A Broker and Job Advertisement Based Grid Scheduling Framework

A.T. Thor, G.V. Záruba, D. Levine
The University of Texas at Arlington
Department of Computer Science and Engineering
{thor,zaruba,levine}@cse.uta.edu

Abstract - *A computational grid is a distributed collection of computer and storage resources created and maintained to serve the needs of virtual organizations where users seek to harness geographically distributed computational and storage resources for large scale intensive problem solving purposes. However, effective resource management and job scheduling poses a challenge when constraints such as resource utilization, response time, and global and local cluster policies need to be taken into account, while dealing with potentially independent sources of jobs, computational, and storage resources. It must be ensured that after scheduling decisions are made they are still valid by the time a job is executed, as well as that all the necessary resources are still available. Another critical problem is the encouragement of individual local domains to compete amongst themselves to become more efficient when accepting jobs. This paper discusses several common problems that have plagued resource management and job scheduling. We propose an extension to resource management and scheduling which adopts the use of brokers with job advertisement, resource matching, and job propagation within a hierarchical grid environment.*

1. Introduction

A computational cluster is a group of networked mostly homogeneous computers created by organizations for processing large data or computationally intensive jobs. Several such (remote) clusters may be combined together to form a computational grid. In a computational grid environment where data and computationally intensive jobs are to be processed, there exists a vast collection of resources ready to process jobs from (virtual) organizations. Resources within a computational grid are likely to be heterogeneous, i.e., individual clusters or computers may be of different architectures, capacities, running different operating systems and having diverse libraries. This often requires some form of resource-to-request matching [1],[2],[3] to effectively map jobs to resources. However, resource availability may rapidly change over time due to fluctuating supplies and demands.

In order to provide better scheduling and in an attempt to balance between global and local goals, some status information about the resources needs to be obtained. However, this brings up a contentious issue: at what resolution should monitoring be performed. Since jobs are being constantly submitted throughout the grid, while resources are used to process such jobs, monitoring information should be updated frequently. On the other hand, monitoring too frequently occupies valuable resources and bandwidth which could otherwise be used for job execution. Thus, there is a need to reach a balance between the risk of having outdated resource status information (which leads to incorrect scheduling decisions) and performing too much monitoring (wasting limited resources).

In a conventional grid environment, such as the ATLAS project [9] (a very large particle physics accelerator), system administrators are often required to monitor the activities of a selected group of sites and submit jobs to those sites if they are capable of processing such tasks. The sites chosen, however, may not always be the best sites for processing those jobs. This results in underutilized resources and stagnant efficiency of sites as there is neither incentive for improving efficiency nor remaining competitive. One of the main reasons for sub-optimal resource allocation is that when taking factors such as system resource utilization, response time, global and local policies into account while dealing with potentially independent sources of jobs, computational and storage resources, the job of managing resources and job scheduling becomes too tedious for a centralized entity to perform. In addition, a centralized scheduling entity may not have access or control over varying local policies; hence it could only perform scheduling based on a best effort basis. It is often up to the job-receiving host to perform the final leg of scheduling, based on its locally defined policies. As such, scheduling within a grid is often a multi-tiered process, as opposed to traditional scheduling techniques, with the job-submitting host scheduling its job to the best of its knowledge of the current state of the grid environment, while the receiving host handles the final phase of the scheduling process. Dividing the task of scheduling among several sites adds the advantage of easing the load and complexity of performing scheduling at a single location. However, in order to motivate individual local domains to compete in becoming more efficient while becoming more aggressive

in competing for accepting more jobs, an incentive mechanism should be applied. The main goal of this paper is to study the behavioral and performance aspects of two conventional grid scheduling schemes with our proposed scheme, with emphasis on profit maximization, while enforcing self motivated resource management technique with a combination of job optimization schemes (time and cost), along with support for re-negotiation if matching fails. We will also look into factors enticing participating entities to remain trading within a grid environment, and the effects of implementing a penalty scheme.

