A Highly Available Job Execution Service in Computational Service Market[1]

Woochul Kang, H. Howie Huang, and Andrew Grimshaw

Computer Science Department, University of Virginia

Charlottesville, VA, 22904

{wk5f, huang, grimshaw}@cs.virginia.edu

***Abstract*— One of the major challenges in managing resources of computational Grids with diverse shared resources is how to meet users' QoS requirements and rationally distribute resources at the same time. In particular, even though less reliable desktop PCs are dominant resource providers of computational Grids, they are often underutilized because they do not exhibit qualities required by typical scientific and business applications targeting computational Grids. Economy-based markets are expected to foster the utilization of those underutilized low quality resource via supply-and-demand. However, our experiment shows that price has its limitation in controlling the supply-and-demand in computational markets. This situation necessitates Highly Available Job Execution Service (HA-JES) which fosters the balanced resource consumption by dynamically and transparently replicating jobs with underutilized and under-priced resources. In particular, the process of job replication in HA-JES occurs in market-driven efficient way; underutilized and therefore cheap resources are exploited to build a high quality resource and hence facilitate balanced resource usage. Our simulation results show that HA-JES benefits all actors in the Grid market in terms of resource utilization, market capacity, and market stability.**

I. INTRODUCTION

Computational Grids consists of diverse computing resources ranging from cheap and less available desktop machines to highly available and high performance clusters and super computers. One of the major challenges in managing these diverse resources is how to meet users' QoS requirements and rationally distribute resources at the same time. If users' QoS requirements are biased toward some specific properties, for example, all users want 99% reliable resources for their 24 hour jobs, then the resources which meet those requirements can be over-utilized while the rest of them are idle and underutilized. This kind of imbalanced resource usage pattern is often found in many computational Grids.

A market-based computational economy [6] has been proposed for effective resource management, where pricing of a resource and limited "money" of a user works as a feedback to enforce users' rational behavior. In market-based resource management, computational resources are allocated through markets where the cost of using a resource is determined by supply and demand; Underutilized resources will have lower prices which are expected to foster more usage of them and conversely over-utilized resources will have a high price which in turn discourages use. However, a resource which does not meet certain requirement of a user is rarely useful regardless of its price. For example, a resource which guarantees only 50% reliability for a 24-hours job is rarely useful for a user who wants 99% reliability guarantees. This imbalance between resource supply and demand is inherent unless the distributions of users' quality requirement and quality of real resources are perfectly matched. We assert that this imbalance can not be resolved by the pricing mechanism alone in the market when the gap between what the market supplies and what users want is big.

We assume job execution environment like Condor [15] which deals with the execution of idempotent jobs with diverse shared resources ranging from a desktop PC to super computers. In such environment, satisfying the QoS requirements for jobs can be especially challenging. Even though less reliable desktop PCs are dominant resource providers of such systems, they are underutilized because they do not exhibit qualities required by typical scientific and business applications targeting computational Grids. Our result in Section IV shows that the pricing mechanism has its limitation in controlling mismatches between QoS requirements from clients and what the market supplies.

In this paper we describe a *highly available job execution service* (HA-JES) which dynamically and transparently virtualizes underlying low-level computational resources to meet imbalanced and unpredictable resource usage requirements. From the Grid user's perspectives, HA-JES is same to ordinary job execution service; it takes a job description and requirements from the user and executes the job if it can meet the user's requirements. From the architectural point of view, HA-JES is similar to resource broker as it acts as a mediator between a user and Grid resources. However, instead of merely brokering resources which meet the user's requirement, HA-JES actively composes underlying underutilized low-quality resources to build a high quality resource satisfying the user's requirement. In particular, the process of virtualization in HA-JES occurs in market-driven efficient way; underutilized and therefore cheap resources are exploited to build a high quality resource and

[1] This material is based upon work supported by the National Science Foundation under Grant No. 0426972. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

hence foster balanced resource usage. Even though there are many different QoS properties, performance, reliability, security, etc., we focus on the reliability of a computational resource because major concern in allocating a computational resource for a job execution is whether it will successfully complete without failure within a time bound. To increase the reliability of a virtualized resource, HA-JES replicates jobs in time and in space. The degree of replication and the raw resource selection process is determined by the market.

