claim being successfully converted into a lease naturally depends on the level of oversubscription. In the event of a claim being rejected, the victim can seek recourse by presenting the rejected claim to the issuing agent, so that the responsible agent can compensate them. In the instance that a resource is undersubscribed, the resource claims are effectively hard reservations.

Of course, SHARP does not function as an entire resource management system. Particularly, the onus is on resource providers to manage and enforce allocations created and held by participants that were generated by SHARP in the form of tickets and leases. As SHARP is intended for use with systems without global trust, where resources claims can be arbitrarily divided and given or on-sold to anonymous third parties, security and abuse is a large problem. The extent of this problem is largely dependent on the platform SHARP is deployed on, and its ability to isolate users from one another, and to track prevent abuse. SHARP mitigates some of this risk by only allowing access (but not control) to slices of resources for fixed periods of time. If an abusive user is detected their access can be revoked. The specifics of this revocation are again left to the local resource managers themselves.

Motivated by the SHARP leasing framework, Irwin et al. extend this further with SHIRAKO, a generic and extensible system for on-demand leasing of shared network resources [18]. SHIRAKO brings dynamic logic to the leasing process, allowing users to lease groups of resources from multiple providers over multiple physical sites, through the services of resource brokers. SHIRAKO enhances the functionality of SHARP by adapting to dynamic resource availability and changing load conditions through autonomous behaviour of the 'actors' in the system. In particular, SHIRAKO allows 'flexible' resource allocation through leases which can be re-negotiated and extended via mutual agreement, which was not possible using SHARP alone. This removes the need to obtain a new lease for any residual demand that exists once an existing lease expires, reducing resource fragmentation and continuity. Additional logic is provided for making resource leases more flex-

ible for users. A request can be defined as 'elastic' to specify a user will accept fewer resources if its full allocation is not available. Requests can be 'deferrable' if a user will accept a later start time than what is specified in the lease if that time is unavailable. Request groups are also defined, which can satisfy a users need for co-scheduling. A set of tickets can be assigned to a request group, ensuring (where possible) that each request is satisfied within a common time window.

4 Catallaxy market architectures

In the last 3 years there has been an increased research focus on the notion of applying Austrian economist F.A. von Hayek's notion of a 'Catallaxy' and applying it to market-driven grid computing. The idea of 'Catallactics' considers markets where prices evolve from the actions of economically self-interested participants. Each participant tries to maximise their own gain whilst having limited information available to them.

Eymann et al. have investigated the issues and requirements of implementing an electronic grid market based on the concept of 'Catallaxy', a 'free market' economic self-organisation approach [13]. However, they found that solving this problem is a complex multi-attribute allocation problem, found to be NP-complete in previous work by other researchers in the field. The authors note that in interrelated markets, the allocation of resources and services in one market invariably influences the outcomes in the other market. These interactions should occur without relying on a traditional centralised broker. Participants should be self-organising and follow their own interest, maximising their own utility. A catallaxy approach works on the principle that there are autonomous decentralised agents, which have constant negotiation and price signalling occurring between them. Indeed, changing conditions (availability, competition) on the resource market will be reflected by cascading price changes that reflect the respective scarcity and demand for a resource. Participants must read these signals and react accordingly.

The principles of the catallaxy as applied to computation grids are stated by the authors as follows:

1. Participants work for their own self-interest; each element is a utilisation maximising entity.

2. Entities do not have global knowledge; they can only act on information as it is made available to them. They must adapt to constantly changing signals from downstream and upstream entities.

3. The market is simply a communications bus; price changes (e.g. increases

and decreases) will dictate whether an entity looks for alternative sources to procure a specific resource, making the market dynamic.

A prototype model for the 'Catallaxy' was presented by the authors, consisting of an application layer and a service layer. In the application layer, complex services are mapped to basic services. The service layer maps service requests to actual resources provided by local resource managers.

The potential scenarios for existing grid marketplaces at the service layer typically fall into three categories; Contracting resources in advance, contracting resources after finalising the service contract with client and contracting resources during negotiation.

When contracting resources in advance, future demand must be forecast. It is centralised and static solution, and can lead to poor utilisation. Contracting resources after finalising the service contract with client can be risky, as insufficient resources could be available. Contracting resources during negotiation is highly desirable, as a user can choose between multiple providers based on cost, and can also adapt to changing prices, reducing risk and ensuring the market is balanced. The authors use the latter approach to form the basis of Catallaxy inspired grid market.