The rest of the paper is organized as follows: section 2 discusses related work. Section 3 provides an outline of how the proposed scheme works. Section 4 introduces our custom-made discrete event grid simulator, while section 5 shows simulation results and provides with a short analysis of the results. Finally, section 6 concludes the study as well as outlines our directions for future work.

2. Related Work

Buyya et al. [4],[8] investigate economy grids and requirements of economy based grid systems. The authors discuss ideas and challenges for implementing auction models for grid resource allocation, including the English, Dutch, and Double auctioning scheme. In the first two schemes (English and Dutch auction), potential resource buyers bid for the right to use resources. In the Double auctioning scheme, sellers set the selling price of their resources and the buyers set their respective budget for resource purchase. A “middleman” (GMA) acts as a broker and matches the two participants if the prices meet. In the first two schemes, however, buyers compete for the right to use resources by placing higher bids, which limits the motivational aspects for the sellers to improve themselves in terms of efficiency and resource management in order to stay competitive. The Double auctioning scheme attempts to split the competition between buyers and sellers; it is important to note that although this is a resource allocation scheme, the sellers are providing a service to those who have the purchasing power within the market. As a result, the pressure of staying competitive should fall on the sellers rather than those who are paying for a service. Moreover, creating competition amongst buyers would lead to deprivation of service to some potential buyers who cannot afford to pay high prices for a service, hence bringing down the number of potential trades between sellers and buyers. In addition, by shifting the competition to the sellers, there is an indirect advantage of motivating individual sellers to improve themselves so as to stay competitive within the market, aiding in enhancing the quality of resource management throughout the grid environment.

Yeo et al. [5] proposed an extension to the system centric cluster Resource Management Systems (RMS) to support utility driven resource allocation and management by introducing four mechanisms: (1) Pricing, (2) Economy-based Admission Control, (3) Economy-based Resource Allocation, and (4) Job Control. The Economy-based Admission Control unit primarily determines if a job is to be accepted, based on the job details and QoS requirements. The Economy-based Resource Allocation unit performs the task of resource and job matching, along with dispatching jobs to the matched nodes. The Job Control unit assumes the role of monitoring for jobs and resources. In this scheme, jobs are submitted to the cluster RMS using user-level job submission specification, where the admission control mechanism determines the feasibility of accepting a job. If a job is deemed acceptable, the resource allocation mechanism would perform the necessary matching of the job to an execution node. However, if the job is not viewed as acceptable by the admission control unit, the rejection decision would be fed back to the user. In other words, the responsibility of job submission lies upon the user to make smart decisions on where to submit the jobs to be processed, such that the probability of getting an acceptance is high (much like what is employed in numerous conventional job assignments in the grid today). In a conventional scheme, the user attempts to make a smart decision as to where to submit the jobs, and performs the job submission process. Although the targeted site may not be the best suited site for the job, the job is usually accepted by the receiving host unless its gateway is overwhelmed and starts dropping jobs. In which case, the user would have to either resubmit the job at a later time, or determine another potential execution site and attempt to send it there for execution. In this scheme, however, due to the potential rejection of jobs by the admission control mechanism, it would be necessary for the user to constantly monitor the status of each submitted job for rejection, as well as monitoring all potential clusters in order to increase his chances of getting his jobs accepted.

Xiao et al. [6] present GridIS targeting incentive based grid scheduling focusing mainly on the aspects of aggressiveness in resource reservations. According to Xiao the grid is essentially divided into resource consumers and providers, which may not always be the case in real systems as the resource consumer may also be a provider of other resources. The authors make two assumptions when designing the scheduling mechanism: i) execution time of all jobs is sufficiently long to make the overhead of remote job execution relatively negligible; ii) every provider is able to receive all job announcements. This works well in an environment with a small number of time consuming jobs that need to be executed in a fully connected network. However, depending on numerous factors such as availability of resources, site status of a site, network topology, and cost of job execution at remote sites; it may sometimes be more efficient and economical to execute those jobs locally. In addition, although fairness in job distribution is

an attractive goal in grid computing, it often comes with the price of excessive network flooding and added delay from propagating job announcements throughout the network. As a result, it is beneficial to have some mechanism for assessing overhead costs and evaluating propagation of job announcements, to aid in determining the worthiness of sending a job for remote execution. In addition [6] calculates resources allocated to each agent based on price ratio of agents paid to the computational resource price of that resource. In reality, it may be more desirable to have monitoring agents located at each site to handle the monitoring of resource usage.

