NISTIR 7463

Investigating Resource Allocation in a Standards-Based Grid Compute Economy

Christopher Dabrowski

National Institute of Standards and Technology
U.S. Department of Commerce

NISTIR 7463

Investigating Resource Allocation in a Standards-Based Grid Compute Economy

Christopher Dabrowski
Software Diagnostics and Conformance Testing Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8530

October 2007

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology
James M. Turner, Acting Director

Investigating Resource Allocation in a Standards-based Grid Compute Economy

Christopher Dabrowski

Abstract

This paper investigates the use of emerging web service and grid computing standards to support resource allocation in a simulated grid compute economy. In this simulated economy, client consumer and service providers, acting independently, employ standards-based software components to enter into service-level agreements that assign grid resources to client tasks. To investigate whether standards-based components can be used to create viable resource allocations in a grid compute economy, a series of trials is conducted using a simulation model. In these trials, workload is varied over a simulated workday from moderate to overload levels. The results show that a standards-based grid compute economy can adapt to different workload levels to produce resource allocations that economically benefits clients, providers, and larger community that depends on the grid. Further, standards-based components can be used to allocate resources efficiently and consistently, without giving rise to anomalous influences that perturb market forces.

1. Introduction

Grid computing has been envisioned to offer high-performance computing as a commodity for those who require large-scale computing resources for tasks such as drug or engine component design, financial risk analyses, and many other commercial compute-intensive activities. To date, the benefits of grid computing have been demonstrated in preliminary commercial implementations. However, these systems have been largely limited to single enterprises or groups of collaborating enterprises, in which resource allocation is centrally controlled and allocation decisions are guided by non-market factors. In contrast, the longer-term commercial success of grid technology will be linked with the emergence of large-scale grid compute economies—distributed systems in which service providers and consumer clients interact through a de-centralized electronic marketplace to allocate grid resources. Here, providers and consumers, acting in their own self-interest, will enter into service-level agreements (SLAs) [1], [2] to purchase grid services for a price which fluctuates on the basis of supply and demand. While the realization of a grid compute economy is still years away, continued progress toward this goal will depend, in part, on use of standardized software components that enable interoperability in a large-scale, dynamic, unrestricted, marketplace.

In recent years, researchers from industry and academe have produced a set of standard specifications [1], [3], [4], [6]-[12] intended to create large-scale, interoperable grids. One important question is whether a computing grid can support a large-scale, electronic marketplace that is profitable to its participants and benefits the larger community using the grid? Can a grid compute economy react and adapt to changes in workload level and composition to maintain economic benefits to its participants? These questions are answered affirmatively in [13] and here. However, if emerging standards are to be a viable basis for a market-based grid compute economy, another important question is whether standards-based components can be used to produce coherent and efficient resource allocations in a grid compute economy. Are resource allocations consistently explainable in terms of adaptive response to market forces, or does operation of standards-based implementations impact resource allocation and hinder response to market forces? Worse, do anomalous behaviors arising from standards-based components impede resource allocation and influence marketplace decisions? The contribution of this study is a detailed investigation that focuses on the use of emerging specifications to support a grid compute economy.

The paper first presents a simulation model [14] in which standards-based components are used in a grid compute economy that exhibits economic behaviors motivated by [15]-[21]. In this grid economy, clients try to maximize profits by seeking providers that can execute their tasks for most profit, while providers allocate grid resources to tasks that are most profitable to them. The model contains multiple clients and providers that act independently in their own self-interest. To investigate the questions posed in this study, the operation of the grid economy is analyzed in detail and compared with the operation of the economy when providers allocate grid resources on the basis of traditional utilization thresholds. A series of trials is conducted in which provider resource allocation methods are varied as the grid compute economy operates over an 8-hour day under increasing workload. Results are evaluated using time-series analysis, well-known system efficiency measures, economic metrics, and a new metric for overall system welfare.

The results show that the grid compute economy based on standard specifications, in which

providers and consumers use profit as a basis for resource allocation decisions, benefits individual participants economically. In overload, profit-based resource allocation adapts to excess demand to provide more economic benefit than traditional utilization-based resource allocation, while yielding comparable efficiency. Time-series analysis shows that profit-based resource allocation forms coherent global schedules consistently in response to changes in marketplace conditions during a typical workday and that these schedules contrast in predictable and explainable ways with schedules formed through utilization-based allocation. The analysis finds no evidence of behaviors arising from standards-based components that interfere with operation of the market. The study demonstrates the feasibility of using standards-based components in a grid compute economy.

This paper is organized as follows. Section 2 overviews the basic grid compute model. Section 3 describes the grid compute economy based on this model while section 4 describes provider resource allocation behaviors. Section 5 defines the experiment design. Section 6 then discusses overall experiment results followed by detailed analysis of time series results in section 7. Section 8 concludes. Additional scenarios are reported in an appendix; related work is described in [13].

2. Basic Grid Model and Implemented Standard Specifications

The executable grid model, described more fully in [14], consists of a simulated internet site topology with realistic message delays. As in well-known grid models, e.g., [5], this model contains two types of sites: service providers and client consumers. Service providers manage a single cluster processor, capable of parallel execution. Clusters are of two sizes: 80% are small (500 processors) and 20% are large (5500 processors), all operating at the same speed. Each execution of the model simulates an 8-hour workday. Each client is initially given a set of tasks to complete during the workday; each task is assigned a start time when it enters the system, an execution duration, d_{task}, and target and maximum deadlines, $t_{taskTrg}$ and $t_{taskMax}$ respectively that are chosen from distributions defined in Table 1. A distribution of values for d_{task}, with a mean of 4320 s, is defined on the basis of workload studies [22], [23]. Each task has a cluster processor requirement, s_{task}, selected from a distribution in which 50% of tasks require 250 processors, 30% require 500, 10% require 750, and 10% require 1500, resulting in a mean of 500 for all tasks in the system. Knowing d_{task}, and s_{task} allows derivation of $cycles_{task}$, the number of CPU cycles needed for the task.

Providers and clients communicate using components that simulate standard web-service specifications for messaging [5], addressing [6], and stateful resources [7]. Each client discovers available grid cluster resources using a monitoring and discovery service that employs a two-level hierarchy of directories, linked together through simulated query aggregators modeled on the index service in Globus Toolkit 4. The directory hierarchy employs simulated information and index servers specified by WS service group [8]. To advertise availability of a cluster resource, each provider periodically registers a service description for its cluster processor with a local information server. Meanwhile, clients periodically execute a discovery cycle in which they query the directory hierarchy to retrieve relevant service descriptions for their assigned tasks. If the capabilities in the retrieved description fulfill task requirements, a client may offer to enter into an SLA with the provider for execution of the task. An SLA identifies relevant parties involved in the agreement and describes the terms that each party is expected to fulfill. The agreement also specifies monetary rewards each party is to receive upon fulfilling terms and

penalties incurred in the event of failure to do so. As described in the WS-Agreement specification [1], the client first contacts the provider and downloads an agreement template that provides an outline of an SLA. The template contains creation constraints which specify bounds that term values can take on. A client may then complete the template with specific term values and send the completed template to the provider as an offer. Following the protocol [1], a provider acknowledges receipt of the offer and then determines whether to accept or reject, using economic criteria described below. The provider sends the client an acceptance or rejection notice, which ends negotiation. Tasks accepted by a provider are placed on a pending queue and managed using the distributed resource management specification [10]. Queued tasks remain in a waiting state until executing task(s) finish. More than one task can run in parallel if their combined processor requirements do not exceed cluster capacity. The model allows no preemption, however providers backfill using a first-fit procedure. Upon completion, providers transmit execution results to clients. Each client periodically repeats discovery and negotiation until all assigned tasks are complete or the simulated workday ends.

