Future Generation Computer Systems 27 (2011) 1–9

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An evaluation of the benefits of fine-grained value-based scheduling on general purpose clusters

Ruben Van den Bossche *, Kurt Vanmechelen, Jan Broeckhove

Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium

ARTICLE INFO

Article history:
Received 4 May 2010
Received in revised form
17 June 2010
Accepted 28 June 2010
Available online 6 July 2010

Keywords:
Scheduling
Simulation
Distributed systems

ABSTRACT

General purpose compute clusters are used by a wide range of organizations to deliver the necessary computational power for their processes. In order to manage the shared use of such clusters, scheduling policies are installed to determine if and when the jobs submitted to the cluster are executed. Value-based scheduling policies differ from other policies in that they allow users to communicate the value of their computation to the scheduling mechanism. The design of market mechanisms whereby users are able to bid for resources in a fine-grained manner has proven to be an attractive means to implement such policies. In the clearing phase of the mechanism, supply and demand for resources are matched in pursuit of a value-maximizing job schedule and resource prices are dynamically adjusted to the level of excess demand in the system. Despite their success in simulations and research literature, such fine-grained value-based scheduling policies have been rarely used in practice as they are often considered too fragile, too onerous for end-users to work with, and difficult to implement. A coarse-grained form of value-based scheduling that mitigates the aforementioned disadvantages involves the installation of a priority queuing system with fixed costs per queue. At present, however, it is unclear whether such a coarse-grained policy underperforms in value realization when compared to fine-grained scheduling through auctions, and if so, to what extent. Using workload traces of general purpose clusters we make the comparison and investigate under which conditions efficiency can be gained with the fine-grained policy.

1. Introduction

In many organizations, clusters fulfill the ever increasing need for computational power and data storage. Their scaling potential and performance have ensured that clusters have remained an important part of an organization's IT infrastructure, despite the advances in the performance of personal computers. As a consequence of their scale and cost, clusters are typically shared by a number of users whose resource requirements vary over time. The possibilities for parallel execution on these systems, combined with their ability to multiplex and enqueue user workloads, allow them to deliver high performance under a high level of system utilization. The key software component that determines the *efficiency* under which such clusters operate is the job scheduling system. Efficiency can hereby be expressed as a function of a wide variety of metrics such as system utilization, job turnaround times or job throughput.

In many clusters, job scheduling has long allowed for a prioritization of jobs through the definition of a discrete number of job queues. A job that is submitted to a high priority queue can thereby gain precedence over jobs in lower priority queues. The right to submit a job to a particular queue can be constrained by for example the user's priority level or the (user-estimated) runtime of the job. In that case, the highest priority queues are typically only available to jobs with relatively short runtimes, in order to avoid small jobs to be delayed by a few very long-running jobs.

In recent years, a renewed user-oriented view on efficiency has fueled the development of job scheduling systems that attempt to directly take the *value* that a user attributes to the completion of its computation into account [1–9]. Job scheduling systems that adhere to such a value-oriented efficiency goal are termed *utility-based* or *value-based* scheduling systems. In value-based scheduling approaches, one allows for the direct expression of user value, irrespective of any constraints on job runtimes or other job characteristics. In order to discourage users to consistently express the highest possible value for their jobs, an accounting system charges users according to the service level they have obtained.

We distinguish between two different ways for users to express their valuations. In a *coarse-grained* value-based scheduling policy, users can signal one of a few discrete, predefined values to the scheduling mechanism. An example implementation of such a policy is a *priority queue system* in which a particular charging rate is associated with each queue. On the other hand, a *fine-grained* value-based scheduling policy allows users to express their

* Corresponding author. Tel.: +32 3 265 38 57; fax: +32 3 265 32 04.
E-mail address: ruben.vandenbossche@ua.ac.be (R. Van den Bossche).

0167-739X/$ – see front matter
doi:10.1016/j.future.2010.06.009

valuation within a continuum. This allows the scheduler to enforce a more precise prioritization on the executed jobs.

Many works have investigated economically inspired approaches to job scheduling in which market mechanisms are used that allow a user to express his valuation in a fine-grained manner [10–15]. An advantage of such an approach is that the service cost can be set dynamically depending on the level of congestion in the cluster and the value that other users attribute to their jobs. In this manner, supply and demand for resources can be balanced, leading to optimal economic efficiency. This level of efficiency can be attained without the need for an administrator to manually intervene and, for example, configure the queue prices in a priority queuing system.

A disadvantage of fine-grained value-based schemes is that users are now asked to formulate their valuation as a value in a continuum. These valuations are often dependent on the expected completion times of the job. This is a non-trivial task that can be considered time consuming, especially if there are possibilities for a user to obtain a more attractive service level to cost ratio, by acting strategically. Although the installment of *(pseudo)-incentive compatible* [16] market mechanisms can remove the options for such strategic behavior, value elicitation in a non-strategic setting can remain a user burden.

Many studies have shown the efficiency gains, in terms of generated user value, that value-centric scheduling approaches can bring compared to traditional system-centric scheduling approaches such as First-Come First-Serve (FCFS) or Round-Robin (RR) scheduling [1,7,17–20]. A round-robin scheduler assigns equal time slices to each job in the system. However, only a limited number of studies have compared the performance of a fine-grained value-based scheduling system with a coarse-grained system that is based on priority queues. We have found that when such comparisons are made, insufficient attention is often given to the specificity of the experimental parameters that lead to high efficiency gains for the fine-grained approaches. The goal of this paper is to present such a comparison and pinpoint in which settings fine-grained user valuations can bring significant efficiency gains. Our additional aim is to evaluate to what extent such conditions for increased efficiency are fulfilled in practice, based on real-world cluster workload traces.

2. Related work

Chun et al. [3] compare a market-based algorithm called *FirstPrice* with a three-queue priority system called *PrioFIFO*. They model synthetic workloads on a small cluster consisting of 32 nodes, and combine time-varying bids with static bids. Their study shows that for sequential jobs, coarse-grained static valuations are just as effective as fine-grained, time-varying valuations. As parallelism in the workload increases, so do the benefits of the fine-grained time-varying approach. Chun et al. report increases in realized utility up to 250% for highly parallel workloads. We take a closer look at these results in Section 6.