In our framework, we seek to decentralize and shift the task of resource management to the individual resource providers, while providing incentives for them to evaluate their resource usage and efficiency so as to remain competitive within the grid environment. In order to automate the task of site monitoring and job submissions from users, broker agents will be summoned to take over such responsibilities.

3. Broker and Bid Based Scheduling

In our grid model, each cluster is associated with a head-node (gateway) running a software broker and other management software. An organization may have several head-nodes arranged into a meta-cluster, where the broker from one of the head-nodes is the broker of the meta-cluster (the set of all meta-clusters forms a grid). In this grid environment there are resource consumers (entities that submit jobs) and resource providers (entities that service job requests). Brokers of resource consumers post jobs; brokers of resource providers bid on these jobs by posting their fees and estimated job completion time. The posting broker evaluates bids and selects among them. The use of brokers alleviates the mundane and tedious tasks of resource monitoring and decision making from any single centralized entity within the grid environment. The automated brokers are responsible for submitting job advertisements, initiating and maintaining the job bidding process, participating in job bids, and performing re-negotiations should the need arise. A broker may play the role of a consumer (request broker), a provider (tender broker), or both at any point in time.

If a site has jobs to execute, a job advertisement object consisting of job description, job type definition and constraints is created and passed to its gateway broker. A job placement may require one of two types of optimization, time or cost optimization. A time optimized job places emphasis on completing the job at the earliest possible time, while not exceeding its available budget set aside for the job (best effort time but limited cost). Whereas a cost optimized job emphasizes on finding a cheapest tender for processing the job, while still completing the job by its specified deadline (limited turnaround time but best effort cost).

After receiving the job advertisement, the gateway requesting broker places the job advertisement which propagates to all of its nearby neighbors, as well as cluster brokers within the same local site. A tender broker receiving advertisements will perform a query of its local cluster monitors to search for nodes meeting the job requirements and constraints. The tender broker will then perform an analysis step determining which node to engage in the bid submission (attempting to maximize profits while keeping resource usage minimal).

The price determination process is dependent on the length of the estimated job execution time, amount and type of resources available during that interval, time of day, and the resource pricing profile. It calculates the expected returns in the follow ways: For each bid, an estimated operating cost $\upsilon_{cluster}$ is calculated using the formulae presented in Equation 1. As each cluster is made up of different combinations of resources and hence incurs varying operation costs, $\upsilon_{cluster}$ which represents a unique operation cost for a particular job over specific execution duration. The choice of whether to proceed in an ongoing bidding process depends on the resulting expected revenue ($Revenue=price-\upsilon_{cluster}$). Upon completion of bid creation, the tender broker will place a bid with the requesting broker. This round of bidding process is a closed bid, whereby all tender brokers have no prior knowledge of the budget allocated for this job.