HA-JES is being implemented as an extension of *OGSA basic execution service* (OGSA-BES) [10] in the Genesis-II Grid platform [1]. Before deploying in the active Grid, we tested the impact of HA-JES to the Grid market using simulation with various workloads. Our experiment shows that HA-JES benefits all actors in the Grid market. For clients, HA-JES gives more choices in resource selection and gives higher chance of successfully finding a resource satisfying their QoS requirements. For resource providers, more revenues are earned with otherwise wasted resources. And finally for the Grid market as a whole, HA-JES increases the computational capacity of the whole Grid and this increased capacity results in desirable market characteristics such as price stability.

The rest of this paper is organized as follows. In Section II, we place HA-JES in context by examining relevant work in a number of related areas. In Section III, we describe HA-JES in detail. In Section IV, we present our simulation results which quantify the benefits of HA-JES in Grid market. Finally in Section V, we conclude with summary and future work.

II. Related Work

The work of this paper is based on two distinctive but related fields; replicated task execution and economical resource management.

A. Replicated Task Execution

Replication as a tool for high reliability in computer systems originates back to von Neumann's work [16], which schedules redundant copies of the tasks to increase the probability of having at least one of them successfully completes. Since then, along with checkpointing and roll-back-recovery [15], replication has been extensively used as a primary tool for reliable task execution.

Previous researches are mostly concerned with the management of replicas; how to make them have the same state and how to manage the group of replicas. In contrast to that, HA-JES is quite free from such issues occurring in traditional replicated job execution environment. This is mainly because the job execution service in HA-JES is idempotent. In HA-JES and job execution services based on OGSA-BES, the requirement of a job including input files required for execution is specified at the job submission time and copied in before the execution and the results are copied out after successful job execution. During the job execution, no interaction with outside entities is assumed. This makes HA-JES idempotent and consistency between replicas unnecessary.

Another important issue in replication is to decide the degree of redundancy [20]. In probability theory, the reliability analysis of a replicated job can be easily done if the probability of each replica completing is known [18].

However, the problem of previous approaches in determining the degree of redundancy is that they do not consider the cost of having redundancy. Because each replica takes up physical resource, the degree of redundancy should be controlled based on the overall resource usage status of the system and the value that the user gives on that job. In HA-JES, unlike previous approaches, the degree of redundancy is determined by the market state and values that the user puts on his job. If the system has abundant idle resources, then a higher redundancy may be affordable. But if resources are scarce, a user has to put more value on his job to get the same level of resource quality. In HA-JES, this is all determined by supply and demand in the market.

B. Economical Resource Allocation

Inspired by the efficiency and stability of real markets, computational economy has been studied as a metaphor for effective management of computational resources [6]. In market-based computational economy, unlike traditional approaches where resources allocation is driven by system-centric measures such as system utilization and throughput, each subject of the economy tries to selfishly maximize their utility and this leads to globally desirable resource allocation.

Several computational market models have been proposed for effective resource management, which includes Spawn [19], Tycoon [14], G-commerce [4] and Nimrod-G [5].

Among them Spawn by Waldspurger et al. is one of the earliest and most representative implementation of computational markets. In Spawn, resource allocation is determined by a sealed-bid and second-price auction. An idle resource holds an auction for the CPU time slots and gets bid from clients. Their experiment shows that funding rates are effective in achieving proportional-share resource allocation.

The work of Stonbreaker et al. [17] and Irwin et al. [12] are most similar to HA-JES where processing sites bids for servicing client's requests. Stonbreaker et al. proposed a market-based distributed database system whose name is Mariposa. In the system, each query is allocated a budget and this budget is used to buy a service and data fragments. A broker holds an auction on behalf of clients and resource owners bid for servicing that request. Irwin et al. considered the task execution service market as in our study; the task service sites make bid on client's request of task execution. In both works, a contract is made between service provider and clients on QoS assurance and price through negotiation. This service level negotiation and agreement procedure is an essential part in service oriented architecture [9]. HA-JES also follows this service negotiation and agreement model.

However, our approach is different from both Stonbreaker et al. and Irwin et al.'s works because HA-JES dynamically virtualizes low-quality resources when demand from users are different from what current market is able to supply. The primary goal of our study is to investigate the effectiveness of HA-JES in computational market.

Despite the large body of work, there has been little study on how to meet the user's QoS requirement when the quality distribution of available resources is quite different from users' quality requirements.