AuYoung et al. [4] compare a batch scheduler with four priority queues based on the Maui scheduling algorithm [21] and a conservative backfilling algorithm with an implementation of a periodic combinatorial auction (CA) [22]. Maui is an open source job scheduler for clusters. In a combinatorial auction, users submit bids specifying resource combinations accompanied with an amount of money they are willing to pay for that combination. Periodically, the auction clears and determines a number of winning bids. The problem of clearing such a combinatorial auction in a value-maximizing manner is known to be NP-complete [23]. AuYoung et al. use a greedy approximation algorithm in order to clear the market in a timely fashion. Their simulation was based on a one week workload trace of the SDSC Blue Horizon cluster, a publicly available trace from the Parallel Workloads Archive [24]. They obtained up to 400% improvement with the CA compared to the batch scheduler. The differences between the batch scheduler and the combinatorial auction-based approach are however big: while the combinatorial auction uses advance reservations in order to optimize the schedule, the batch scheduler schedules jobs in an online manner. It is therefore difficult to indicate what underlying cause elicits this difference. Because the CA's implementation details and algorithm, the clearing period, the allocation window and the workload data are not specified in [4], we were unable to reproduce their results.

In [2], Lee et al. generate complex piecewise linear utility functions for jobs in workload traces. They compare a heuristic based on a genetic algorithm (GA) that maximizes the aggregate value generated in the system according to these utility functions with the performance of the *Priority-FIFO* algorithm (similar to the *PrioFIFO* algorithm in [3]), and an EASY and conservative backfilling algorithm. A genetic algorithm is a search heuristic that emulates the natural evolution process, and is used to generate solution to search and optimization problems. Backfilling allows short jobs to skip ahead in the queue provided they do not delay any other job in the queue (conservative) or they do not delay the job at the head of the queue (EASY). Using a workload containing 5000 jobs from the SDSC Blue Horizon cluster, Lee et al. show that the GA heuristic outperforms FIFO with backfilling and *Priority-FIFO* on aggregate utility by respectively 14% and 6%. Lee et al. focus on *load* and *decay type* of the user's utility function as the parameters that cause these differences and point out that both *Priority-FIFO* and the GA heuristic perform well in high load conditions. The higher the load, the greater the performance difference is between value-based and traditional schedulers.

Libra is an economy-based job scheduling system for clusters, based on a proportional share scheduling algorithm. In a proportional share algorithm every job has a weight, and jobs receive a share of the available resources proportional to the weight of each job. Sherwani et al. [25] present the details of this scheduler, as well as a detailed performance analysis. They show that Libra's proportional share algorithm performs better than a FIFO scheduler. No comparison is however made with a priority queue scheduling policy.

Many others [7,17–20] have compared fine-grained value-based and traditional scheduling methods. In these studies little or no attention was paid to the use of priority queues as an alternative for fine-grained value-based schedulers.

3. Value-based scheduling

We assume that all users have a certain understanding of the value they associate with the jobs they want to run on a cluster. The expression of value is done through a medium common to all users of the shared resource. A real or virtual *currency* is used to fulfill this role. The communication of these user valuations to the scheduling mechanism allows for the construction of an economically efficient schedule, in the sense that scarce resources are allocated to users who value them the most. In order to prevent users from consistently communicating the highest possible value to the scheduler, users are endowed with limited budgets that they use to pay for the execution of their jobs. Unless incentive compatible mechanisms are used, the value the user communicates to the scheduler does not necessarily correspond to the user's *private value* for the job's execution. In non-incentive compatible mechanisms, users have a potential gain by not revealing their private value truthfully. Techniques such as second pricing [26] or k-pricing [27] can be used to achieve incentive compatibility. In the context of this work however, we do not consider such strategic behavior and focus on the differences

between a coarse-grained and fine-grained expression of user valuations.

In a fine-grained value-based scheduling policy the expression of a user's valuation is typically done by means of a *bid*. A bid can be time-varying (as presented in [3,28,2]) or static (as in [29,8,30–32]). With time-varying bids, the user submits a bid as a monotonically decreasing function of the job's completion time. The scheduling policy can then take the evolution of the job's value with respect to the completion time into account to optimize the schedule. A static bid does not include this information.

In a coarse-grained value-based scheduling policy a user picks one of the discrete values defined by the policy to express his valuation. In the priority queue system that we consider as a model for the coarse-grained approach in this paper, this involves a choice for the job queue whose charging rate best fits the user's private valuation. Expressing a time-varying valuation in such a priority queuing system would be burdensome, as it would require resource requests to hop from one queue to another. Therefore, in order to obtain a fair comparison between both scheduling approaches, we use a model in which a user expresses his valuation as a static value, an amount of money he is willing to pay for a job he submits to the cluster.

The jobs submitted on a general purpose cluster are not always trivially parallel[1] and may therefore require co-allocation, i.e. the simultaneous allocation of multiple processors to one job. Adding co-allocation support in a scheduling algorithm gives rise to significantly higher delays and lower utilization due to schedule fragmentation, because jobs have to wait for sufficient processors to become available before being scheduled. We define the *parallelization degree* of a job to be the number of processors that need to be allocated to the job. The higher the average parallelization degree of a workload, the higher the chance utilization on a cluster will drop due to fragmentation and higher delays.

In order to minimize fragmentation and maximize the utilization without violating the value-based decisions made by the scheduler, both algorithms presented in this section implement EASY backfilling as described in [33]. EASY backfilling allows short jobs to skip ahead in the queue provided they do not delay the job at the head of the queue. If the first job in the queue cannot start, the backfilling algorithm attempts to identify a job that can backfill. Such a job must require no more than the currently available processors, and it must not delay the first job in the queue: either it terminates before the time the first job is expected to begin, or it only uses nodes that are left over after the first job has been allocated its processors. To implement this feature, we have to assume the knowledge of the exact runtime of each job. Determining estimates of task runtimes is a complex problem that has been extensively researched [34–39]. For workloads that are executed repeatedly (e.g. in silico analysis of protein folding processes for a pharmaceutical company) and for which there is a clear relationship between the application's parameters and its runtime (e.g. a parameter sweep application), these authors show that it should be possible to build fairly accurate models. Depending on the type of the application, we acknowledge that making exact predictions can be difficult and that the error on the predicted runtimes can be significant. Examining the influence of errors in the runtime of jobs on a backfilling algorithm falls beyond the scope of this paper. The resulting scheduling error made by the backfilling algorithm would however be the same for both our fine-grained and coarse-grained algorithms. Therefore, this issue has no influence on the results of the comparison made in this contribution.

[1] Applications are called trivially parallel when the jobs that form the application are fully decoupled and have no precedence constraints.

Because only few cluster systems currently allow jobs to be interrupted by the scheduler once they are started, our scheduling algorithms do not use preemption. Once the execution of a job has started, it cannot be interrupted to give place to a job with a higher value.