The design of the service discovery mechanism is also critical. Eymann et al. claim that centralised registries are not appropriate due to the decentralised nature of buyers and sellers. Distributed Hash Tables may be suitable but lack scalability in dynamic networks due to the state constantly changing, leading to high communication and computational overhead.

The authors note that the choice of auction strategy used is also important in a 'Catallaxy' driven grid marketplace. These can include fixed prices, Dutch auctions, English auctions and Double auctions. When buyers and sellers set fixed prices, if they are both in the 'closure' zone, a deal is reached. In a Dutch auction, only the seller performs concessions, and the buyer remains at start price. Under an English auction, the buyer performs concessions, and the seller remains at start price. In a Double auction, both agents get closer after each negotiation step. However, we note that any auction approach chosen must be appropriate for the scale and dynamism of linked 'Catallaxy' inspired grid marketplace, completing rapidly with minimal computation and communication overhead.

Ardaiz et al. explore a decentralised, economic and utility-driven resource allocation approach for resource allocation in a grid marketplace [3]. In such environments, each participant (client, service providers and resource providers) independently try and maximise there own utility by following a self-interested strategy. A decentralised approach is evaluated for grids of various densities and reliability's, and is compared against a centralised, statically optimal approach. In effect

there are two markets operating one for resources (where service providers procure resources from resource providers), and one for services (where clients procure services from service providers). As such, the client is not aware of the particulars of the resource provider, and vice versa. However, the service provider participates and tries to maximise utility in both markets.

An experimental framework is presented in order to simulate varying grid markets under realistic settings with a topology following that of the Internet. There are highly connected, high bandwidth and compute power centres (e.g. HPC facilities) and also smaller resources available toward the regional centres and edges of the system. The various strategies and tendencies for negotiation in the two markets are reflected probabilistically from a previous study of the catallaxy. The range valuations and pricing limits that participants place on their negations are very narrow, which is inconsistent with previous observations in deployed grid marketplaces, where 4 orders of magnitude difference has been observed in valuation, with little correlation to the service requirements needed. Each request unit also requires an equal amount of computation and bandwidth power, and does not reflect the mix of data and compute intensive jobs that are found in grid computing workloads.

Under high load, high density grids, total utility and resource efficiency decrease for the baseline and distributed approaches, due to difficulties in making successful allocations caused by high contention for resources. Response time is worse for the distributed approach due to the flooding discovery mechanism used to find resources. As grid density increases it takes longer for the distributed catallactic approach to find and allocate resources, as the discovery time increases due to resources being situated on the edges of the network. A small increase is seen in the baseline approach, due to the need to monitor a larger number of resources. As the network becomes more dynamic and prone to failure, the catallactic approach shows better response time, as it is more resistant to poor allocation choices and can re-negotiate with other resources in place of failed resources.

5 Mercato: A 'Catallaxy' inspired architecture for utility computing

This section proposes a decentralised architecture, called Mercato, for utility computing based on market mechanisms. Mercato builds on the work of systems discussed in the previous sections to enable an architecture where different entities adopt interactions that enable them to improve their own utilities. Mercato uses the Catallaxy paradigm as the basis of a decentralised framework where every participant seeks to maximise their own utility and thereby, improve the utility of the