III. HIGHLY AVAILABLE JOB EXECUTION SERVICE

Because HA-JES exploits cheap and underutilized resources in computational service market, we first consider our computational service market.

A. Service Model

In a computational service market, clients request job execution on computing resources which are provided by the service providers or resource owners. Both clients and service provider have choices; clients can choose a service provider who executes a job reliably with the smallest charge; and service providers can choose a client whose job will maximize its revenue. Both parties negotiate to reach an agreement on the price and QoS levels which both parties will adhere to afterward. The major factors determining the successful establishment of contracts are the QoS level and the price. The price of using a service is determined by auction. First, a client sends bid requests specifying job requirements to a few eligible service provides. Service providers respond with a bid specifying the price that will be charged for the job and QoS level that it will assure and penalty on job failure[1]. A client selects the service provider who assures best QoS level at lowest and affordable price. The negotiation process can be supplemented by additional entities such as resource brokers and information services. The broker coordinates clients and service providers. The information service provides information required for entities to make a decision. The whole architecture of the computational service market is shown in Figure 1.

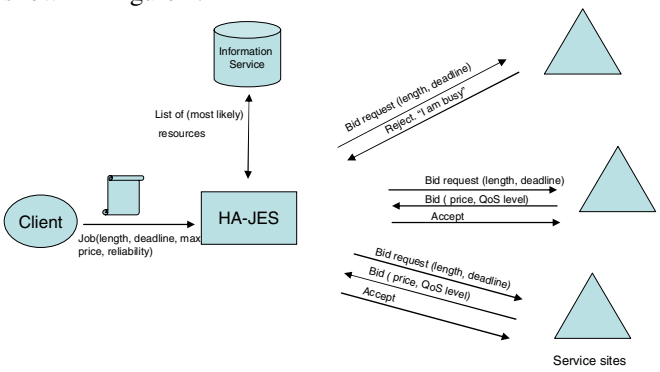


Figure 1. Computational service market.

We explain each components of computational service market in the next section.

B. Components of Computational Service Market

Clients

Job requests are submitted by clients. The form of a request is *(l, d, p, r),* where *l* is the estimated job execution time which is normalized to a canonical machine, *d* is the deadline, *p* is maximum price, and *r* is reliability requirement. This request says, "I'd like to run a job whose estimated execution time is *l* and should be finished by time *d*. And I want my job to be run at the service site who guarantees the reliability of *r* with less than price *p*". This request can be submitted either to a service site directly or to a broker. Once the job request is submitted and contract is established, then the client does not need to stay on-line; it can check the job progress using a unique identifier later.

HA-JES

A HA-JES instance is a broker that selects computing resources based on job requirements from the client. This instance holds an auction and sends a bid requests to multiple job execution service sites by forwarding the client request, *(l, d, r)*. The maximum price is masked out from the original job request to encourage service sites to assess the job request and reveal the true cost of job execution through bids. The list of most eligible service sites are obtained from an information service. Upon receiving the bids from the service sites, HA-JES compares bids to find a service site that meets the QoS level with minimum price. However, in many cases the matching may not be successful because service sites who meet the QoS level may demand a higher price than the client can afford. Instead of rejecting the job request, HA-JES may select multiple service sites whose suggested QoS levels and prices are less than client's requirement. After the resource selection, jobs are replicated to these low quality and cheap resources until either aggregate QoS level or total cost is no less than the QoS level, *r*, and maximum cost from the client, *p*. The detail of this process follows in section III-C. The actual job allocation can be done by HA-JES itself or by the help of other brokers.

Service Sites

Service sites execute a job that is requested from a client upon contract establishment. Upon receiving the bid request, the service site assesses the probability of meeting the deadline and profit from executing the job. When it is found profitable to execute the job, then a bid is made to the HA-JES. We assume that service sites reveal true valuation of the job through the bid and do not play games to win the auction.

Service providers can vary from single desktop to cluster computers. However, they provide the same service interface to clients and they are distinguishable only by QoS levels they provide and resource attributes such as architecture types.

Information Services

Information services [8] are core components in the Grid infrastructure. They provide diverse information and functionalities for Grid, which includes service discovery, resource monitoring, and resource characterization. Information services may support complex query interfaces to find resources satisfying constraints. In this study the

[1] In this work, we assume that service site is not penalized for job failures. For more detailed discussion on the penalty, see [12].

information service simply provides a random list of n live service providers that are willing to accept a new job.