3.1. Priority queues

We have chosen to adopt a priority queuing scheme as an implementation of a coarse-grained value-based mechanism as it provides a good fit for this scheduling model and it is used in practice. In this priority queue implementation, all jobs from a higher priority queue are executed before jobs in lower priority queues, while requests in each queue are handled in a FIFO manner. Each queue has a predefined and fixed charging rate associated with it, so that low valued jobs map in a low priority queue and high valued jobs map into a high priority queue. This algorithm is also used in [2,3].

When building such a queue-based system, two parameters are to be determined. Both parameters have a significant impact on the total value generated by the queues.

First, the *number of queues n* determines the granularity of our system. There are extremes for this parameter where $n = 1$, an equivalent to a FIFO queue where no valuation information is used to schedule the jobs, and $n = J$ with J the total number of jobs, thus creating an equivalent for a fine-grained system in which each separate valuation is taken into account.

The second parameter is the configuration of the *queue's charging rates*. When users associate a value v with the execution of a job, they submit their job to queue i with the highest charging rate P_i lower than v. Given a scheduling mechanism with n queues and lower bound a and upper bound b for the user valuation distribution, we define P_i as given in Eq. (1).

$$\forall i \in \{0 \ldots n-1\}: \quad P_i = a + i \cdot \frac{b-a}{n}. \tag{1}$$

Note that the above queue price setting mechanism is not necessarily optimal. An optimal, value-maximizing price setting mechanism is however highly dependent on the distribution of the user valuations and it is therefore hard to find in the general case.

3.2. Auction

As an implementation of a fine-grained value-based scheduling mechanism, we use an approach based on a first-price auction. In such an auction, the user communicates a bid to the auctioneer and pays for the execution of his job in accordance to his bid. From a value-maximizing point of view, a schedule is optimal when a job with value v can only be delayed by jobs with a value higher than v. Our fine-grained scheduling policy is therefore implemented as an auction in which all bids are sorted in a list. The scheduler considers each bid separately and greedily schedules the bid with the highest value, then the bid with the second highest value, and so on.

In this paper we want to compare the impact of the choice for a coarse- or a fine-grained value-based scheduling policy. Therefore, we try to minimize the differences between both approaches. We believe that this simple auction model without preemption provides a good basis for comparison with the aforementioned priority queue approach. Note that, due to the online and greedy nature of the policy and the absence of preemption, optimal efficiency in terms of generated user value is not guaranteed. Such a "spot market" model without preemption is however common in literature [40,29,6,17]. More complex models would include preemption [41] or allow for advance reservations [42,4].

4. Simulation

In this section we discuss the details of the simulated environment that we use to compare both scheduling policies. The use

of simulation enables us to efficiently study the consequences of varying parameters on the performance of both algorithms in a controlled manner.

4.1. Workload

We use real-world workload traces from the Parallel Workloads Archive [24] to model the workload in our simulation. This archive contains raw log information regarding the workloads on parallel machines. There are more than 20 traces available. In order to obtain results that are independent of specific workload characteristics, we have selected three traces with a significantly different workload. We discuss these traces in detail in Section 5.2.

In addition to general simulation information such as number of processors, number of users and total duration, we also extracted the User ID, the job's submission time, the number of processors requested and allocated and the runtime for each job in the trace.

4.2. Valuation distribution

While traces are very useful to model realistic workloads, this is not the case for modeling valuation distributions. The currently available workload traces include little or no information on the users' valuations for each job. The only information on user valuations available in some of the workload traces is the ID of a priority queue in the cluster's scheduling system. While converting from a continuously distributed valuation to a priority queue system is easy, the reverse operation is much more complicated. Lee et al. [28,2] attempt to use these user-assigned priority levels and wait times to generate complex utility functions for each job in a workload trace. We however believe that the choice for a certain priority level is user specific, and it does not necessarily give a precise picture of the user's real valuation for that job. Different users have different methods for mapping their tasks on a queue, and some users probably do not have any method at all. In our opinion, the user's choice for a certain priority level is in most cases not a purely value-based decision, but may instead be influenced by for example the number of queues, the available charging rates, the load at the moment of job submission, the implemented currency system and budget distribution or additional constraints such as a restriction on the maximum runtime or maximum parallelization degree of a job.

Because of lack of robust real-world statistical data on this matter, we have chosen to model our valuation distribution as a normal distribution with a mean μ and a varying density σ.

4.3. Grid Economics Simulator

We have evaluated the proposed scheduling algorithms in a simulated market environment delivered by the Grid Economics Simulator (GES) [43]. GES is a Java-based simulator that has been developed in order to support research into different market organizations for economic cluster and grid resource management. The simulator supports both non-economic and economic forms of resource management and allows for efficient comparative analysis of different resource management systems. In order to run experiments efficiently, the simulator provides a framework based on Sun's Jini [44] technology to distribute experiment workloads over clusters and desktop machines, which is used in this study. In each experiment we have conducted 5 iterations for every sample point of the independent variable to obtain statistically significant results. Further on, we present and discuss average values for each of these sample points. Because the relative standard deviations of all simulations were very low, with an upper bound of 1.25%, they are not mentioned in the discussion of the simulation results.

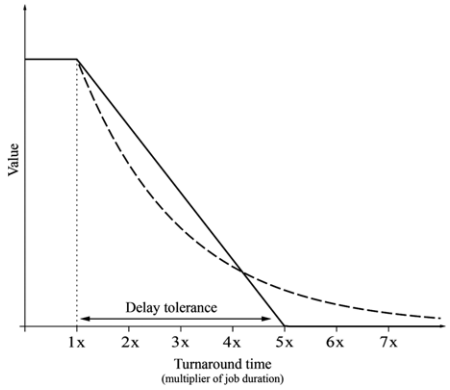


Fig. 1. Valuation function.

5. Evaluation

In this section we describe the metric we use to compare coarse- and fine-grained scheduling systems presented before. We also discuss the factors which potentially have an impact on the difference between both approaches, and describe our experimental setting.

5.1. User delay tolerance metric

A common metric in the evaluation of value-based scheduling mechanisms is the *aggregate value* that is generated by the execution of all the jobs in the schedule. In our simulation, all jobs in the workload will be fully executed. If we derive the aggregate value from the static values users communicate to the scheduling policy, the total generated value will be the same with all scheduling policies; only the sequence in which the jobs are processed will differ. We already discussed that a value-based schedule is optimal when a job with value v can only be delayed by jobs with a value higher than v. In order to evaluate scheduling policies in that respect, we need a metric that gives more weight to a faster turnaround time of a job by taking into account the *delay* a job suffers. Moreover, we want this delay to be evaluated in relation to the value the user associates with the job.