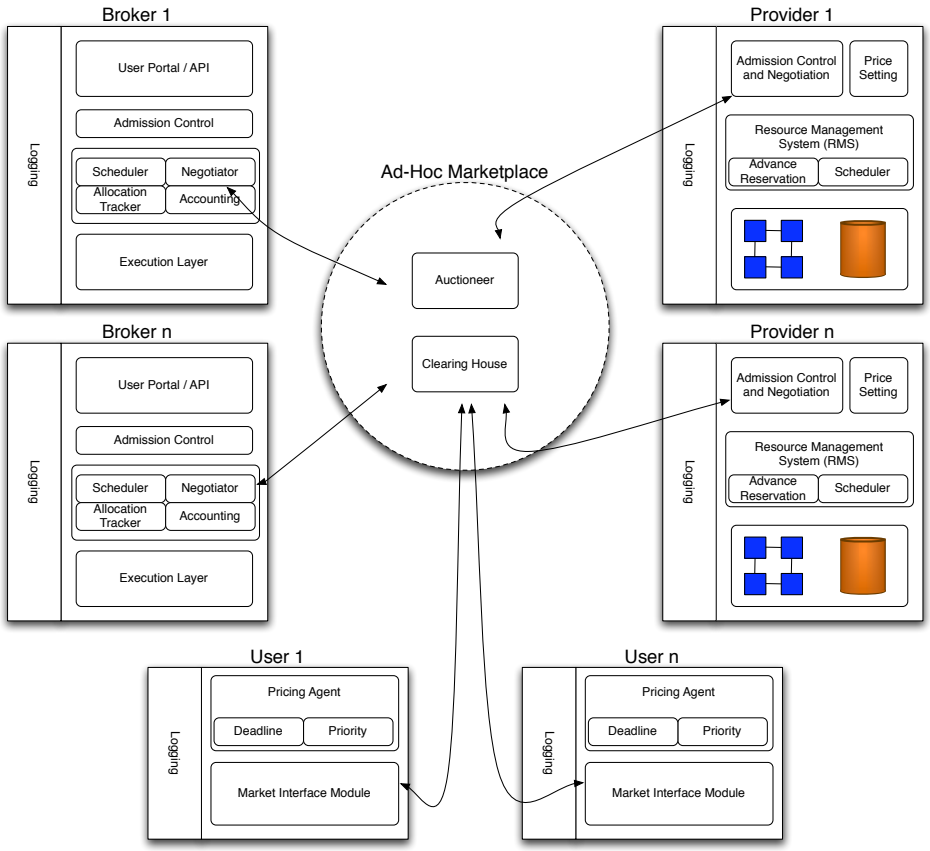

Figure 1: Mercato Architecture

entire system. However, unlike other systems surveyed in the previous section, it does not impose nor propose specific market mechanisms for enabling the interactions within the architecture. In this manner, the architecture reflects real-world market economies in which different participants interact with each other using mechanisms of their choosing.

As presented in Section 2.1, there are three main kinds of participants in Mercato: users, brokers and providers. *Users* aim to have their own utility functions that cover factors such as deadlines for executions and fidelity of results. They are also constrained by the amount of resources that they can request at any time, usually through a limited budget. *Providers* allocate shares of computational, data or network resources to different users or brokers based on the gain in their utility The service providers could maximise their utility either by maximising resource utilisation (e.g. research computing centres) and/or by maximising profit from resource leases (commercial resource providers). *Brokers* mediate between the users and the service providers. Brokers obtain resource shares from different providers that they then resell to the users. A broker can accept requests from many users who have a choice of submitting their requirements to different brokers. Brokers gain their utility through the difference between the utility given to the users and that obtained from the providers. The users, brokers and providers are bound to their requirements and related compensations through Service Level Agreements (SLAs). An SLA specifies the details of the service expected to be provided in terms of metrics agreed upon by all parties, and the rewards or penalties for meeting or violating the expectations, respectively. Figure 1 shows the participants and the interactions between them in detail.

As presented previously, Eyemann, et. al. [13, 3] have proposed a Catallactic architecture with two markets, one for resources and the other for services. The service providers can be mapped to the brokers in Mercato. However, we propose that interactions between participants be carried out through *ad hoc markets*, markets are created through the interactions of different components: i.e. by the users, brokers and providers. These markets can be simple such as a broker and provider coming to an agreement or they can be complex such as an resource exchange in which bids from the brokers and providers are cleared at regular intervals. Thus, by using ad hoc markets, our architecture differs significantly from those surveyed previously.

5.1 Providers

A resource provider contains the following sub-systems:

Price Setting Mechanisms Sets the current price for the resource based on market conditions, user demand and current level of utilisation of the resource. Pricing can be either fixed or variable

Admission Control and Negotiation Decides based on brokers' proposals, which of them are worth negotiating and which are not (based on an initial estimate of the utility that the proposal provides). Once a broker is accepted, then the negotiation proceeds until a deal is reached or the participants break off. At the end of the process, the participants agree to a binding SLA.