C. Replicated Job Execution in HA-JES

The most notable difference of HA-JES from ordinary resource brokers is that it dynamically and transparently replicates a job execution using underutilized resources. The three factors that determine the degree of replication and the selection of service sites to run replicas are (1) the price of resources, (2) client's QoS requirement, and (3) the amount of money that a client is willing to pay for the job. The resource allocation algorithm in HA-JES is shown in Figure 2.

On arrival of job request *(l, d, p, r)*;
l:job length, *d*: deadline, *p*: maximum price, *r*: reliability

1. get the list of candidate resources from an information service
2. requests bids to candidate service sites by sending *(l, d, r)*
3. get bids (p_s, r_s) from service sites until time expires; p_s : price, r_s: probability of meeting the deadline
4. sorts bids in the ascending order of $func(p_s, r_s)$
5. iterate and add a resource *s* to target resource list until either
6. (a) $\sum p_s \geq p$
7. (b) $1-\prod(1-r_s) \geq r$
8. if $1-\prod(1-r_s) \geq r$, dispatch jobs to selected resources
9. otherwise, notify submission failure to the client

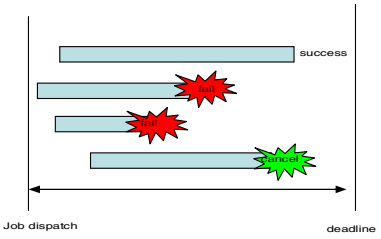
Figure 2. Algorithm for resource allocation in HA-JES.

The basic idea behind the algorithm is that if n resources are required to achieve the same level of reliability of a single resource, which is priced p, then the sum of the prices of n resources should be at most equal to p. Note that this can not be true without HA-JES. Without HA-JES resources unqualified for the client's QoS requirement can not be utilized and its price would be much lower or virtually zero if this situation continues for sometime. The low price of the underutilized resource should foster the active usage of the resource. However, as we will see later in Section IV this may not be the case.

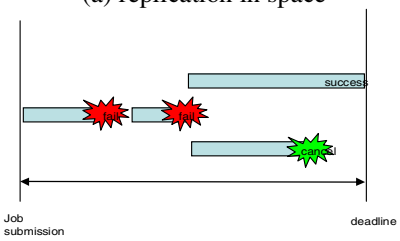
The first step of the resource allocation is to get bids. After getting bids, the "best" resource should be found to meet the QoS requirement and deadline with minimum cost. The "best" resource can be either a single service site or a group of service sites if a single site can not meet the deadline and QoS level with the available money. We should note that finding the best set of resources in this problem can be reduced to 0/1 knapsack problem, and the problem is well-known to be NP-hard [7]. Therefore, instead of trying to find the best set of resources, we resort to find a possibly-best set of resources using a greedy approach.

In our greedy approach, bids are sorted first in the ascending order of $func(p_s, r_s)$ where p_s is a price and r_s is the probability of meeting the deadline. The function can be defined in different ways with the different preferences. The function is defined as $p_s \bullet (1-r_s)^2$ in our study. By the sorting, the bids are possibly enumerated in the order of the smallest probability of missing the deadline with smallest price.

After sorting the bids, target resources are selected by iterating and adding the sorted bids one by one. The iteration continues until either the cost of using selected target resources are equal to or greater than available money, or the aggregate probability to meet the deadline is equal to or greater than the reliability requirement from the client. If the selected group of resources can meet both price and QoS level requirement, the job is replicated to the selected group of resources.



(a) replication in space



(b) replication in time and space

Figure 3. Replication strategies in HA-JES; *x*-axis shows the elapses of time and *y*-axis shows the allocation of replicated jobs to resources.

The possible problems with the above job replication approach, we call it replication-in-space-only hereafter, is that (1) a user can be charged more than necessary if one or more than expected number of replicas complete successfully, and (2) if the deadline is not imminent, this can incur unnecessary high resource utilization without further gain. The first problem depends on the charging policy; the client may or may not be charged against the utilization of the resource if the job does not complete successfully. In this paper, we assume a client is charged regardless of success or failure of the job. If charging policy is established so that only a successful completion is charged, then the line 6 of the algorithm should be changed to $(\sum p_s \bullet (1-\prod(1-r_s))) \geq p$, instead of $\sum p_s \geq p$; this implies higher redundancy is possible. The charging issue is out of scope of this paper. For more details on accounting and charging issue, readers are referred to [3]. The second problem incurs when the deadline in not imminent. If the deadline is long enough to try a job again when it fails, then redundancy in time is better in the view point of effective resource utilization and cost minimization.