In this contribution, we use a decreasing valuation function to measure the delay in relation to the value the user associates with the job. The initial value is thereby assumed to be equal to the public value the user communicates to the cluster's scheduler, and remains constant for the duration of the job. When a user's submitted job is executed immediately, the generated value is equal to this public value. If, however, the job's execution cannot start immediately, the generated value decreases. We model the slope of this decay as a decreasing function of the user's patience, measured as multiples of the job length.

Following [3,45], we therefore introduce a linearly decreasing function modeling the *delay tolerance* of the user. According to a survey in [28], users sometimes have even more complex job valuation functions. Lee et al. observed that many of the user-provided functions show a very steep drop in value in the moments after job submission, with a leveling off later. To examine the influence of the shape of the decreasing delay tolerance function on our comparison, we also introduce a exponential decreasing function modeled after one of the proposed utility functions in [28]. Both curves are illustrated in Fig. 1.

In the linearly decreasing function, the delay tolerance equals the time until the delivered private value is 0. The exponential function initially decreases faster than the linear function, but never becomes zero. The Eqs. (2) and (3) show respectively the piecewise linear function f and the piecewise exponential function

Table 1
PWA workload traces.

Name	Duration (months)	# Jobs	# Users	# CPU's	ϕ (%)
SDSC Blue Horizon	32	243.314	468	1152	3.78
SDSC SP2	24	59.725	437	128	10.75
HPC2N	42	202.876	257	240	2.64

g, where v is the user's public value for the job and δ is the user's delay tolerance.

$$f_\delta(x) = \begin{cases} v: & 0 \le x \le 1 \\ -\dfrac{v}{\delta} \cdot (x-1) + v: & 1 \le x \le \delta + 1 \\ 0: & x \ge \delta \end{cases} \tag{2}$$

$$g_\delta(x) = \begin{cases} v: & 0 \le x \le 1 \\ v \cdot \mathrm{e}^{\frac{1-x}{\tau}}: & x \ge 1. \end{cases} \tag{3}$$

In order to obtain the value of τ in function of the delay tolerance δ in Eq. (3), we need to define the relationship between functions f and g. We state that the integral from 0 to infinity of f and g for the same delay tolerance δ should be equal, as stated in Eq. (4). Solving this equation as shown in Eqs. (5)–(8) results in a value of $\tau = \frac{\delta}{2}$.

$$\int_0^\infty f_\delta(x)\,\mathrm{d}x = \int_0^\infty g_\delta(x)\,\mathrm{d}x \tag{4}$$

$$\int_1^{\delta+1} -\frac{v}{\delta} \cdot (x-1) + v\,\mathrm{d}x = \int_1^\infty v \cdot \mathrm{e}^{\frac{1-x}{\tau}}\,\mathrm{d}x \tag{5}$$

$$\frac{v \cdot \delta}{2} = \lim_{x\to\infty} -v \cdot \tau \mathrm{e}^{\frac{-x}{\tau}} \tag{6}$$

$$\frac{v \cdot \delta}{2} = v \cdot \tau \tag{7}$$

$$\frac{\delta}{2} = \tau. \tag{8}$$

In the remainder of this paper, we use the aggregate value generated by the schedule as a metric to compare the performance of our scheduling policies. Following [2,3], this aggregate value is defined as given in Eq. (9), where $value_j(delay_j)$ denotes the value that the execution of job j with completion delay $delay_j$ generates for the user.

$$\text{Aggregate value} = \sum_{j \in jobs} \text{value}_j(\text{delay}_j). \tag{9}$$

The higher the value for this metric in a certain scheduling policy is, the smaller is the delay experienced by the higher valued jobs. It is important to understand that these decreasing valuation functions are only a measure for the delay suffered by a job in the scheduling algorithm. Therefore, these functions are private, they will never be communicated to the cluster's scheduling algorithm, and optimizations to increase the total generated private user value can only be made by correctly scheduling all jobs with a higher value before any job with a lower value.

5.2. Workload traces

The traces from the Parallel Workloads Archive used in this contribution are listed in Table 1. All traces are characterized by a long duration and relatively high load. The long duration is beneficial to the robustness of our experiments, while the relatively high load is necessary to bring out the qualitative differences in terms of value realization between different scheduling policies.

We pointed out earlier that the delay experienced by the jobs is an important metric in our comparison and that the average parallelization degree of a workload will probably be a determining factor in this regard. It is therefore important that the parallelization degree of a job is evaluated in relation to the total number of processors in the cluster. For a job that needs 100 processors for example, it is indeed clear that scheduling the job on a cluster with 128 processors will induce a much higher delay for all subsequent jobs than scheduling the same job on a cluster with 1152 processors. We therefore introduce the measure ϕ as the average parallelization degree of the workload, expressed as a percentage of the size of the cluster. If we take a closer look at the cluster traces available in the Parallel Workloads Archive, we observe that for all cluster traces ϕ has a 95% confidence interval of [2.84%, 5.94%].

The SDSC Blue Horizon log has been previously used by many other scheduling studies, including [4,2,46,47]. Its value for ϕ lies in the confidence interval, which makes it a suitable starting point for our comparison. We will also take a look at a trace from the SDSC SP2 cluster, which is smaller than the Blue Horizon cluster. The average parallelization degree for the SP2 trace is much higher ($\phi = 10.75\%$) than in most other traces. Next to these traces from the San Diego Super Computer Center, we also included a trace containing jobs from the High-Performance Computing Center North (HPC2N) in Sweden. The HPC2N trace is more recent than the other traces, and it has an average parallelization degree below the confidence interval ($\phi = 2.64\%$). On this cluster, the average job size is thus relatively small.

5.3. Experiments

In experiment 1, we will evaluate the difference in aggregate value between our auction-based scheduler and a priority-based scheduler when considering a varying number of queues ranging from 1 to 5. We also investigate the impact of an increasing user delay tolerance. Prices for the queues are set as specified in Section 3 in an interval between 100 and 10,000. We assume the user's delay tolerance function to be linearly decreasing, and the user's valuations to be normally distributed with $\mu = 5000$, $\sigma = 2000$ and a lower bound of 100.