Table 1. Key statistical distributions for experiment.

Symbol	Name	Distribution	Parameters and constraints
c_{cycle}	Cycle cost (¢)	normal	μ =0.001, ρ =0.001, $0.0009 \le c_{cycle} \le 0.0011$
m_{min}	Minimum profit margin	uniform	μ =0.175, ρ =0.0875, $0.15 \le m_{min} \le 0.2$
m_{max}	Maximum profit margin	uniform	$\mu = (0.3 - m_{min})/2$, $\rho = (0.3 - m_{min})/4$, $(0.3 - m_{min})/2 \le m_{trg} \le 0.3$
d_{task}	Task duration	normal	μ =4320 s, ρ = 1500 s
$t_{taskStart}$	Task start time	exponential	μ =3600 s, $t_{taskStart} < 6221$ s
$t_{taskTrg}$	Task target deadline	normal	$\mu = t_{globalTD}$ ρ = 3420 s, $(t_{sysEnd} - t_{globalTD})/2$, $t_{globalTD}$ =21960 s, t_{sysEnd} =28800 s
$t_{taskMax}$	Task maximum deadline	normal	$\mu = t_{sysEnd}$ ρ = 1710 s, $(t_{sysEnd} - t_{globalTD})/4$.

3. Grid Compute Economy

The executable grid model is extended to add simulated marketplace behavior, in which clients seek providers that can execute their tasks for most profits, while providers also accept offers that increase their profits. Each client and provider acts independently, without external direction. An accepted offer creates an SLA that establishes the price for completing a task. Over the simulated 8-hour day, d=28800 s, market price fluctuates in response to supply and demand, where supply depends on cluster processor availability and demand on the number and value of client tasks in the system.

At the start of a simulated day, each provider is assigned an operating cost for a single compute cycle c_{cycle} from a distribution in Table 1. Knowing the number of processors in the provider's cluster, $s_{cluster}$, the processor speed given as the global constant G_{speed}, and a duration d, the total provider operating cost over d can be calculated by

$$c_{oper} = c_{cycle} \bullet s_{cluster} \bullet G_{speed} \bullet d. \quad (1)$$

Each provider is also assigned a minimum profit margin, $m_{\min}$ from the distribution shown in

Table 1 and initially assumes an expected processor utilization, $u_{\text{expected}} = 0.75$. The provider then calculates a base price for a single compute cycle, b_{base}, as

$$b_{base} = c_{cycle} / u_{\text{expected}} \bullet (1 + m_{\min}), \quad (2)$$

which is the least amount the provider can charge to make a minimum profit during d, given u_{expected}.

Each provider maintains a *price schedule* divided into $i = 1...n$ time intervals, which is posted in the agreement template downloaded by clients. The schedule is periodically updated (every 120 s) as follows. For each interval i, each provider independently calculates the utilization u_i based on previously admitted tasks expected to run on its cluster during interval i, computes their average per-cycle price b^i_{avg} for i, and updates the posted asking price for i, b^i_{ask}, using the formula

$$b^i_{ask} = b^i_{avg} + (u_i/u_{expected} - 1)^{1/2}. \quad (3)$$

As the provider accepts tasks, b^i_{ask} rises for intervals that are heavily utilized. If demand wanes, b^i_{ask} correspondingly falls. Providers allow unlimited rise in b^i_{ask} but never allow it to fall below b_{base}. As the provider completes tasks, it accumulates revenue, r_{task} for each task and can calculate its profit by $\sum_W r_{task} - c_{oper}$ for the set of tasks completed W, given its operating cost c_{oper} computed by (1).

On the client side, in addition to its set of tasks, each client is initially assigned a minimum and maximum profit margin, m_{min} and m_{max}, from Table 1. To allow clients to initially estimate task cost, the model assumes clients have some foreknowledge of prevailing market prices, as might occur in an actual market. To simulate this, the average provider per cycle base price, $\bar{b}_{base}$, is calculated for all providers and initially made available to clients, which allows clients to compute the expected cost of each task, c_{expect}, as $\bar{b}_{base} \bullet cycles_{task}$. The maximum task revenue, r_{task}, is then calculated as

$$r_{task} = c_{expect} / (1 - m_{max}) \quad (4)$$

Note by (4), r_{task} depends on $cycles_{task}$, so more valuable tasks require more compute cycles, more resources, and run longer a factor in the analysis to come.

A key feature of this model is the concept of revenue decay, as used in [19]-[21]. A client can fully realize r_{task} by completing the task by its assigned target deadline, $t_{taskTrg}$. A task that completes after $t_{taskTrg}$ decays in value up until its maximum deadline, $t_{taskMax}$, after which it has no value. The *decay rate*, k_{task}, is $k_{task} = r_{task} / (t_{taskMax} - t_{taskTrg})$. The actual revenue r'_{task} earned for a task ending at $t_{taskEnd}$ is given by

$$r'_{task} = \begin{cases} r_{task} & \text{if } t_{taskEnd} \le t_{taskTrg} \\ r_{task} - k_{task} \bullet (t_{taskEnd} - t_{taskTrg}) & \text{otherwise} \end{cases} \quad (5)$$

The concept of revenue decay is shown in fig. 1.

Each task is given a budget based on minimum and maximum profit margins. The task maximum budget (i.e., the most a client is willing to pay) is set to $r_{task} - (r_{task} \bullet m_{min})$, while the minimum budget is set to $r_{task} - (r_{task} \bullet m_{max})$. In this way, revenue and budget values exceed the expected task cost by a reasonable amount, as typically occurs in an actual market. Following previous studies of grid compute economies, 20% of tasks are randomly designated to be *high revenue* tasks where their maximum revenues and budgets are increased by a factor of 5. The remaining 80% are considered *low-revenue* tasks. This allows clients to offer more money for high-revenue tasks in hopes of completing them sooner.