Resource Management System The resource management system performs job scheduling and allocation of nodes. It has an Advance Reservation component that can identify nodes (or equivalent resource units) to be reserved in advance and can ensure that they are reserved through a transaction-oriented mechanism. The AR is backed by the pervasive Logging component in this regard.

5.2 Brokers

Brokers mediate between users and providers and resolve users' requirements into SLAs with the providers. Traditionally the Gridbus broker has been a user agent, wherein it will discover and schedule jobs on to resources to satisfy given application and QoS requirements (only deadline and budget supported). The broker's utility is also tied to the users' utility. In Mercato however, the broker has its own utility function which is separate from that of the users. This utility comes from the difference between what the broker pays the provider for the resource shares and what the user pays the broker for executing the application. The broker therefore, supports multiple users and selects only those users whose applications can provide it maximum utility. This draws on previous work in Sharp [15], Tycoon [22] and Shirako [18] projects.

A broker therefore consists of the following components:

User Interface/ API is the primary interface for users to submit their applications and QoS requirements. The interface will support an API for user agents to talk to the broker.

Admission Control Since the broker is interested in maximising its own utility, the admission control component determines which users to accept based on their potential to provide better utility than the others that have been rejected. (Expand role)

Resource Allocation and Management This component consists of four functional sub-components.

- Scheduler - schedules users' jobs based on the available resource shares, the capability of the resources, the cost of using the shares and the utility derived from the user. The role of this component is essentially to meta-schedule over resource shares and therefore, most of the problems will be knapsack-type.
- Negotiation Module - interacts with resource providers and other brokers to gain or to trade resource shares. The negotiation module is informed by the current conditions of the resources and the current demand to make its decisions.
- Allocation Tracker - keeps track of the shares that have been gained by the broker from the resource providers or from other brokers. (Expand role)
- Accounting Mechanism - keeps track of the shares used by the users

Execution Layer Runs the users' application on the selected resources either by starting/requesting a virtual machine instance or by submitting jobs (follows mechanism within the Gridbus broker for instance)

Logging Logs all the transactions to provide audit trails and recovery capabilities.

5.3 Users

Users in Mercato attempt to gain utility by ensuring that their service demands are met at the least cost possible. Users participate in the market(s) by interacting directly with providers or by selecting a broker to fulfil their requirements. Using brokers may ensure that the users are able to access resources through preferential agreements negotiated by the former with the providers. A software program acting on behalf of the user, also called *user agent*, may have the following functions embodied in it:

- Pricing Agent - The pricing agent receives input about the current price levels of resources from markets, brokers and/or providers. It also takes into account the *deadline* to complete the user's job and the *priority* accorded to it to compute an internal valuation of the application. Based on these inputs, the pricing agent computes the budget that is required to achieve the

user's requirements. It can also advise the user, based on its estimates, as to whether the deadline is too constrained or if the budget is lacking.

- Market Interface Module - This component discovers and negotiates with the brokers and providers who may be able to execute the user's application within the given deadline. The negotiations can be conducted through market mechanisms such as auctions or bargaining. The market interface uses the pricing agent functions to identify the best deals for the users

6 Discussion

Based on our study of existing utility-driven and market based distributed computing architectures, we proposed a general purpose, ad-hoc utility computing framework called 'Mercato' in Section 5. The functionality of the Mercato framework was inspired by the emerging 'Catallaxy' paradigm, where multiple linked computing marketplaces, each consisting of self-interested, utility-maximising entities, interact and evolve depending on market conditions. Participants in these decentralised systems are constantly adapting to current conditions depending on the market signals.

In this section we discuss how Mercato intends to address many of the limitations of existing utility computing systems, and complements the 'Catallaxy' view of the next generation of computing marketplaces.

Many existing systems (such as Bellagio [4], Mirage [10], etc.) have restrictive price setting and negotiation policies. Auctions are held at fixed intervals, and only one type of auction is allowed (e.g. First Price, Second Price). Like GRACE [8, 9] in Mercato we leave the choice of negotiation and pricing protocols up to each participant in the system. This is crucial as the choice of pricing (fixed, variable) and negotiation protocol (auction, one-to-one, agent driven, etc.), with the permutations and combinations thereof having an enormous effect on the utility gained by the participants, depending on the current market conditions. Issues such as resource scarcity (or glut), economic conditions, time constraints, and the number of participants (i.e. competition) will result in very different results for each combination of pricing and negotiation.