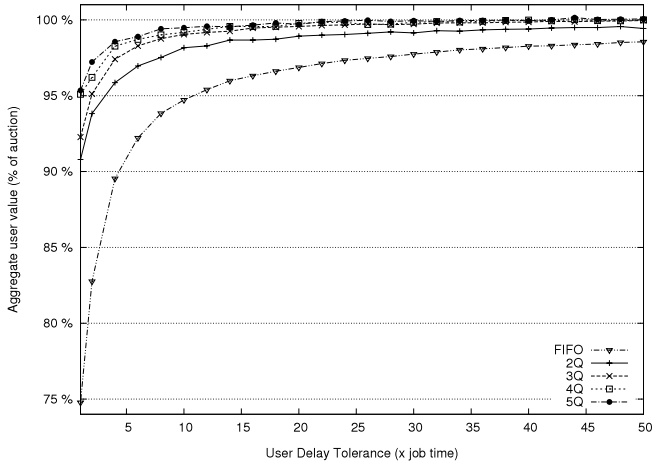
One of the primary goals of this work is to pinpoint which experimental parameters have a significant impact on the differences between coarse- and fine-grained scheduling approaches. Therefore we will take a closer look at a few of these parameters we expect to affect the differences between them. Lee et al. [2] already emphasized the influence of the load and the decay type of user's valuation function. We concentrate on:

- *Parallelization degree* — In experiment 2 we increase the parallelization degree of our workload trace with a factor ρ, while keeping all other parameters the same as in experiment 1.
- *Load* — We increase the load in experiment 3 by scaling the inter-arrival times of the jobs with a factor θ.
- *Type of user delay tolerance function* — In experiment 4 we change the slope of the user's delay tolerance function from linear to exponential, as shown in Fig. 1.
- *Valuation distribution* — In experiment 5, we vary the spread σ of the normal valuation distribution.

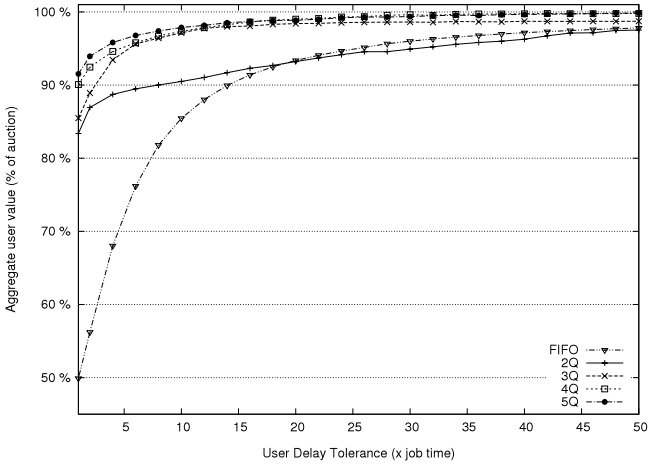
The simulation details of the experiments are summarized in Table 2.

6. Results

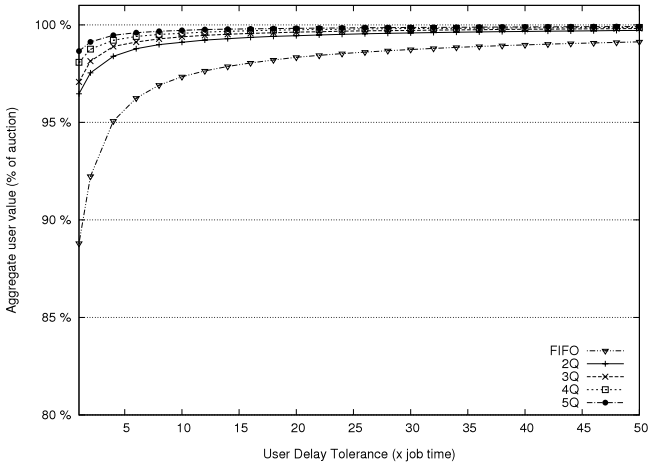
Before taking a look at the influence of the specific parameters presented in Section 5, we will discuss the results of the first



(a) SDSC Blue Horizon.



(b) SDSC SP2.



(c) HPC2N.

Fig. 2. Experiment 1.

Table 2
Experiment parameters.

Ex.	Val. func.	μ	σ	ρ	θ
1	Linear	5000	2000	1	1
2	Linear	5000	2000	[1, 4]	1
3	Linear	5000	2000	1	[0.7, 1]
4	Exponential	5000	2000	1	1
5	Linear	5000	[1000, 4000]	1	1

experiment shown in Fig. 2. The figures show the aggregate user value generated by the schedule, as a percentage of the value generated by the auction-based scheduling policy.

In Fig. 2(a) we present the results of the SDSC Blue Horizon trace. We observe that in case of a small delay tolerance, the value-based algorithms outperform the FIFO scheduler with a 15% margin. The number of queues in a priority queue system seems to have only a small impact on the generated value. When the number of queues increases, and thus the granularity of the coarse-grained system decreases, the generated user value grows. It thereby quickly approaches the value of the fine-grained auction.

Furthermore, when the user delay tolerance increases, the difference between the performance of the priority queue policy and the auction-based policy diminishes. The priority queue systems clearly perform better than the FIFO scheduler, and even approximate the performance of the fine-grained auction. The advantage of using a fine-grained scheduler in this case is thus limited, and a small number of queues are sufficient to attain a satisfactory level of prioritization in the schedule.

We also show the results for both the SP2 and the HPC2N trace in Fig. 2(b) and (c). We recall that the SP2 cluster trace had a high average parallelization degree ϕ, and the HPC2N trace had a lower value for ϕ. As expected from our assumption that the average parallelization degree is a determining workload factor for the performance of the scheduling systems, we observe that for SP2 the relative aggregate value of both the FIFO and the queue schedulers are much lower than with the SDSC Blue Horizon workload. The FIFO and 2Q algorithm perform badly, but policies with a higher granularity accomplish to limit the loss in aggregate value compared to the fine-grained policy to less than 15%. On the other hand, the HPC2N results are the opposite. The FIFO algorithm's aggregate user value is always within 12% of the fine-grained algorithm, and the queues achieve values less than 4% below the auction's result.

It is important to note that the general trend in function of user delay tolerance in all three workload traces is the same. In order to identify the workload characteristics that influence these trends, we now isolate each of these characteristics by altering the first SDSC Blue Horizon workload model in the next experiments.