HA-JES uses the combination of replication in time and space. If it is estimated that the deadline is far enough to do retries, then jobs are retried in sequence. When the job fails during the execution or is not completed in the expected time, then the job is restarted. When the deadline becomes imminent the job is dispatched in parallel to multiple resources. One disadvantage of this approach is that the

auction must be held several times. When an auction is held, HA-JES schedules the redundant executions based on the optimistic view that currently available resource will be available in the next time, but this assumption does not hold in many cases. HA-JES should hold a new auction to find currently available resources. We call this replication strategy as replication-in-time-space hereafter. The impact of having different replication strategies is shown in Section IV.

IV. SIMULATION

In order to evaluate the effectiveness of HA-JES in computational service market, we have conducted a number of simulations which served to quantify its ability to make use of idle resources, contribution on market stability, and total capacity of the computing pool. In our experiment, the changes of the market status during 180 days with and without HA-JES were observed and the results are shown in Figure 6~Figure 12. The replication-in-space-only strategy was used as default. The impact of using different replication strategies is presented in Section IV-D.

A. *Simulation Settings*

The hardest part in the simulation of a computational service market is how to simulate the behavior of each subject of the market including clients and service sites. The decision on buying and selling resources and setting the appropriate price on it depends on human behavior and this is hard to model and simulate.

Resource price determination rules for resources 1. If utilization for the past day is higher than threshold α_{high}, increase the price by μ_{high} % 2. If utilization for the past day is lower than threshold α_{low}, decrease the price by μ_{low} %
Purchase price determination from clients 1. If job success rate for the past day is higher than threshold β_{high}, increase the price by (average price of past day * ν_{high})% 2. If job success rate for the past day is lower than threshold β_{low}, decrease the price by (average price of past day * ν_{low})%
Workload determination 1. If job success rate for the past day is higher than threshold γ_{high}, increase the job submission rate by (average price of past day * ξ_{high})% 2. If job success rate for the past day is lower than threshold γ_{low}, decrease the job submission rate by (average price of past day * ξ_{low})% 3. If price for the past day is higher than threshold δ_{high}, decrease the job submission rate by (average price of past day * σ_{high})% 4. If price for the past day is lower than threshold δ_{low}, increase the job submission rate by (average price of past day * σ_{low})%

Figure 4. The control policies of the market entities. These rules are enforced on every monitoring periods.

In our simulation, the behaviour of each market entity is simulated by an agent who works on behalf of the entity. The decision of the agent is controlled by a few tunable policies. The market is monitored every day and each entity runs the control policy to control its behavior. Figure 4 shows some of the control policies. Note that the job arrival rate changes dynamically based on how many jobs were successfully submitted in the past day; this is based on the assumption that people are reluctant to submit their jobs overloaded Grid system because it will gives them very low probability of job success.

Figure 5 shows other simulation parameters. The reliability distribution of the resources follows bimodal distribution. One mode represents highly reliable resources and the other represents less reliable resources. The job length also follows bimodal distribution. One mode is for long jobs with the average of 12 hours and the other is for short jobs with the average of 1 hours. In the simulation, the computational service market ran for 180 days. These simulation settings including the ratio of high vs. low quality resources, the reliability distributions of resources follow the real resource characteristics which were found in our previous resource monitoring study [11, 13].

Parameter	Value
total number of resources	720
number of high quality resource	72 (10% of total resources)
number of low quality resource	648 (90% of total resources)
reliability of high quality resource	time-to-fail follows Normal(30,5) days
reliability of low quality resource	time-to-fail follows Normal(1,0.3) days
job arrival rate	controlled by the algorithm in Figure 4 to keep the Grid pool have 90% job success rate
length of long jobs	Normal (12, 3) hours
length of short jobs	Normal (1, 0.3) hours
ratio of long and short jobs	1:1
length of deadline	Normal(3 * job length, job length) hours
simulation period	180 days

Figure 5. Simulation parameters.