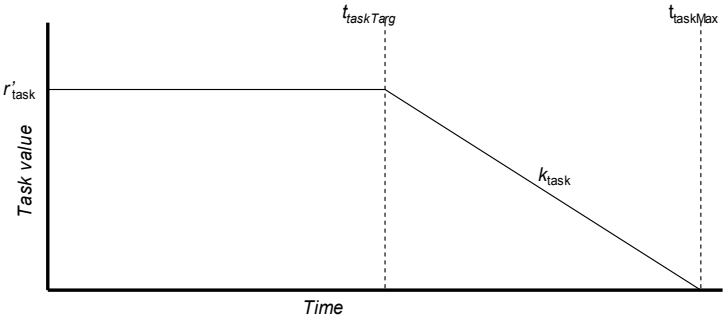

Fig. 1. Schematic of linear decay in task value over time

Once a client has found a set of providers that meets processor requirements for a task using the monitoring and discovery service, the client downloads each provider agreement template, in which the current provider utilization and price schedule information serve as agreement template creation constraints. For each provider template, the client iterates over its price schedule and, using utilization information to estimate task completion times, finds the interval i yielding the estimated highest profit, $r'_{task} - c_{task}$, where r'_{task} is computed by (5) and the client's cost, c_{task} by $b^i_{ask} \bullet cycles_{task}$. The client prepares an offer for each provider in which the offered fee is a minimum of b^i_{ask} and the task maximum budget, but not less than the task minimum budget. Each offer contains proposed agreement with terms for task fee, d_{task}, $cycles_{task}$, $t_{taskTrg}$, $t_{taskMax}$, k_{task}, and network addresses where input data can be retrieved and output can be sent. A client orders offers by decreasing profit and submits them to each provider. The client waits for a response before sending the next offer, as required by the [1]. Acceptance of an offer creates an SLA that precludes negotiation with lower-ordered providers. Once task execution completes, the actual fee is computed using (5) and sent by the client. A cancellation fee of 5% of the fee is assessed if either client or provider cancels the agreement.

4. Provider Strategies for Admission

Providers determine whether to accept or reject offers using an *admission strategy* and then respond to clients using the protocol in [1]. Admission strategies are the means by which providers allocate their resources. This study considered 3 classes of admission strategies, those that use: (a) profit maximization as the admission criterion; (b) utilization thresholds as the admission criteria; and (c) random or no control. Each class had 2 strategies. In the *profit-based strategies* class, a task, j_{new}, is admitted if its estimated profit exceeds the profit lost by delaying previously admitted tasks to complete j_{new}. Here, task profit is estimated as $r'_{task} - c_{prv}$, where r'_{task} is calculated by (5) and c_{prv}, which estimates the cost to the provider, by $cycles_{task} \bullet c_{cycle}$. In

one variant, *netProfit*, $\{j_1 \ldots j_n\}$ previously admitted tasks are ordered by estimated profit and j_{new} is then placed into a new proposed schedule $\{j_1 \ldots j_k, j_{new}, j_l, \ldots j_n\}$. If j_{new} is delayed by being placed behind $\{j_1 \ldots j_k\}$ other tasks, its profit may be reduced by the resulting decay as per (5). Similarly, if $\{j_l, \ldots j_n\}$ previously admitted tasks are further delayed by being ranked behind j_{new}, in the proposed schedule, their decay is computed and referred to as *deferred cost*. If the adjusted profit of j_{new} exceeds deferred cost, j_{new} is admitted; otherwise it is rejected. The second variant, *netProfitR*, uses the same procedure, but tasks are ordered by revenue, r'_{task} instead of profit. In this way, profit-based admission estimates the impact of a new task on the provider's profitability. Admitted tasks execute in the order determined by their respective ranking.

In contrast with profit-based strategies, *utilization-based strategies* take the more traditional systems management approach of attempting to maximize the number of tasks scheduled and completed and to ensure no cluster is over-utilized. In deciding whether to admit a task, utilization-based admission first identifies previously admitted tasks that are likely to execute during an interval I, bounded by the current time and the maximum end time of the incoming task, j_{new}. Utilization of a single task is measured by multiplying its estimated duration, d_{task}, by the number of cluster processors needed, s_{task}. If the combined utilization of j_{new} and the other tasks co-scheduled during I is less than 95% of total cluster capacity, the task is admitted. Otherwise, the task is rejected and must seek a less-utilized provider. The two utilization-based strategies are distinguished by the ordering criteria used to form execution schedules: *Shortest-job-first*, or *SJF*, and *First-Come-First-Serve*, or *FCFS*. In the third strategy class, the *random* strategy uses a coin flip to determine admission and to order tasks for execution, while *noControl* strategy admits all tasks and executes them in *FCFS* order.

5. Experiment and Metrics

The experiment compared the effect on global resource allocation of the six provider strategies as load was increased. The experimental topology consisted of 30 provider sites and 10 client sites, which operated over a simulated 8-hour workday. Load level was varied from 5% to 200% in 5% increments by generating varying numbers of tasks for each client (over 1100 for all clients at 200%). Variables for client tasks, and providers were assigned values from distributions described in Table 1. Task start times were distributed so that most tasks arrived in the early in the day. For each provider strategy, 100 repetitions were conducted at each load level.

To assess results, the analysis relied on well-known system metrics, including queue length, service time, and processor utilization. These metrics were supplemented by a *system welfare* metric, which combines measures of direct monetary benefit to all clients in the set of clients L and all providers in the set of providers P, together with the indirect benefit to a larger user community. Client benefit, B_{client}, is measured by summing proportion of maximum revenue realized (r'_{task}/r_{task}) and profit margin realized ($r'_{task} - c_{task})/r'_{task}$ over V, the set of all client tasks, and multiplying these terms in (6). Provider benefit, $B_{provider}$, is computed similarly over W, the set of tasks admitted by providers, in (7). However, to compute proportion of revenue realized, (7) sums the price stated in the SLA for the task over the maximum possible task price, where the latter is estimated using m_{min}, the minimum client profit margin for each task. To compute provider cost, (7) sums the operational cost, c_{oper}, for all providers in P. Note that (6) and (7) are left in un-simplified form for explanatory purposes.

$$B_{client} = \frac{\sum_{V} r'_{task}}{\sum_{V} r_{task}} \bullet \frac{\left(\sum_{V} r'_{task} - \sum_{V} c_{task} \right)}{\sum_{V} r'_{task}} \tag{6}$$

$$B_{provider} = \frac{\sum_{W} r'_{task}}{\left(\sum_{V} r_{task} - \left(\sum_{V} r_{task} \bullet \frac{\sum_{V} m_{\min}}{\|L\|} \right) \right)} \bullet \frac{\left(\sum_{W} r'_{task} - \sum_{P} c_{oper} \right)}{\sum_{W} r'_{task}} \tag{7}$$

Larger community benefit is measured by the ratio of V completed, $P_{complete}$. System welfare, S, is given by

$$S = (B_{client} + B_{provider}) \bullet P_{complete}. \tag{8}$$

The analysis also measured communication cost, C, which is defined as the sum of the averaged ratios of the total number of messages sent by clients and providers to discover resources and to negotiate an SLA for a task versus the minimum number of messages needed for these operations.. The formula for communication cost is provided in Appendix A.