$$\psi = est_job_runtime * \left(\frac{\$CPU + \$MEM}{Unit_time} \right)$$

$$\zeta = total_est_stagein \,\&\, stageout_files_size * \left(\frac{\$HDD}{GB} \right)$$

$$\rho_{system_est} = est_system_load = \int_{est_job_start_time}^{est_job_end_time} res_resource_type$$

$$\rho_{cluster_est} = est_cluster_load = \int_{est_job_start_time}^{est_job_end_time} res_resource_type$$

$$\tau_{system} = System_scaling_factor = \left(\frac{\rho_{system_est} - \rho_{system_at_50\%}}{\rho_{system_at_100\%}} \right)$$

$$\tau_{cluster} = Cluster_scaling_factor = \left(\frac{\rho_{cluster_est} - \rho_{cluster_at_50\%}}{\rho_{cluster_at_100\%}} \right)$$

$$\upsilon_{system} = est_cost_after_system_scaling = [\psi + \zeta] * (\tau_{system} + 1)$$

$$\upsilon_{cluster} = est_cost_after_cluster_scaling = [\psi + \zeta] * (\tau_{cluster} + 1) * \upsilon_{system}$$

Equation 1. Cost computation for individual site

At the requesting broker, only bids received prior to bid closing time are considered during the bid evaluation phase. Once the bid closes, all submitted bids are analyzed based on the optimization settings of that particular job. In cases where there are submitted bids meeting all the job requirement constraints, a winning bidder may be determined based on how closely it matches the job optimization constraints. However, in cases where none of the bids matches the constraints, a re-negotiation phase may be initiated. During the renegotiation phase, only those tender brokers who have submitted bids previously (signifying that they have the

available resources for the particular job) can participate. This second round of bidding is of an open bid nature where the budget allocated for that particular job will be announced. In addition, the requesting broker will be willing to negotiate as long as the new price charged by the tender broker falls within 110% (or other user specified values) of the budget originally set aside for that particular job, while the promised job completion time falls within the sum of the original deadline and a predetermined-time grace period. The requesting broker re-evaluates the new bids and selects the best match among those that meet the constraints. If no such bid is found then the job fails.

Figure 1 shows the proposed bidding process (the re-negotiation process is not shown as it is similar to the bidding process). After determining the winning bidder, a contract will be sent to the winning bidder as an agreement between the brokers. Only upon signing the contract can the tender broker begin job execution. If, for any reason the tender broker rejects the contract, the bid will be voided. A penalty price will also be included in the contract should the winning broker fail to complete the job execution by the agreed upon time (tentatively set to 30% of the bid price of jobs in our simulations).

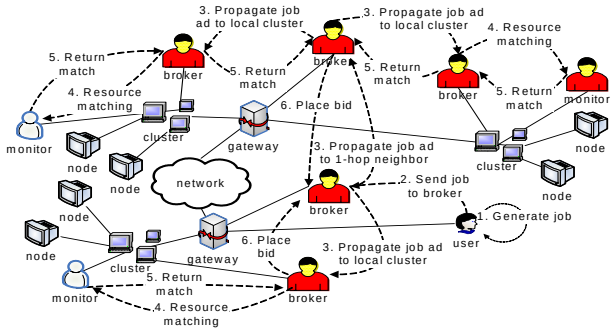


Figure 1. The Bidding Process

4. Bid Based Grid Simulator

To evaluate our bidding based grid job scheduling we have designed and implemented a discrete event grid simulation tool in C++ (Figure 2 shows the architectural layout). The simulator is designed to follow a hierarchical approach, where a network of routers is created, followed by subsequent gateways, clusters, and nodes. Each gateway consists of a gateway broker (serving as a representative broker for the entire gateway), a monitor (for monitoring all resources within the gateway), and users (generating job requests). At each cluster, brokers are used to perform inter-cluster brokering; monitors are used to monitor resources within the cluster (and reporting to the gateway); and nodes represent individual computers (with the available resources to the cluster). In order to simulate a more realistic grid environment, various random types of cluster and node instantiation schemes have been adopted to generate sites. Within each site, overall resource status is monitored by the gateway monitor, while the cluster monitor keeps track of all local nodes' types and job schedules, as well as currently used resources within the cluster. To create a scenario and study the behavior of the simulation under full load, the user object is configured to constantly provide a steady stream of new jobs which keeps the broker busy. Jobs are generated following a Poisson arrival process, with job execution time ranging from 1 hour to 5 days (based on a 2.4GHz processor).