Management of the computational economy becomes considerably more complicated when many disparate resource providers are linked together in the same "marketplace". Most of the systems studied in this paper are closed economies where money is distributed and recycled through the system. The currency is typically virtual, not real, and as such have the associated strengths and weaknesses described in Section 3.3.1. Most notably, virtual currencies lack liquidity and flexibility, but offer low stakes and low risk for service providers in the event of abuse.

To utilise these systems, users must *earn* credit through either in-kind contribution of resources, or via periods of inactivity and judicious use of the available services. Unfortunately, this can raise the barrier of entry, making it unsuitable for a user who needs to access the service immediately, despite the fact they could be willing to financially compensate a service provider if the option was available.

Many commercial service providers offer their resources in exchange for real currency (e.g USD, Euro, etc). In these system the currency market is open, well defined and liquid, and addresses some of the access issues that plague virtual currency systems. Subject to availability, users can simply 'buy in' to get access to a providers' resources. Conversely, systems utilising real currency can face many of the same issues that occur in real marketplaces, where access to systems can be determined by who has the most money, and the distribution of wealth held by users can be heavily skewed.

The Mercato broker is intended to broker interaction and interoperation between these two very different service paradigms. It is not appropriate to simply dictate that one style of currency is used. For many distributed systems, grids and testbeds, the intent of the system is to provide equitable access for as many people as possible. These systems may have been created with government or grant funds for greater social benefit, and as such any attempt to profit from these systems would be inappropriate, beyond basic cost recovery. In this instance, the constrained virtual currency is simply a means to regulate access to the system. For commercial providers, profit is the overriding motive, so virtual currency is not appropriate means of compensation for the services it offers.

The broker in an ad-hoc Mercato marketplace acts as a 'middleman' between the user and one or many service providers. These providers can be either commercial or open access in nature. The broker is essentially a value-added service, aggregating access to multiple providers that may not necessarily be accessible to an end-user. In the case of open access systems that utilise virtual currency that are tied to a particular system, a broker can 'earn' access for later use, via in-kind contribution of resources, or simply accumulating credit by joining the system and remaining idle. The broker can then utilise this credit when end-users request access to these systems, removing the onus on the user to join or contribute to that particular system and giving them immediate access. Such access could be procured from a broker for real currency (where the cost is dictated by the broker), or by mutual agreement a user could offer virtual credit at a different computing facility in exchange for access to the service provider offered by the broker. For instance, we could envisage a scenario where a user offers a broker X access credits from Mirage for Y access credits at Bellagio (depending on the relative worth, negotiated by the user and the broker). Such trading arrangements could also be made between different brokers in a marketplace.

In Mercato, resource reservation, allocation and service guarantees largely depend on the service provider. The service provider can choose whether it offers 'hard' or 'soft' guarantees, and whether compensation is given for missed deadlines or lack of service. A broker can then decide if it is satisfied with these guarantees when reselling the services of a provider. In the event that a service provider continually fails to meet agreed quality of service targets, a broker can seek compensation (if available) or simply terminate it's arrangement with that provider, and find another. The broker itself could potentially over-subscribe access to a given service provider by selling more resources than it has acquired or reserved from a provider. This can be risky as a provider (which has it's own resource allocation and admission control policies) can simply refuse a request for access to resources, leaving the broker liable for over-promising on resources it does not have. In this instance, a user could seek compensation from a broker (if available), or simply find another, more reliable system.

A reputation system would also complement systems like Mercato, where continual offenders could earn a poor reputation, allowing others to avoid them where possible. Conversely, good service could be rewarding with a positive reputation, attracting more customers for service providers and brokers alike. Participants (i.e. users, brokers and service providers) could track this individually, based on their own experiences, or through a co-ordinated, decentralised registry.

As discussed in Section 2.3, the emergence of widely available, commodity virtual machine technology has simplified administration and allocation of resources for service providers. Virtual machine infrastructure such as Xen [6] and VMWare [1] have allowed these providers to partition resources such as memory, disk space and processor cores into virtual machines with arbitrary capabilities, with minimal overhead. This can allow greater efficiencies for providers through improved statistical multiplexing when hosting multiple virtual machines.