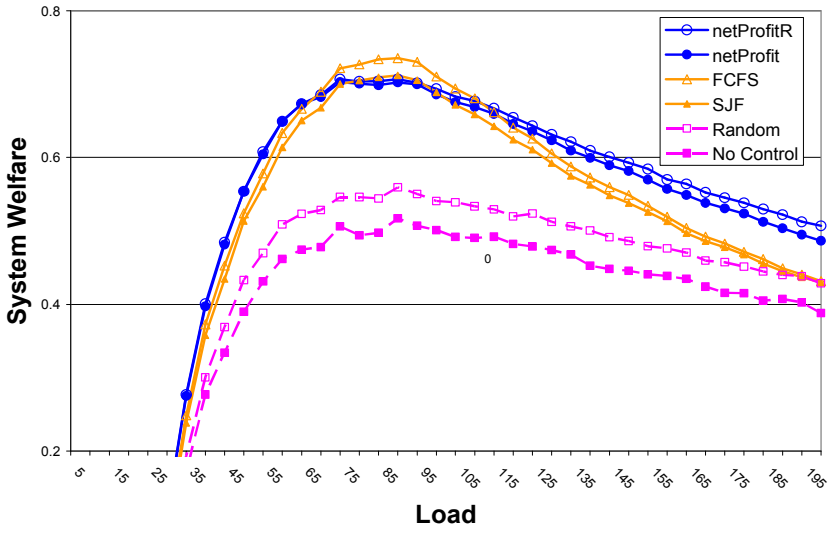

Figure 2. System welfare for 6 strategies over load ranging from 0-200% (100 repetitions)

6. General Observations

Fig. 2 and Table 2 show that in moderately-loaded range (50%-100% load), the simulated grid compute economy exhibited comparable system welfare for both profit-based and utilization-based strategies. Within this range, sufficient provider capacity exists to handle all client tasks; therefore, the ability to prioritize tasks by profit (or revenue) provides less advantage. Further,

utilization strategies incurred less communication cost, making them a potentially viable option when workload is at moderate levels. The random and no-control strategies performed more poorly as expected, because these strategies caused providers to become over utilized. As a result, tasks were delayed, leading to greater decay in task revenue and more cancellations.

In overload (≥ 105%), use of profit-based strategies resulted in higher system welfare. Decomposing the system welfare metric (8) in Table 2 shows that in overload, use of profit-based strategies resulted in higher proportion of potential revenue realized and higher profit margins for providers and clients. Table 2 shows that use of profit-based strategies resulted in completion of a higher proportion of high-revenue tasks than occurs with utilization-based strategies. On the surface, this is attributable to the operation of the profit-based admission strategy that prioritizes high-profit, or high-revenue, tasks for assignment to increasingly limited resources. By the same argument, the relative decline in system welfare for utilization-based strategies in overload is attributable to their inability to prioritize high-profit tasks and ensure that they run on scarce resources, which reduces revenue realized for both clients and providers. Near 200%, the decline in system welfare in the *FCFS* and *SJF* strategies in fig. 2 shows they are no more effective in completing high-revenue tasks than are random and no-control strategies. Table 2 also shows profit-based strategies exhibited task completion rates, P_{comp}, and communication cost that were more comparable to utilization-based strategies in overload.

In both moderately-loaded and overload conditions, the performance of profit-based strategies, in which providers and clients act independently to maximize their profits, evidences the feasibility of grid compute economies implemented using standard specifications. The results also show that use of profit-based strategies results in decidedly better performance in overload. However, these results do not fully answer questions about whether standards- based components can be used to support a coherent resource allocation process. Ideally, if the standards-based components are working properly, resource allocation decisions should be consistently explainable by the profit motive and by adaptive response to market conditions and workload composition. The resulting global schedules should be similarly explainable in terms of economic factors. There should be no influence on decision making arising from operation of standards-based components.

Table 2. Summary statistics for 4 provider strategies averaged over indicated load ranges.

Strategy	Range	*S*	Client		Provider		Proportion Complete			*C*
			Revenue Realized	Profit Margin	Revenue Realized	Profit Margin	Total	High Value	Low Value	
Net Profit	50-100	0.58	0.90	0.27	0.80	0.50	0.92	0.96	0.91	13.95
	105+	0.39	0.67	0.25	0.61	0.68	0.67	0.85	0.63	48.70
Net ProfitR	50-100	0.59	0.90	0.26	0.81	0.51	0.92	0.97	0.91	13.45
	105+	0.41	0.69	0.25	0.63	0.69	0.68	0.88	0.63	46.50
FCFS	50-100	0.59	0.88	0.26	0.79	0.50	0.96	0.95	0.96	7.17
	105+	0.34	0.59	0.25	0.53	0.63	0.68	0.68	0.69	45.56
SJF	50-100	0.56	0.87	0.26	0.78	0.49	0.94	0.94	0.94	7.58
	105+	0.33	0.58	0.26	0.52	0.62	0.68	0.68	0.68	45.08

7. Analysis of underlying dynamics

These questions motivated a more in-depth investigation into the resource allocation process underlying use of profit-based strategies. To gain insight into this process, it is useful to compare profit-based resource allocation with traditional allocation methods where utilization thresholds are used as decision criteria, since the latter do not involve profit or monetary factors. This comparison could be expected to more clearly reveal the impact of economic factors in the profit-based allocation process. By focusing on the overload range ($\geq$ 105%), it is possible to examine the system as it undergoes greatest stress and performance of strategies differentiates.

Selection of Time Series. Time-series analysis provides a good tool to compare characteristics of tasks that were accepted or rejected using profit- and utilization-based strategies, as well as to examine characteristics of resulting global schedules formed using the different strategies. To determine which task variables to generate time series for, recall that in profit-based admission, a new task is ordered in relation to previously admitted tasks on the basis of profit (*netProfit*) and revenue (*netProfitR*) to form a new proposed schedule. The new task is admitted if its profit exceeds the deferred cost of tasks delayed to complete the new task, where delay causes task value to decay according to (5). The extent that a lower-ranked task is delayed depends on the execution duration, or *task length*, and number of processors used, or *task width*, of task(s) ranked in front of it. Since task length and width is also used by utilization-based strategies to determine system utilization, these variables provide a good basis for comparison of the strategies. Time series data was therefore generated for task length and width over the 28,800 s experimental time period in 500 s intervals. To learn when tasks were admitted during the experimental period and how price varied over time, time series were generated for average frequency of admission and average price. The resulting analysis below is limited to *netProfit* and *FCFS*; *netProfitR* and *SJF* had very similar plots and are omitted.

Analysis of Resource Allocation. Fig. 3 shows the time series for tasks accepted by the *netProfit* and *FCFS* provider strategies in the overload range. Table 3 shows summary statistics for the variables plotted in fig. 3. Fig. 3a shows nearly all tasks submitted early in the day were admitted by 8500 s. However, *netProfit* providers accepted more high-revenue tasks than *FCFS* prior to 8500 s, while *FCFS* providers accepted a higher proportion of low-revenue tasks before 8500 s, consistent with the general observations and Table 2 statistics. In addition, fig. 3a shows *netProfit* providers admit some low-revenue tasks later, as evidenced by the slightly larger tail for *netProfit*. Table 3 confirms the general observation that *netProfit* and *netProfitR* providers accept tasks having higher average value, while in utilization-based strategies, values of accepted and never accepted tasks are similar.