B. *Resource Utilization and Market Capacity*

The primary goal of the HA-JES is to foster the utilization of underutilized low quality resources. Even though a market is supposed to foster the utilization of those underutilized resources through pricing mechanism, our result shows that this can not happen without some mechanisms like HA-JES that replicates in space and/or time. Figure 9 and Figure 11 show the utilization and its price respectively over 180 days when HA-JES was not used. With the low utilization, around 3%, the price is getting lower and finally it is virtually zero. However, this low price can not foster the utilization. Both the utilization and price remains at low level during the entire experiment. This is because even though the price is low the low quality resources still can not meet the quality requirements from clients. On the contrary, the experiment shows that HA-JES enables the utilization of these

underutilized low quality resources. With HA-JES, the utilization of these low quality resources is almost 57% in average in Figure 11 and its average price is around 500. Compare the price of low quality resources and high quality resources. The price of high quality resources are around 1500 on average. This price ratio of low and high quality resources is very reasonable when we consider that HA-JES used 2.58 low quality resources on average to satisfy the quality requirement while a single high quality resource can meet the requirement alone.

The utilization of low quality resources which would be otherwise wasted, allows the increase of the total computational capacity of the Grid pool. Figure 6 shows how many jobs were submitted each day during the experiment period. In our experiment, jobs are submitted to the Grid as far as the Grid can accept and successfully complete jobs with 90% assurance. With the abundance of low quality resources, the total capacity of the Grid pool is about 8 times higher with HA-JES than without HA-JES.

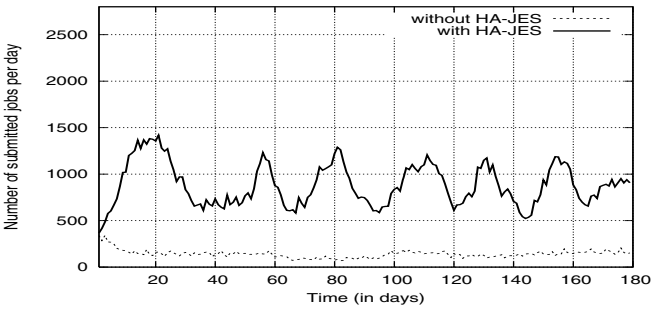


Figure 6. The number of job submission.

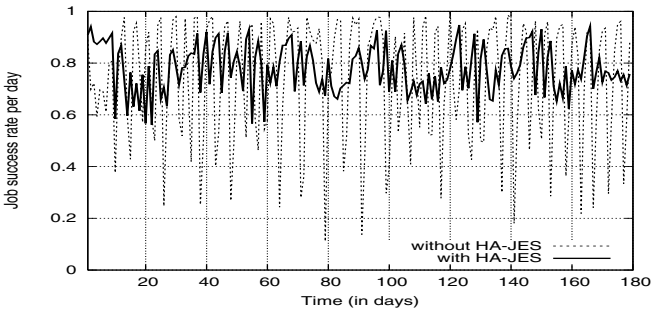


Figure 7. Job success rate.

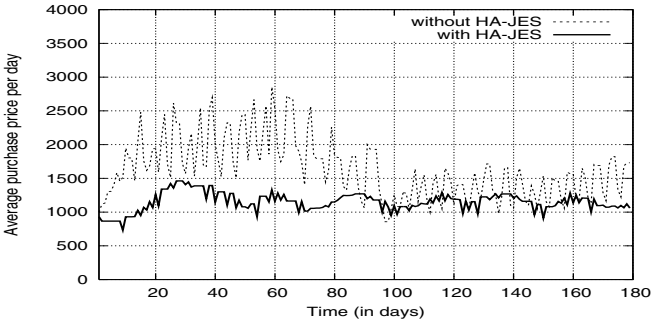


Figure 8. Purchase price.

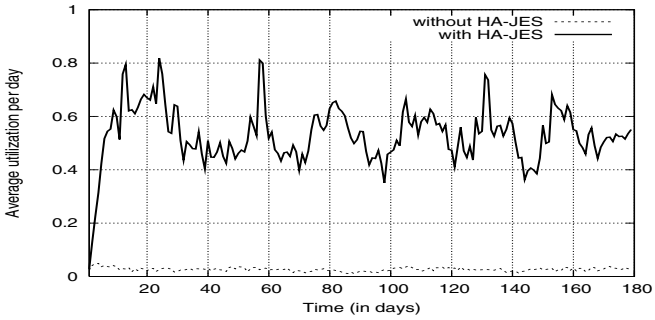


Figure 9. Utilization of low quality resources.