Despite this, brokers can still play an important role in VM-driven computing marketplaces. They can provide numerous value-added services, such as providing pre-made virtual machine images for common tasks, or base images for users to trivially build upon and add their own application logic. A user could utilise such base images to create their own custom, self-contained container for execution, along with any data, libraries and applications that are needed for the duration of execution. This removes a significant amount of complexity for users, removing the need for them to ensure the relevant data, libraries and applications are available on the target execution environment hosted by a service provider.

7 Conclusion

In this work, we have evaluated the state of the art in market-drive utility computing platforms. In Section 2 we provided an overview of the key components of these platforms, identifying the roles and responsibilities of the participants, the effect of market-based techniques and the emerging technological advances that are propelling market-driven utility computing platforms forward. In Section 3 we examined the state of the art in utility-oriented computing markets, identifying the strengths and weakness of current related systems and approaches to pricing, negotiation, resource and economy management. The Catallaxy approach to computing marketplaces was highlighted in Section 4, showing the benefits of this new decentralised, utility maximising framework in addressing the utility computing problem. Motivated by this, and our prior survey, the Mercato framework was presented in Section 5. This flexible, ad-hoc utility computing framework seeks to address many of issues highlighted in our survey. In Section 6 we discussed how Mercato addresses the limitations of many existing utility computing systems, and identify areas that need further attention for these approaches to flourish.

References

[1] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ASPLOS '06: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 2–13, New York, NY, USA, 2006. ACM Press.

[2] Amazon.com, Inc. Amazon Elastic Compute Cloud (Amazon EC2), 2007. Available at http://aws.amazon.com/ec2.

[3] Oscar Ardaiz, Pau Artigas, Torsten Eymann, Felix Freitag, Leandro Navarro, and Michael Reinicke. The catallaxy approach for decentralized economic-based allocation in grid resource and service markets. *Applied Intelligence*, 25(2):131–145, 2006.

[4] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *OASIS '04: Proceedings of the 1st Workshop on Operating System and Architectural Support for the Ondemand IT InfraStructure*, October 2004.

[5] A AuYoung, L. Grit, J. Wiener, and J. Wilkes. Service contracts and aggregate utility functions. In *HPDC '06: Proceedings of the 15th IEEE International*

Symposium on High Performance Distributed Computing, pages 119–131, June 2006.

[6] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[7] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating system support for planetary-scale network services. In *NSDI '04: Proceedings of the Symposium on Networked Systems Design and Implementation*, Mar 2004.

[8] R. Buyya, D. Abramson, and J. Giddy. Economy driven resource management architecture for computational power grids. In *PDPTA '00: International Conference on Parallel and Distributed Processing Techniques and Applications*, 2000.

[9] Rajkumar Buyya, David Abramson, and Srikumar Venugopal. The grid economy. *Proceedings of the IEEE*, 93(3):698–714, March 2005.

[10] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D.C. Parkes, J. Shneidman, A.C. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *EMNETS '05: Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, May 2005.

[11] Brent N. Chun and David E. Culler. User-centric performance analysis of market-based cluster batch schedulers. In *CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 30, Washington, DC, USA, 2002. IEEE Computer Society.

[12] Brent N. Chun, Chaki Ng, Jeannie Albrecht, David C. Parkes, and Amin Vahdat. Computational resource exchanges for distributed resource allocation, 2004. Unpublished manuscript, available at http://www.eecs.harvard.edu/c̆haki/doc/share04.pdf.

[13] Torsten Eymann, Michael Reinicke, Werner Streitberger, Omer Rana, Liviu Joita, Dirk Neumann, Bjorn Schnizler, Daniel Veit, Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, Michele Catalano, Mauro Gallegati, Gianfranco Giulioni, Ruben Carvajal Schiaffino, and Floriano Zini. Catallaxy-based grid markets. *Multiagent Grid Systems*, 1(4):297–307, 2005.

[14] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 2001.

[15] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: an architecture for secure resource peering. *SIGOPS Operating Systems Review*, 37(5):133–148, 2003.

[16] Robert L. Henderson. Job scheduling under the portable batch system. In *IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 279–294, London, UK, 1995. Springer-Verlag.

[17] David Irwin, Jeff Chase, Laura Grit, and Aydan Yumerefendi. Self-recharging virtual currency. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 93–98, New York, NY, USA, 2005. ACM Press.