In fig. 3b, price of high-revenue tasks rises for both strategies until 8500 s, reflecting strong demand early in the day. After 8500 s, price evens out as admission falls. After 5000 s, it also appears that *netProfit* accepts high-revenue tasks that are slightly higher in value. However, the price curve for low-revenue tasks in fig. 3b shows a different pattern. Here, *netProfit* providers accept low-revenue tasks of lesser value than *FCFS* providers before 8500 s; afterwards, *netProfit* providers accept low-revenue tasks of higher value. Why does this happen? This result at first seems contrary for a strategy that seeks to increase profit.

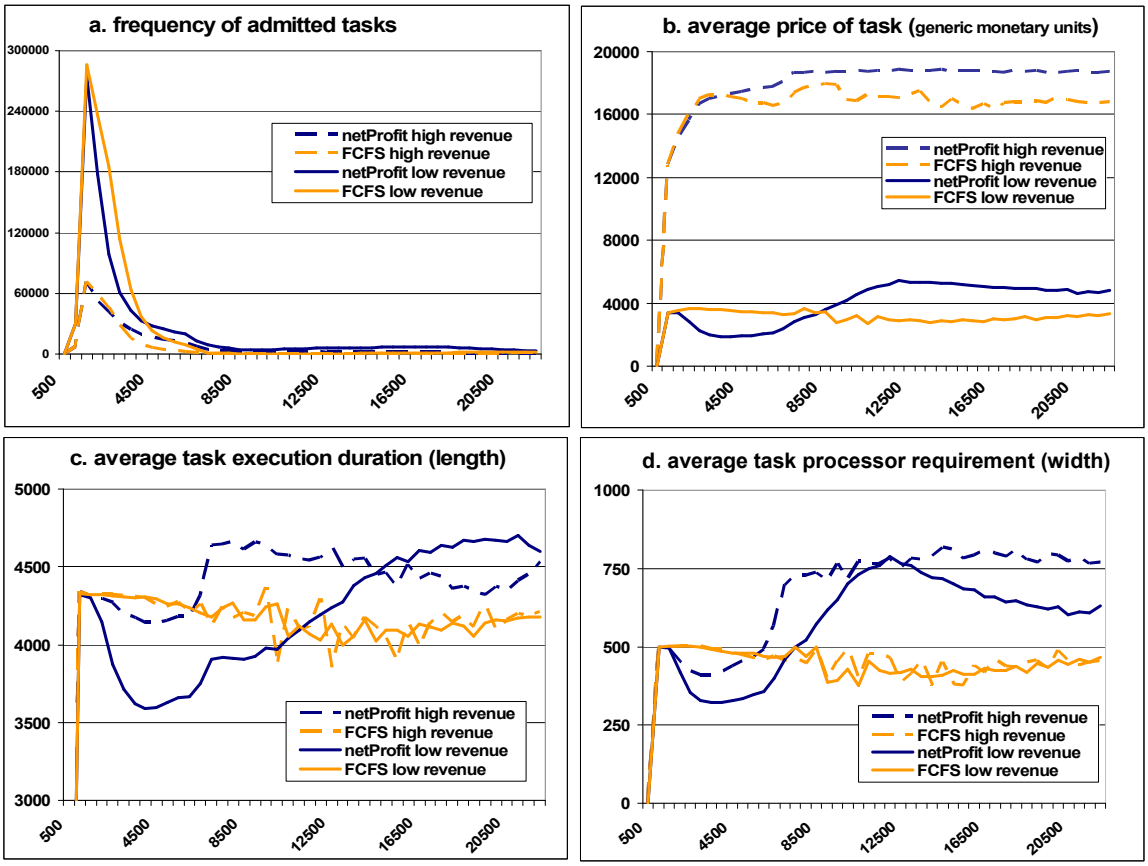

Figure 3. Time series for selected variables of tasks admitted by *netProfit* and *FCFS* utilization admission strategies over interval 0-20,800 s, combining results of all repetitions for loads ≥ 105%. In each diagram (a-d), horizontal axis represents time, while the vertical axis represents values of variables in diagram titles.

The answer to this question lies in the time series for task length and width. Fig. 3c and fig. 3d show that in the critical period prior to 8500 s, length and width of both high and low-revenue tasks accepted by *netProfit* providers is below average (recall average width is 500; average length is 4320 s). Above 8500 s, length and width of high and low-revenue tasks accepted by *netProfit* providers rises. In contrast length and width of tasks accepted by *FCFS* providers remains closer to the average during the entire 28800 s interval. (The 95% utilization limit caused *FCFS* providers to select tasks of lower length and width over time, but the effect is weaker.) Table 3 shows that, overall, average length and width of *all* tasks accepted by profit-based providers is less than that of tasks never accepted, while in utilization-based strategies, task length and width is comparable for accepted and never accepted tasks. The preference of *netProfit* providers for tasks of smaller length and width is attributable to the operation of the profit-based admission algorithm. Tasks with smaller length and width are more likely to be admitted because they are less likely to displace and delay previously scheduled tasks in a proposed schedule, thus reducing deferred cost. Similarly, previously admitted tasks with smaller lengths and widths are less likely to be displaced and delayed, also reducing deferred cost. (These same patterns are also observed for *netProfit* and *FCFS* tasks rejected in overload and for accepted and rejected tasks in the moderately-loaded range. See Appendix B.)

Table 3. Task length, width, and client revenue for admitted tasks versus tasks never admitted for four provider strategies over indicated load ranges.

Strategy	Range	Task Length		Task Width		Client Revenue	
		Admitted	Never Admitted	Admitted	Never Admitted	Admitted	Never Admitted
Net Profit	50-100	4281.9	5106.2	490.0	698.5	8035.6	7611.9
	105+	4128.1	4851.9	461.7	606.0	8581.6	6412.6
Net ProfitR	50-100	4283.0	5113.7	488.2	745.6	8026.2	7990.5
	105+	4132.9	4870.3	453.2	637.5	8567.3	6482.6
FCFS	50-100	4316.2	4367.1	499.4	467.7	8007.3	8305.0
	105+	4298.3	4368.1	489.9	511.2	7910.7	8152.7
SJF	50-100	4315.6	4322.9	499.2	490.3	8004.5	7759.5
	105+	4277.1	4380.8	488.7	512.2	7880.0	8169.7

The in-depth analysis confirms the general observations. However, the analysis also reveals that profit-based admission selects tasks that have smaller lengths and widths, e.g., consume fewer grid resources. This consistently results in formation of global schedules that yield more profit for the amount of computing resource used. The analysis also reveals profit-based resource allocation behavior coherently adapts to changes in workload level and task composition throughout the workday. As figs. 3c and 3d show, this behavior leads *netProfit* providers to admit more high-revenue tasks that use fewer resources than *FCFS* prior to 8500 s and to defer high-revenue tasks that consume more resources until later. The same behavior also leads *netProfit* providers to defer low-revenue tasks that consume more resources until after 8500 s, which explains the low-revenue *netProfit* price curve in fig. 3b. Recall in sec. 3, by (4) task value depends on the amount of resources, or c_{cycles}, used by the task. Low-revenue tasks with higher value are delayed, because the *netProfit* strategy is willing to sacrifice profit in all low-revenue tasks in order to allocate resources to more profitable high-revenue tasks early in the day. This causes the *netProfit* low-revenue price curve to be below the *FCFS* low-revenue curve before 8500 s in fig. 3b.