During the bidding process, job advertisements will be submitted to all neighbors residing one hop away from the requesting host. Note that number of job advertisements propagated is directly related to the number of neighbors. This is done deliberately to limit the flooding of network by performing unnecessary broadcasting of job advertisements. To better simulate the effects of network delays in our simulation, we have included the effects of network processing (assumed constant), queuing, transmission, and propagation delays. Hence, the time taken and cost of data propagation depends on the queue length, data size and distance propagated.

4.1. Resource management

In our computational grid model there are two main entities: requesting brokers and tender brokers. The requesting broker collects jobs to be processed from users and is willing to reward a cluster for successfully completing the job within its timeframe and/or budget. The tender broker offers its service and resources to anyone who has jobs to be executed if the price is right. These two entities control the supply and demand of the grid resource market. The requesting broker is always in search for the least expense way to process its jobs, while getting results at the earliest-time. The tender broker is always trying to maximize its earnings while minimizing usage of its own resources so as to have more available resources to process other jobs. As a result, efficient resource management becomes self motivational as each participant is always trying to get the best "bang for the buck" while being thrifty in its own resource usage.

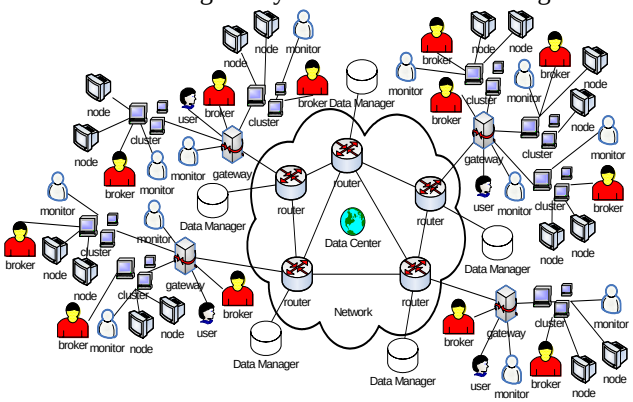


Figure 2. Simulator architectural view

4.2. Price formulation

In order to mimic the variations in the computational grid system load throughout the day, a simplified model has been adopted to be used as a reference to predict the usage of resources at different times of the day. Figure 3 depicts the simplified model used to predict system load throughout the day over a 2 day period. This model represents an estimated system load throughout a day; thus we will be able to estimate the total load between two given times of the day by integrating from the start time to the end time by referring once again to Equation 1. This integration result $\rho_{system_estimated}$ is then multiplied with the respective costs of all the resources required (ψ, ζ) by that particular job over the period of estimated execution time. After obtaining the initial estimated cost, this information is used in conjunction with the reserved resources (within that cluster) for all jobs scheduled to be executed during the same period to estimate the load on that particular cluster $(\rho_{cluster_estimated})$. The estimate of the overall resources used at that cluster during the execution period is then divided by the total resources available at the cluster, which is compared to the same fraction when the cluster is 50% utilized. The resulting difference is used to scale the initial estimated cost of processing that job, arriving at the price to charge for that job $(\upsilon_{cluster})$. For example, if initial estimated system price is 2000 units, and the estimated overall cluster utilization during the job execution period is calculated to be 70%, then the price would be scaled up by 20% to 2400 price units. In order to evaluate the effectiveness of this scheme, two other schemes have been implemented as bases for comparison:

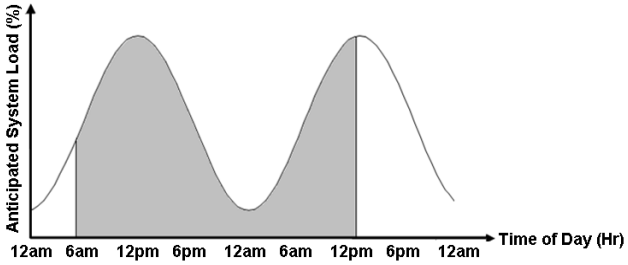


Figure 3. Grid Anticipated System Usage Model

- The first is a conventional grid scheduling scheme (denoted as *conventional* from here on) similar to that of the ATLAS project [9]. Instead of manual site selection by the administrators, this scheme performs random site selection for job execution. This helps in distributing the load of job executions across the system. Note, that clusters and nodes within each site in the conventional scheme are heterogeneous. By simulating clusters with varying resources, we can improve on the number of successful resource matches and hence serve as a more competitive comparison with our proposed bidding scheme.
- The second scheme (denoted as *random* from here on), is similar to the *conventional* scheme, but with the additional ability to perform renegotiations to find a better resource match within a particular site.

5. Simulation Results

Simulated data from the three previously described schemes were collected and compared. Data collected from the *conventional* and the *random* schemes are used for comparison to our proposed *bidding* scheme. Throughout its lifetime, a job may be in one of several stages in its execution phase: queued, staging-in (transferring the required input data sets to the executing host), executing, staging-out (transferring the resulting data sets to the designated receiving host), and *successfully executed*. A job is considered to have *successfully executed* only upon completion of the actual job execution process, with the staging-out of resulting data files delivered. In order to simplify the job stage categorizing process, we have separated them into 2 categories: *successfully executed* and *successfully matched*. *Successfully matched* implies that the necessary matching and bidding process have been completed and the job may be in any one of the previously mentioned stages other than *successfully executed*. However, a job match may fail and be marked as a *match-failed*. *Match-failed* reasons include: i) failure to match the job resource requirements to an available host; ii) failure to meet the budget requirements set forth by the requesting broker; iii) failure to meet the execution deadline; iv) failure to place bids prior to auction closing; or v) failure to sign and return the contract issued by the requesting broker.

In Figure 4, total number of successful renegotiations achieved for the random and the bidding schemes are presented. When a job match fails, renegotiation is performed between the requesting broker and tender broker. Default settings for the renegotiation process in our simulation limits the variation from the original time and budget to 10%. As observed, the random scheme fails to take full advantage of the renegotiation tools due to the limited availability of resources within the single randomly selected site. If the chosen site is already overwhelmed with jobs, further renegotiations may not necessarily improve the matching of jobs to resources. At this point, it is important to note that although the bidding scheme achieved approximately twice as many successful renegotiations as that of the random scheme, it does not imply that the number of jobs *successfully executed* by the bidding scheme performed twice as well as the random scheme. Successful completion of jobs still depends on other factors such as network traffic, data and output file staging processes, and resource availability at the moment in time when jobs are to be executed.

Figure 5 shows the accumulated monetary transactions throughout the simulation. The bidding scheme performs

significantly better than both *random* and *conventional* schemes due to the profit maximizing characteristics of the bidding scheme, along with substantial improvement in the number of successful job matches due to re-negotiations (see Figure 4)Although the random scheme also supported renegotiation, it is limited by its number of available resources from its single renegotiating location.