[18] David E. Irwin, Jeffrey S. Chase, Laura E. Grit, Aydan R. Yumerefendi, David Becker, and Ken Yocum. Sharing networked resources with brokered leases. In *USENIX '06: Proceedings of the USENIX Annual Technical Conference, General Track*, pages 199–212, 2006.

[19] David E. Irwin, Laura E. Grit, and Jeffrey S. Chase. Balancing risk and reward in a market-based task service. In *HPDC '04: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pages 160–169, Washington, DC, USA, 2004. IEEE Computer Society.

[20] Bernardo A. Huberman Kevin Lai and Leslie Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical Report arXiv:cs.DC/0404013, HP Labs, Palo Alto, CA, USA, April 2004.

[21] Kevin Lai. Markets are dead, long live markets. *SIGecom Exch.*, 5(4):1–10, 2005.

[22] Kevin Lai, Lars Rasmusson, Eytan Adar, Li Zhang, and Bernardo A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent Grid Systems*, 1(3):169–182, 2005.

[23] Chunlin Li, Layuan Li, and Zhengding Lu. Utility driven dynamic resource allocation using competitive markets in computational grid. *Advances in Engineering Software*, 36(6):425–434, 2005.

[24] Ahuva W. Mu'alem and Dror G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfill-

ing. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.

[25] Michael Nelson, Beng-Hong Lim, and Greg Hutchins. Fast transparent migration for virtual machines. In *USENIX '05: Proceedings of the 2005 USENIX Annual Technical Conference*, pages 391–394, 2005.

[26] Chaki Ng, Philip Buonadonna, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. Addressing strategic behavior in a deployed microeconomic resource allocator. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 99–104, New York, NY, USA, 2005. ACM Press.

[27] Noam Nisan. Bidding and allocation in combinatorial auctions. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[28] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery. In *HPDC '05: Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing*, Washington, DC, USA, 2005. IEEE Computer Society.

[29] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the internet. *SIGCOMM Computer Communication Review*, 33(1):59–64, 2003.

[30] Martin Placek and Rajkumar Buyya. Storage exchange: A global trading platform for storage services. In *EUROPAR '06: Proceedings of the 12th International European Parallel Computing Conference*. Springer-Verlag, Aug 2006.

[31] Florentina I. Popovici and John Wilkes. Profitable services in an uncertain world. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 36, Washington, DC, USA, 2005. IEEE Computer Society.

[32] Evan Sarmiento. Securing FreeBSD using jail. *Sys Admin*, 10(5):31–37, 2001.

[33] B. Schnizler, D. Neumann, D. Veit, and C. Weinhardt. A multiattribute combinatorial exchange for trading grid resources. In *RSEEM '05: Proceedings of the 12th Research Symposium on Emerging Electronic Markets*, 2005.

[34] Jahanzeb Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat, and Rajkumar Buyya. Libra: a computational economy-based job scheduling system for clusters. *Software Practice and Experience*, 34(6):573–590, 2004.

[35] Jeffrey Shneidman, Chaki Ng, David C. Parkes, Alvin AuYoung, Alex C. Snoeren, Amin Vahdat, and Brent N. Chun. Why markets could (but don't currently) solve resource allocation problems in systems. In *USENIX '05: Proceedings of the 10th USENIX Workshop on Hot Topics in Operating Systems*, June 2005.

[36] Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson. Modeling user runtime estimates. In *JSSPP '05: Proceedings of the 11th International Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–35, Cambridge, MA, USA, 2005.

[37] Carl A. Waldspurger. Memory resource management in vmware esx server. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 181–194, New York, NY, USA, 2002. ACM Press.

[38] Chee Shin Yeo and Rajkumar Buyya. Managing risk of inaccurate runtime estimates for deadline constrained job admission control in clusters. In *ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing*, pages 451–458, Washington, DC, USA, 2006. IEEE Computer Society.

[39] Chee Shin Yeo and Rajkumar Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software Practice and Experience*, 36(13):1381–1419, 2006.

[40] Chee Shin Yeo, Rajkumar Buyya, Marcos Dias de Assuncao, Jia Yu, Anthony Sulistio, Srikumar Venugopal, and Martin Placek. *The Handbook of Computer Networks*, chapter Utility Computing on Global Grids. John Wiley & Sons, New York, USA, 2007.