Globally, the operation of *netProfit* (and *netProfitR*) in overload incurs communication cost comparable to utilization-based admission (Table 2). Overall, as Table 4 shows, *netProfit* (and *netProfitR*) complete a higher proportion of tasks that do not decay in value (e.g., complete by their respective $t_{taskTrg}$). Because they admit tasks of smaller length and width, profit-based strategies also have smaller average processor queue lengths and lower processor utilization in comparison with *FCFS*, and also have a lower average task service time, expressed as a proportion of the 28,800 s experimental period needed to complete discovery, negotiation, and task execution. The *SJF* strategy achieves better efficiency by some measures because it prioritizes shorter tasks for execution, but still suffers reduced system welfare for reasons given above. Overall, profit-based resource allocation provides comparable, or slightly better, efficiency to utilization-based allocation in overload.

Table 4. Task processing statistics for four provider strategies over indicated load ranges.

Provider Strategy	Load Range	Proportion tasks completed without decay	Avg. Service Time	Avg. Queue Length	Avg. Processor Utilization	Avg. Latency for a single operation	
						Discovery	Negotiation
Net Profit	50-100	0.85	0.40	3.24	0.68	3.75	3.85
	105+	0.58	0.49	6.80	0.86	3.56	3.59
Net ProfitR	50-100	0.84	0.40	3.34	0.69	3.77	3.87
	105+	0.58	0.50	7.14	0.87	3.56	3.59
FCFS	50-100	0.82	0.42	4.00	0.72	3.85	4.36
	105+	0.54	0.52	7.82	0.89	3.57	3.61
SJF	50-100	0.83	0.38	3.52	0.70	3.83	4.35
	105+	0.55	0.47	6.79	0.87	3.57	3.61

Impact of Standards-Based Components. The preceding analysis showed that the behavior of the grid compute economy as a whole could be ascribed to causal factors arising from the economy itself, while differences in performance of systems using profit- and utilization-based provider strategies are also attributable to the strategies themselves. In no case did analysis of system or component behavior reveal adverse behavior traceable to specifications [1], [3], [4], [6]-[12]. In particular, there was no evidence that the service directory lookup and retrieval procedure based on [8], [9] or SLA negotiation protocol [1] inhibited operation of the grid compute economy or that it introduced aberrant behavior. This is supported by Table 4 which shows the average observed latency (or delay) to retrieve one provider service description from the two-tier directory structure is consistent with the expected delay that would be incurred by the number of messages needed for the operation (2 pairs of request-response messages, or 4 messages), given the Transmission Control Protocol (TCP) message sequence delays programmed into the model (an average 1.85 s delay for request followed by response). The same is true of average observed latencies incurred to negotiate a single agreement (one request-response to retrieve the agreement template followed by a client offer with 2 concurrent responses for 5 messages). The reduction of latency for all strategies in overload in Table 4 occurs because of increased sharing of established connections by multiple transmissions in overload, an efficiency feature of TCP simulated by the model [14]. The latency to retrieve a resource service description from the directory is similar for both profit-based and utilization-based strategies as is the average latency to negotiate an agreement. The increase in communication cost in Table 2 that accompanies increasing load is due to continued search by clients seeking contracts for the corresponding increasing number of uncompleted tasks that remain. In the case of the remote task management protocol [10], the command set proved adequate to support basic simulated task execution management and output data transfer functions, while [11], [12] provided sufficient basis for timely notification to clients of status of contracted tasks on remote clusters.

8. Summary and conclusions

The results show that independent use of profit-based resource allocation by many consumers and providers, based on standard specifications benefits both parties economically while also benefiting the larger grid community. Under overload situations, when the system was most

stressed, the global schedules formed through profit-based admission yielded decidedly better system welfare and better profits for clients and providers. The in-depth time-series analysis shows that, in overload, use of profit-based strategies by providers produced coherent resource allocations that were explainable in terms of adaptive response to market forces and workload characteristics, leading providers to consistently choose more profitable tasks that consumed fewer resources. Moreover, providers consistently allocated resources to tasks that were more profitable, required fewer processors, and had shorter execution times throughout the simulated day. In overload, use of profit-based strategies led to resource allocations that appeared to be slightly more efficient than those of traditional system utilization strategies. Finally, the analysis showed that there was no interference with the operation of the electronic marketplace from extraneous factors related to the specifications. In conclusion, computing grids that use standards-based components can support a large-scale, open-ended, electronic marketplace, in which resource allocation coherently responds and adapts to market forces and economic characteristics of the workload.

9. References

[1] Web Services Agreement Specification (WS-Agreement), GFD.107, Open Grid Forum, May 2007.

[2] K. Czajkowski, et al., SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, in Lecture Notes in Computer Science, (2002) **2537**, 153-183

[3] Open Grid Forum, http://www.ogf.org, (2006).

[1] [4] I. Foster, et al. (ed.), The Open Grid Services Architecture, V1.5, GFD.80, Open Grid Forum (July 2006).

[5] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, The Physiology of the Grid, An Open Grid Services Architecture for Distributed Systems Integration. Global Grid Forum (June 2002).

[6] SOAP V1.2 Part 1: Messaging Framework, W3C Recommendation (June 2003).

[7] WS Addressing, BEA Systems Inc., International Business Machines Corporation, and Microsoft Corporation, Inc. (March 2004).

[8] WS Resource Lifetime, V1.1, Computer Associates International, Inc., Fujitsu Limited, Hewlett-Packard Development Company, International Business Machines Corporation, and the University of Chicago (March 2004)

[9] WS Service Group, V1.0, Computer Associates International Inc., Fujitsu Limited, Hewlett-Packard Development Company, International Business Machines Corporation and The University of Chicago (March 2004)

[10] Distributed Resource Management Application API Specification 1.0, Global Grid Forum (June 2004)

[11] Publish-Subscribe Notification for Web Services, V1.0, Akamai Technologies, Computer Associates International, Inc., Fujitsu Limited, Hewlett-Packard Development Company, International Business Machines Corporation, SAP AG, Sonic Software Corporation, Tibco Software Inc. and the University of Chicago (March 2004)

[12] WS Services Topics, V1.0, Akamai Technologies, Computer Associates International, Inc., Fujitsu Limited, Hewlett-Packard Development Company, International Business Machines Corporation, SAP AG, Sonic Software Corporation, Tibco Software Inc. and the University of Chicago (March, 2004)

[13] K. Mills and C. Dabrowski, Can Economics-based Resource Allocation Prove Effective in a Computation Marketplace?, Journal of Grid Computing, *In press*, (2007).

[14] K. Mills and C. Dabrowski, Investigating Global Behavior in Computing Grids, in Lecture Notes in Computer Science, (2006), **4124**, 120-136.