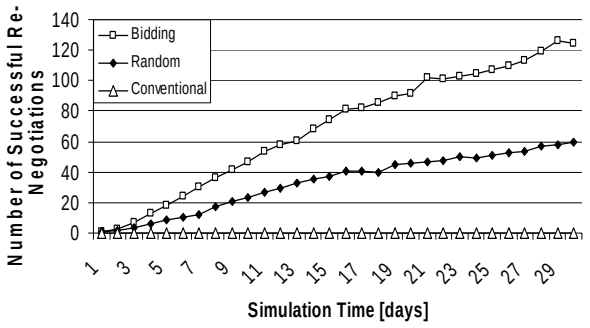


Figure 4. Total number of successful renegotiations

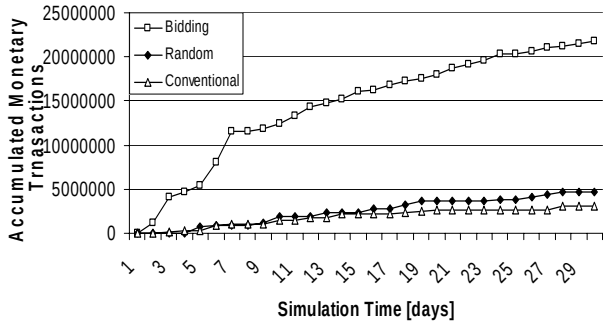


Figure 5. Accumulated monetary transactions

The current penalty implementation in our simulations is simple: if a site fails to abide by the contract, 30% of the initial contract price is returned to the requesting broker as penalty. From Figure 6, penalty from the bidding scheme remained within 14% of overall earnings throughout the simulation process. The conventional scheme, due to its limited number of matched jobs (and hence limited incurred penalties), performed better in terms of penalization charges, when compared to the random scheme (within 18% of its accumulated earnings).

6. Conclusions and Future Work

In this paper we have presented a bidding based grid resource scheduling framework. Our experimental studies showed that the *bidding* scheme performed better than two other, more conventional schemes. In order to take full advantage of the bidding scheme with job advertisement propagation, additional features such as re-negotiations can further enhance the performance. In this study we have used a simple, flat 30% penalty scheme. Our intention is to implement a flexible penalty scheme with demerit scheme where winning brokers who break the contract for any reasons will have a demerit point added to the rejecting broker for either backing out of a winning bid, or failing to abide by the contract. This would in turn affect their ability to participate in subsequent future biddings. We are also looking into improving the current scheme to incorporate multiple-hop job advertisement propagation ability, support for multiple auctioning and better decision-making tools, along with ability to subcontract jobs as well as better job QoS and prioritization support.

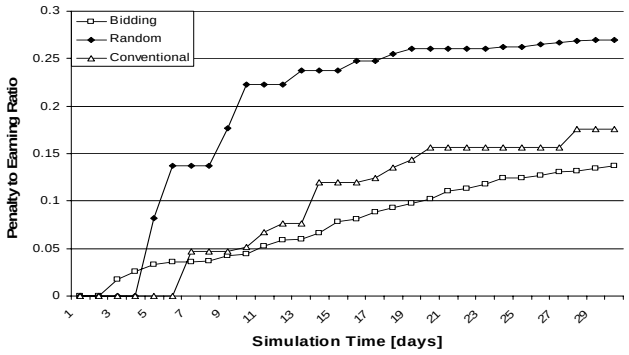


Figure 6. Penalty to earning ratio

References

[1] C. Liu, I. Foster, "A Constraint Language Approach to Grid Resource Selection," In Proceedings of the Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12), June 2003.

[2] C. Liu, I. Foster, "A Constraint Language Approach to Matchmaking", presented at the 14th Int. Ws. Research Issues on Data Engineering, Boston, MA, 2004.

[3] V. K. Naik, C. Liu, L. Yang, J. Wagner, "On-line Resource Matching in a Heterogeneous Grid Environment"

[4] R. Buyya, D. Abramson, S. Venugopal, "The Grid Economy", Proceedings of the IEEE, vol. 93, no. 3, pp. , March 2005.

[5] C. S. Yeo, R. Buyya, "Pricing for Utility-driven Resource Management and Allocation in Clusters", in Proc. of the 12th Internl. Conf. on Advanced Comp. and Communication, Ahmedabad, India, Dec.2004, pp32-41.

[6] L. Xiao, Y. Zhu, L. M. Ni, Z. Xu, "GridIS: an Incentive-based Grid Scheduling", 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)

[7] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical Recipes in C, Second Edition", Cambridge University Press.

[8] A. Sulistio, R. Buyya, "A Grid Simulation Infrastructure Supporting Advance Reservation", in 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), MIT Cambridge, Boston, USA, ACTA Press, CA, USA, November 9-11, 2004.

[9] ATLAS project, http://www.usatlas.bnl.gov/