6.1. Parallelization degree

In this section, we analyze the effect of an increased parallelization degree on the realized user value. In order to perform this

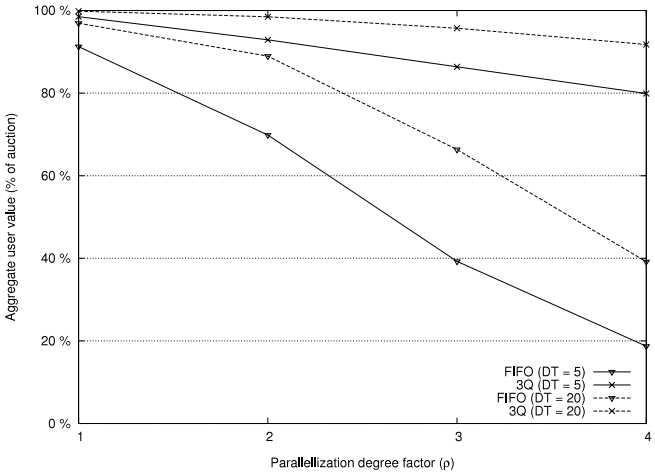


Fig. 3. SDSC Blue Horizon—experiment 2.

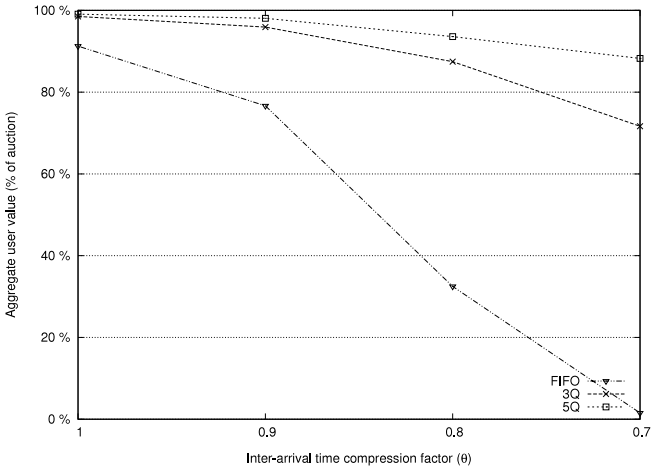


Fig. 4. SDSC Blue Horizon – experiment 3 – delay tolerance 5.

analysis based on the SDSC Blue Horizon trace, we increase the parallelization degree of the workload by multiplying the number of requested processors of each job with a factor ρ. We thereby make sure the number of requested processors remains smaller or equal to the total number of processors available in the cluster. In order to keep the total workload equal, we proportionally decrease the job's runtime with the same factor. The results of this experiment are presented in Fig. 3. The solid lines represent a delay tolerance of 5, dashed lines are used for a delay tolerance of 20.

We clearly observe that the parallelization degree of a workload has a negative influence on the performance for both the FIFO and 3Q algorithms compared to the auction. In Fig. 3, the value generated by the FIFO algorithm sinks away to 18.7% of the value generated in the auction setting with $\rho = 4$. However, the 3Q algorithm does relatively well with a value loss of only 20.1%. We observe similar but more moderate behavior with a delay tolerance of 20.

The study by Chun et al. [3] confirms our findings that the workload's parallelization degree is an important factor when it comes to the difference between coarse- and fine-grained systems. They show that, for parallel workloads, a fine-grained system delivers up to 2.5 times higher performance than a priority queue scheduler, which is much more than in our experiments. They however modeled the parallelization degree as a uniform distribution over the total number of available processors. For the cases in which the fine-grained value-based *FirstPrice* policy significantly outperformed the coarse-grained *PrioFIFO* policy in their study, ϕ ranged from 12.5% to 50%. In the Blue Horizon trace used in our experiment ϕ equals only 3.78%, and we found earlier that for all PWA traces ϕ has a 95% confidence interval of [2.84%, 5.94%]. The average parallelization degrees used in [3] are thus significantly higher than those of the workload traces available in the PWA, which explains the differences between their results and ours.

6.2. Load

It is important for a scheduler to perform well in heavy load conditions. The SDSC Blue Horizon workload already has a fickle load pattern, sometimes reaching quite heavy loads. We further increase the load in experiment 3, by manually scaling the inter-arrival times of the jobs with a factor $\theta = \{0.9, 0.8, 0.7\}$. That way, the simulation's duration is compressed to respectively 90%, 80% and 70% of the original duration, while the job runtimes and number of jobs remain constant. Consequently, the load is inversely proportional to the inter-arrival time compression factor θ, with the highest load occurring when θ is small.

Results are shown in Fig. 4. We again take a look at the situation in case of a user's delay tolerance of 5. The FIFO queue seems to be very sensitive to an increasing load. We hereby can confirm the findings from Lee et al. [2] that an auction-based or priority-queue-based algorithm performs well in high load conditions. The greater the load, the greater the difference between value-based and traditional schedulers such as FIFO.

The advantages of a fine-grained approach over a coarse-grained priority queue system are however much less significant. The 3Q algorithm does much better than the FIFO queue, with a decrease of 28.4% compared to the fine-grained scheduler. Using 5 queues, the coarse-grained method is able to limit this loss to 11.8%.

6.3. Type of user delay tolerance function

In Section 4 we presented two related delay tolerance functions: a piecewise linear and a piecewise exponential function. The exponential function decreases faster in the beginning, denoting an impatient user, but never really becomes zero. No matter how long the user has to wait, he still associates some small amount of value with the execution of his job. When we compare the simulation results of the exponential function with the linear function's results, we observe the aggregate value realized from experiment 4 to be a little bit lower than in experiment 1. The average relative difference between both results is 0.36%, and has an upper bound of 2.41%. It is worth mentioning that, when the load in the system increases as simulated in experiment 3, the relative differences between both user delay tolerance functions for coarse- and fine-grained approaches remain small. As a consequence of these very small differences, the choice for a faster decreasing user delay tolerance function seems to have only a limited influence on the difference between coarse- and fine-grained scheduling mechanisms.

Note that the absolute differences between both delay tolerance functions are more significant. For example, using an exponentially decreasing valuation function, the fine-grained approach generates up to 4.61% less value than using a linearly decreasing function. These absolute differences are however not extensively studied in this work, as they have no impact on our comparative study between fine- and coarse-grained value-based scheduling approaches.