[15] C. Ernemann, V. Hamscher, and R. Yahyapour, Benefits of Global Grid Computing for Job Scheduling, in Proceedings of the Fifth IEEE International Workshop on Grid Computing (GRID 2004), Pittsburgh (November 2004), 374-379.

[16] R. Wolski, J. Brevik, J. Plank, and T. Bryan, Grid Resource Allocation and Control Using Computational Economies, in F. Berman, G. Fox, and T. Hey, eds., Grid Computing: Making the Global Infrastructure a Reality Wiley and Sons, New York (2003), 747–772.

[17] J. Gomoluch, and M. Schroeder, Market-based Resource Allocation for Grid Computing: A Model and Simulation, in Proceedings of the First International Workshop on Middleware for Grid Computing, Rio de Janeiro (June 2003), 211-218.

[18] C. Yeo and R. Buyya, Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility, in Proceedings of the 7th IEEE International Conference on Cluster Computing, Boston (September 2005).

[19] D. Irwin, L. Grit, and J. Chase, Balancing risk and reward in a market-based task service, in Proceedings of 13th IEEE Symposium on High Performance Distributed Computing (HPDC), (June 2004), 160-169.

[20] F. Popovici and J. Wilkes, Profitable services in an uncertain world, in Proceedings of the ACM/IEEE Conference on Supercomputing, Pittsburgh (November 2005), 36.

[21] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes, Service contracts and aggregate utility functions, in Fifteenth IEEE International Symposium on High Performance Distributed Computing, Paris (June 2006), 119-131.

[22] Parallel Workloads Archive, The Hebrew University of Jerusalem, http://www.cs.huji.ac.il/labs/parallel/workload/.

[23] H. Shan and L. Oliker, Job Superscheduler Architecture and Performance in Computational Grid Environments, in Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, Phoenix (November 2003), 44.

Appendix A. Measurement of Communication Cost

This appendix describes measurement of communication cost, which is the sum of the communication cost of discovery and the communication cost of negotiation. Discovery communication cost is measured in terms of messages transmitted to complete discovery. There is an expected number, X_{dis} = 48, of query and response messages needed for a one task to retrieve service descriptions from the two-tier directory server for an average number of relevant providers. During one repetition of the simulation, the actual number of discovery messages (M_{dis}) for each task was measured. The ratio of M_{dis} over X_{dis} defines relative discovery communication cost (in units of X_{dis}) for one task. The average discovery overhead ($\overline{C}_{dis}$) across all tasks in the set of tasks V is calculated as

$$\overline{C}_{dis} = \frac{\sum_{V}(M_{dis} / X_{dis})}{|V|} \tag{A1}$$

The variable $\overline{\overline{C}}_{dis}$ denotes $\overline{C}_{dis}$ averaged over all experiment repetitions for a given load.

Similarly, negotiation communication cost is measured in terms of messages transmitted to complete negotiation. There is an expected number, X_{ngt} = 24, of offer, acknowledgment and response messages exchanged between a client and an average number of providers for one task to conclude a successful negotiation. During one repetition of the simulation, the actual number of negotiation messages (M_{ngt}) for each task in V was recorded. The ratio of M_{ngt} over X_{ngt} defines relative negotiation communication cost (in units of X_{ngt}) for a task. The average negotiation overhead ($\overline{C}_{ngt}$) across all tasks in V is calculated as

$$\overline{C}_{ngt} = \frac{\sum_{V}(M_{ngt} / X_{ngt})}{|V|} \tag{A2}$$

$\overline{\overline{C}}_{ngt}$ denotes $\overline{C}_{ngt}$ averaged over all experiment repetitions for a given load. Summing discovery and negotiation communication cost ($\overline{\overline{C}}_{dis} + \overline{\overline{C}}_{ngt}$) provides a measure of total communication cost incurred.

Appendix B

This appendix shows additional time series for tasks rejected for load values ranging from 105-200% (figure A.1), and for tasks accepted and rejected for loads ranging from 50-100% (figure A.2 and A.3).

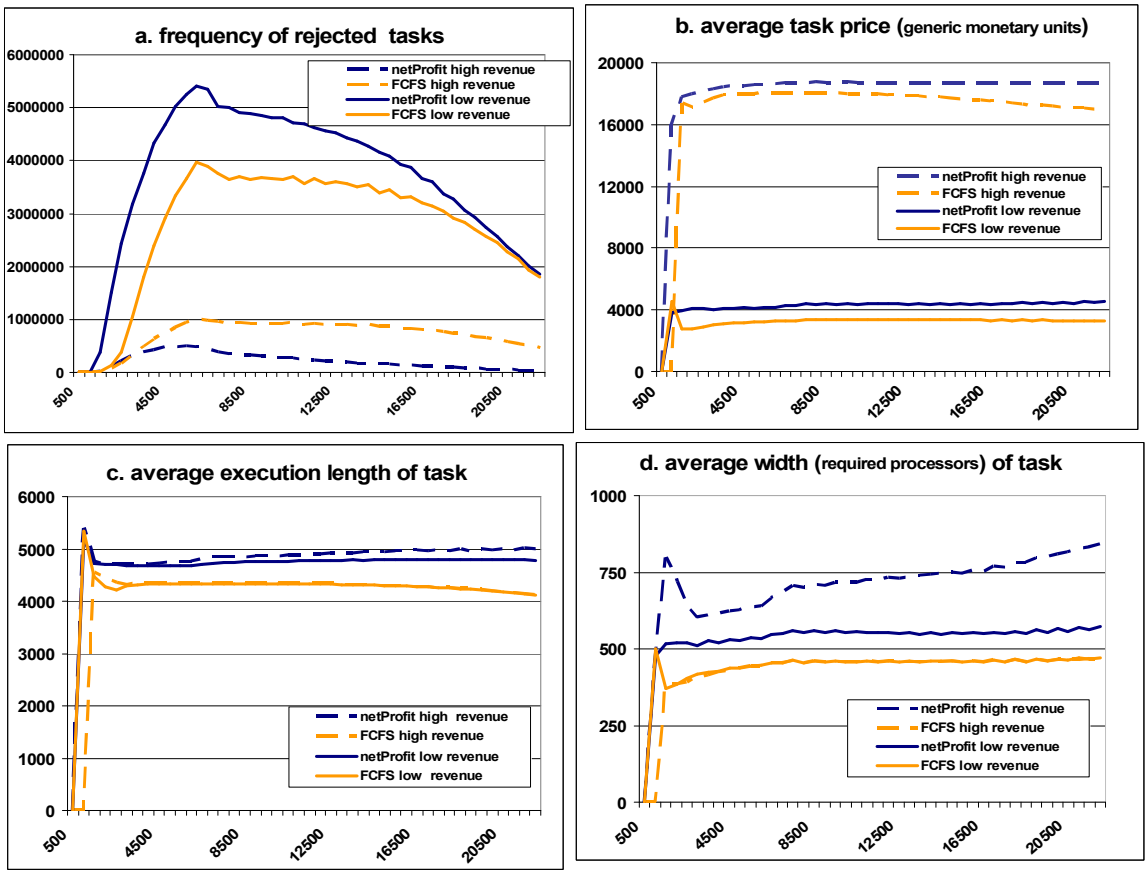

Figure B1. Time series over interval 0-20,500 s combining results of all repetitions at load 105%, for selected variables of tasks *rejected* by *netProfit* and *FCFS* utilization admission functions.

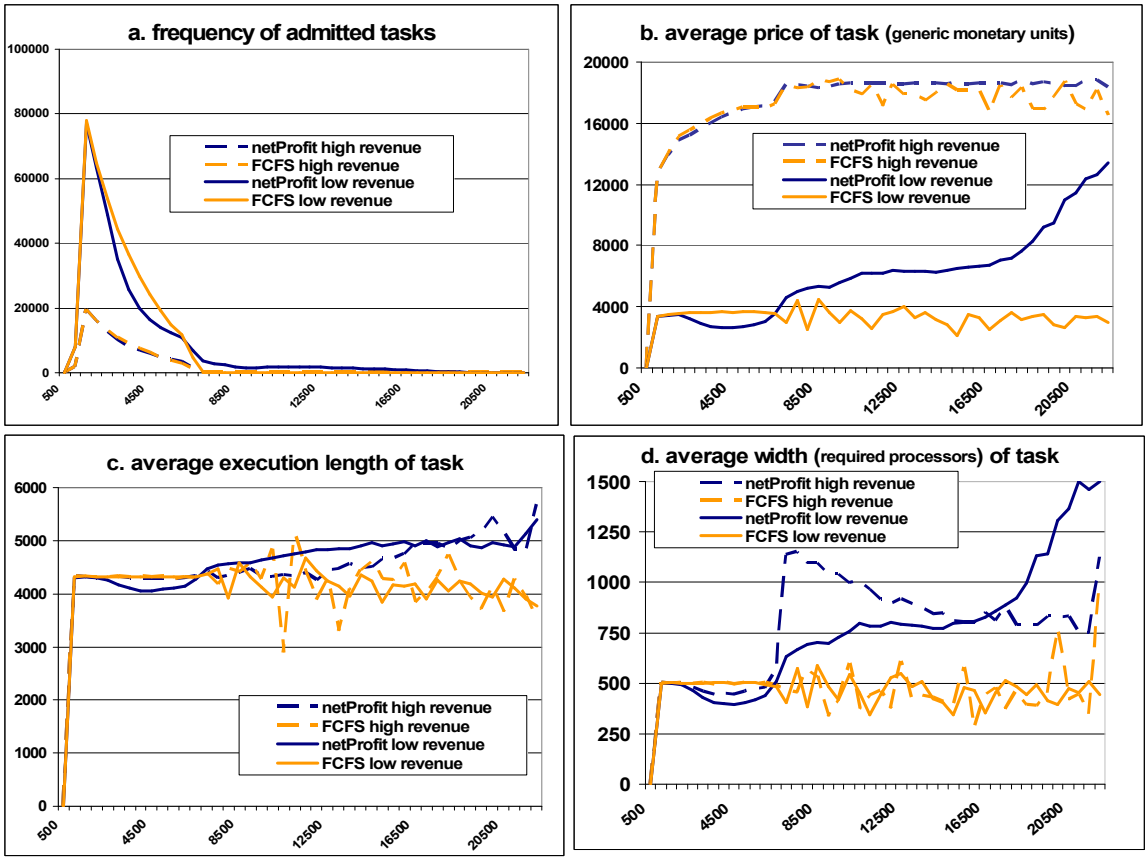

Figure B2. Time series over interval 0-20,800 s combining results of all repetitions at load 50 - 100%, for selected variables of tasks admitted by *netProfit* and *FCFS* utilization admission functions.

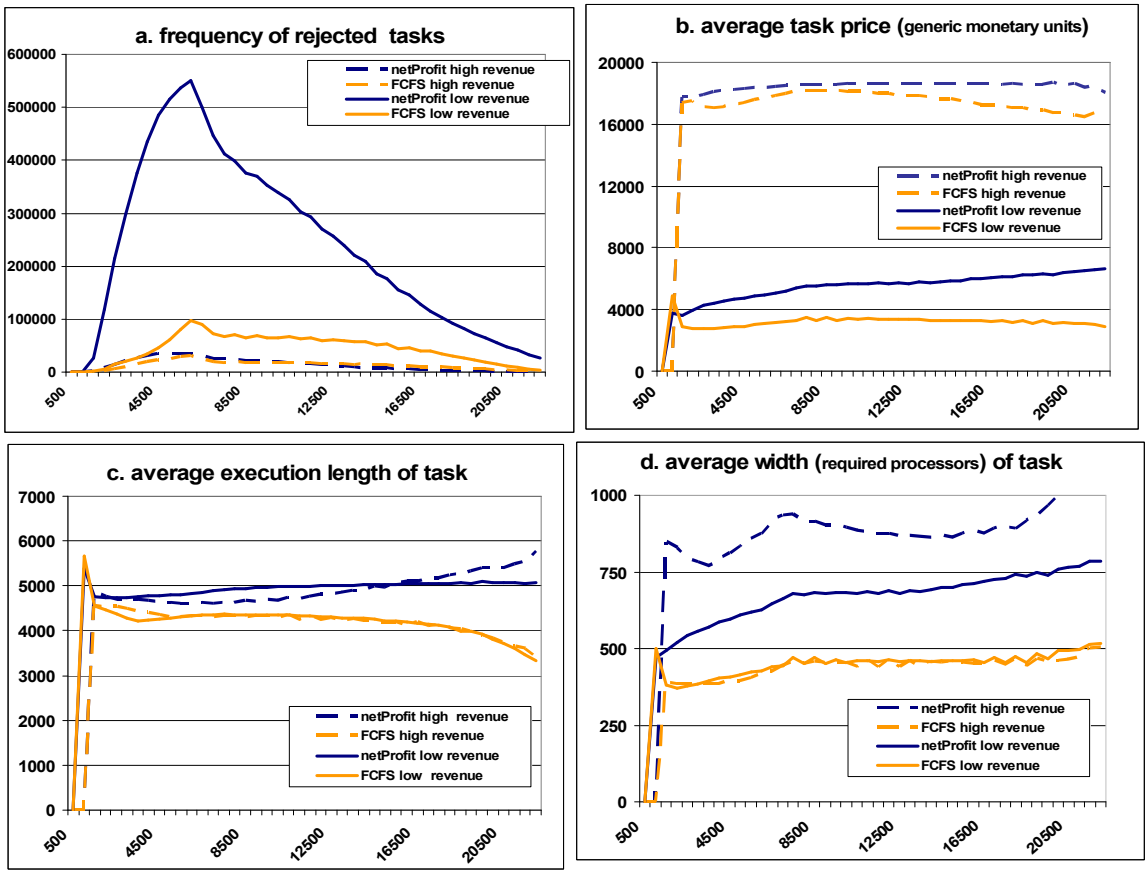

Figure B3. Time series over interval 0-20,800 s combining results of all repetitions at load 50 - 100%, for selected variables of tasks *rejected* by *netProfit* and *FCFS* utilization admission functions.