6.4. Valuation distribution

A much cited advantage of value-based scheduling algorithms over traditional schedulers is the ability to cope with a large spread

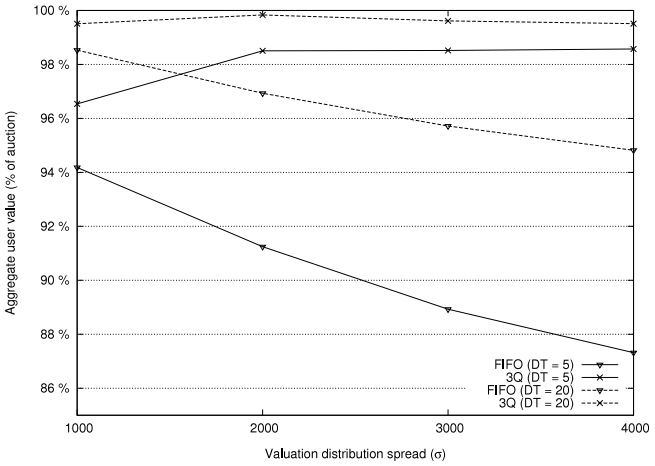


Fig. 5. SDSC Blue Horizon—experiment 5.

of user valuations. The higher the difference between a high value and a low value job is, the bigger become the consequences of an inefficient schedule. We evaluate this claim in experiment 5, by varying the spread σ of our normal valuation distribution while keeping all other experimental parameters equal. The results are presented in Fig. 5. The solid lines again represent a delay tolerance of 5, dashed lines are used for a delay tolerance of 20.

As the figure shows, an increasing spread in the valuation distribution has a significant impact on the generated value of the FIFO scheduler. Our coarse-grained priority queue system with 3 queues however, is able to cope with these larger differences in user valuations. Its relative performance compared to the fine-grained approach is almost constant.

7. Conclusion

Many authors have presented a comparison between a value-based scheduling implementation and traditional schedulers, such as FCFS or SJF, thereby generally showing large efficiency gains in favor of the value-based schedulers. However, only a few studies have compared fine-grained value-based scheduling to scheduling with a coarse-grained value representation through priority queues.

A fine-grained approach allows users to express their valuation as a value in a continuum, which allows the scheduler to maximize the overall generated value by the system. In a coarse-grained approach, on the other hand, users attach a priority to the execution of their job by picking one of a few discrete, predefined values. The latter is much easier, more stable and less cumbersome, while the first has the advantage of being self-sustaining in the sense that supply and demand can be automatically balanced, without the need for a system administrator to configure and maintain queue prices.

We modeled users to have a private valuation that is dependent on a level of *delay tolerance*, and used *aggregate utility* as a metric to compare both scheduling approaches. Using real-world workload traces from the Parallel Workloads Archive, we evaluated under which realistic conditions a fine-grained scheduler, implemented as an online auction-based mechanism with static bids and without preemption, can outperform a priority queue system. We thereby found that a priority queue approach with a relatively low number of queues can closely approximate the performance of the auction-based approach. The higher the number of queues and thus less coarse-grained the system is, the smaller the difference between coarse- and fine-grained scheduling becomes. We further found that, for our fine-grained approach to become profitable, the *parallelization degree* and *load* of the cluster's workload must increase beyond levels that are currently found in publicly available workload traces.

It is important to understand that our implementation of a fine-grained scheduling system is a somewhat simplified approach, developed to pinpoint the differences between fine- and coarse-grained strategies. More complicated fine-grained value-based schedulers will certainly be able to perform much better than our implementation, thereby significantly outperforming a priority queue system. According to our results, these efficiency gains will not be the effect of the use of fine-grained valuations, but they will rather be a consequence of the use of more complex implementation features such as advance reservations and clearing periods, time-varying or combinatorial bids and preemption.

References

[1] C.S. Yeo, R. Buyya, Pricing for utility-driven resource management and allocation in clusters, International Journal of High Performance Computing Applications 21 (2007) 405–418.
[2] C.B. Lee, A.E. Snavely, Precise and realistic utility functions for user-centric performance analysis of schedulers, in: Proceedings of the 16th International Symposium on High Performance Distributed Computing, ACM, New York, NY, USA, 2007, pp. 107–116.
[3] B.N. Chun, D.E. Culler, User-centric performance analysis of market-based cluster batch schedulers, in: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID'02, IEEE Computer Society, pp. 22–30.
[4] A. AuYoung, B.N. Chun, C. Ng, D. Parkes, A. Vahdat, A.C. Snoeren, Practical market-based resource allocation, Technical Report CS2007-0901, University Of California, San Diego, 2007.
[5] K. Lai, B.A. Huberman, L. Fine, Tycoon: a distributed, market-based resource allocation system, Technical Report cs.DC/0412038, HP Labs, 2004.
[6] G. Stuer, K. Vanmechelen, J. Broeckhove, A commodity market algorithm for pricing substitutable grid resources, Future Generation Computer Systems 23 (2007) 688–701.
[7] K. Vanmechelen, W. Depoorter, J. Broeckhove, Economic grid resource management for CPU bound applications with hard deadlines, in: Proceedings of CCGrid 2008, IEEE Computer Society, 2008, pp. 258–266.
[8] K. Vanmechelen, J. Broeckhove, A comparative analysis of single-unit Vickrey auctions and commodity markets for realizing grid economies with dynamic pricing, in: J. Altmann, D. Veit (Eds.), Proceedings of the 4th International Workshop on Grid Economics and Business Models, GECON 2007, in: Lecture Notes in Computer Science, vol. 4685, Springer-Verlag, Heidelberg, 2007, pp. 98–111.
[9] C.-M. Wang, H.-M. Chen, C.-C. Hsu, J. Lee, Dynamic resource selection heuristics for a non-reserved bidding-based grid environment, Future Generation Computer Systems 26 (2010) 183–197.
[10] C.S. Yeo, R. Buyya, A taxonomy of market-based resource management systems for utility-driven cluster computing, Software Practice and Experience (ISSN: 0038-0644) 36 (2006) 1381–1419.
[11] N. Dubé, Supercomputing futures: the next sharing paradigm for HPC resources, Ph.D. Thesis, Universit'e Laval, 2008.
[12] K. Vanmechelen, Economic grid resource management using spot and futures markets, Ph.D. Thesis, University of Antwerp, 2009.
[13] D.F. Ferguson, The application of microeconomics to the design of resource allocation and control algorithms, Ph.D. Thesis, Columbia University, 1989.
[14] R. Buyya, Economic-based distributed resource management and scheduling for grid computing, Ph.D. Thesis, Monash University, Australia, 2002.
[15] H. Izakian, A. Abraham, B.T. Ladani, An auction method for resource allocation in computational grids, Future Generation Computer Systems 26 (2010) 228–235.
[16] L. Hurwicz, On informationally decentralized systems, in: C.B. McGuire, R. Radner (Eds.), Decision and Organization: A Volume in Honor of Jacob Marshak, in: Studies in Mathematical and Managerial Economics, vol. 12, North-Holland, 1972, pp. 297–336.
[17] J. Gomoluch, M. Schroeder, Market-based resource allocation for grid computing: a model and simulation, in: M. Endler, D. Schmidt (Eds.), Int. Middleware Conference, Workshop Proceedings, PUC-Rio, Rio De Janeiro, RJ, 2003, pp. 211–218.
[18] F.I. Popovici, J. Wilkes, Profitable services in an uncertain world, in: SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, IEEE Computer Society, Washington, DC, USA, 2005, p. 36.
[19] K.L. Mills, C. Dabrowski, Can economics-based resource allocation prove effective in a computation marketplace?, Journal of Grid Computing (2008).
[20] I. Stoica, A. Pothen, A robust and flexible microeconomic scheduler for parallel computers, 1996.
[21] D.B. Jackson, Q. Snell, M.J. Clement, Core algorithms of the Maui scheduler, in: JSSPP'01: Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag, London, UK, 2001, pp. 87–102.

[22] P. Cramton, Y. Shoham, R. Steinberg (Eds.), Combinatorial Auctions, MIT Press, Cambridge, MA, 2006.
[23] D. Lehmann, R. Maller, T. Sandholm, The winner determination problem, in: P. Cramton, Y. Shoham, R. Steinberg (Eds.), Combinatorial Auctions, MIT Press, Cambridge, MA, 2006, pp. 297–317.
[24] Parallel workloads archive, 2010. http://www.cs.huji.ac.il/labs/parallel/workload/.
[25] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, R. Buyya, Libra: a computational economy-based job scheduling system for clusters, Software Practice and Experience 34 (2004) 573–590.
[26] W. Vickrey, Counterspeculation, auctions, and competitive sealed tenders, Journal of Finance 16 (1961) 8–37.
[27] B. Schnizler, Resource allocation in the grid—a market engineering approach, Ph.D. Thesis, University of Karlsruhe, 2007.
[28] C.B. Lee, A. Snavely, On the user-scheduler dialogue: studies of user-provided runtime estimates and utility functions, International Journal of High Performance Computing Applications 20 (2006) 495–506.
[29] R. Wolski, J. Plank, J. Brevik, T. Bryan, Analyzing market-based resource allocation strategies for the computational grid, International Journal of High-Performance Computing Applications 15 (2001) 258–281.
[30] Z. Tan, J. Gurd, Market-based grid resource allocation using a stable continuous double auction, in: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, 2007, pp. 283–290.
[31] U. Kant, D. Grosu, Double auction protocols for resource allocation in grids, in: Proceedings of the IEEE International Conference on Information Technology: Coding and Computing, ITCC 2005, IEEE Computer Society, 2005, pp. 366–371.
[32] M. Wieczorek, S. Podlipnig, R. Prodan, T. Fahringer, Applying double auctions for scheduling of workflows on the grid, in: SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, IEEE Press, Piscataway, NJ, USA, 2008, pp. 1–11.
[33] D.G. Feitelson, A. Weil, Utilization and predictability in scheduling the ibm sp2 with backfilling, in: IPPS'98: Proceedings of the 12th International Parallel Processing Symposium on International Parallel Processing Symposium, IEEE Computer Society, Washington, DC, USA, 1998, p. 542.
[34] S. Verboven, P. Hellinckx, F. Arickx, J. Broeckhove, Runtime prediction based grid scheduling of parameter sweep jobs, Journal of Internet and Technology 11 (2010) 47–54.
[35] M.A. Iverson, F. Özgüner, G.J. Follen, Run-time statistical estimation of task execution times for heterogeneous distributed computing, in: Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society, 1996, pp. 263–270.
[36] M.A. Iverson, F. Özgüner, L.C. Potter, Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment, IEEE Transactions on Computers 48 (1999) 1374–1379.
[37] W. Smith, I.T. Foster, V.E. Taylor, Predicting application run times using historical information, in: IPPS/SPDP'98: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag, London, UK, 1998, pp. 122–142.
[38] F. Nadeem, M.M. Yousaf, R. Prodan, T. Fahringer, Soft benchmarks-based application performance prediction using a minimum training set, in: E-SCIENCE'06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, IEEE Computer Society, Washington, DC, USA, 2006, p. 71.
[39] Y. Zhang, W. Sun, Y. Inoguchi, Predict task running time in grid environments based on cpu load predictions, Future Generation Computer Systems 24 (2008) 489–497.
[40] R. Buyya, D. Abramson, J. Giddy, Economic models for resource management and scheduling in grid computing, Concurrency and Computation: Practice and Experience 14 (2002) 1507–1542.
[41] L. Amar, A. MuAlem, J. Stösser, The power of preemption in economic online markets, in: GECON '08: Proceedings of the 5th international workshop on Grid Economics and Business Models, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 41–57.
[42] K. Vanmechelen, Economic grid resource management using spot and futures markets, Ph.D. Thesis, University of Antwerp, 2009.
[43] K. Vanmechelen, W. Depoorter, J. Broeckhove, A simulation framework for studying economic resource management in grids, in: Proceedings of the International Conference on Computational Science, ICCS 2008. Vol. 5101, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 226–235.
[44] W.K. Edwards, Core Jini, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1999.
[45] D.E. Irwin, L.E. Grit, J.S. Chase, Balancing risk and reward in a market-based task service, in: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, HPDC'04, IEEE Computer Society, 2004.
[46] M.D. de Assuncao, A. di Costanzo, R. Buyya, Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters, in: HPDC'09: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, ACM, New York, NY, USA, 2009, pp. 141–150.
[47] R. Ranjan, A. Harwood, R. Buyya, A case for cooperative and incentive-based federation of distributed clusters, Future Generation Computer Systems 24 (2008) 280–295.

Ruben Van den Bossche is a Ph.D. student in the Department of Mathematics and Computer Science at the University of Antwerp (UA), Belgium. His research interests include scheduling mechanisms in grid and cloud computing.

Kurt Vanmechelen is a post-doctoral fellow in the Department of Mathematics and Computer Science at the University of Antwerp (UA), Belgium. His research interests include resource management in grid and cloud environments in general, and the adoption of market mechanisms in such systems in particular. In 2009 he received his Ph.D., in Computer Science, from the University of Antwerp (UA), Belgium.

Jan Broeckhove is a professor in the Department of Mathematics and Computer Science at the University of Antwerp (UA), Belgium. His research interests include computational science and distributed computing. He received his Ph.D., in Physics, in 1982 from the Free University of Brussels (VUB), Belgium.