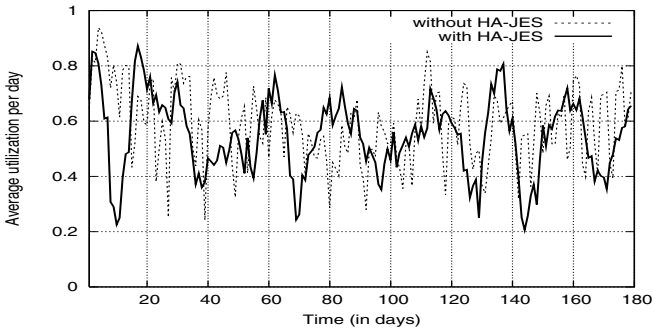


Figure 10. Utilization of high quality resources.

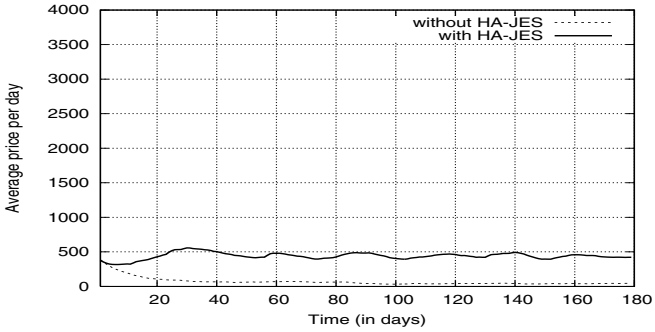


Figure 11. Price of low quality resources.

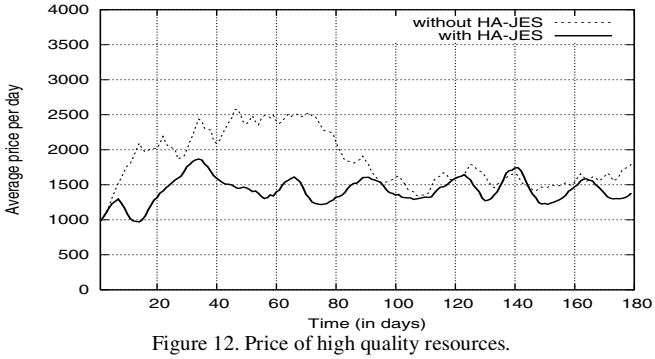


Figure 12. Price of high quality resources.

C. *Market Stability*

Another desirable effect that we expected from using HA-JES was market stability. In real world market, the price of resources can fluctuate easily if there are not enough resource providers; they can easily control the market and the price of the resource. With the abundance of resource providers and

higher freedom of choice, this situation can be resolved by the market.

Our experiment result conforms to our initial expectation as shown in Figure 7 ~ Figure 12. In Figure 8, without HA-JES, the purchasing price has higher deviation from equilibrium and takes longer time to stabilize than with HA-JES; Without HA-JES the overshoot is almost 1500 and takes 100 days to get stabilized. However, with HA-JES, the overshoot is about 300 and reaches the price equilibrium in about 30 days. This result is equally found in the selling price as shown in Figure 12.

D. *Comparison Between Replication Strategies*

The changes of the market status when HA-JES used replication-in-time-space are shown in Figure 13 ~ Figure 19.

We can identify the computational capacity increased more than 30% compared to replication-in-space-only strategy in Figure 13. In replication-in-time-space strategy, jobs are replicated on additional resource only if the current job execution fails, therefore each job execution requires less resources. This reduced resource consumption per job results in the surge of the total capacity. Note that even if the capacity is increased, the job success rate is same as the replication-in-space-only strategy.

We also found that the replication-in-time-space has better market stability. Both the deviation from the equilibrium and time to stabilize is smaller than replication-in-space-only strategy. Especially the price of high quality resources is more stable than replication-in-space-only strategy once reaches equilibrium. We believe that this market stability results from the increased computational capacity as Figure 13. This is similar to what we saw in Section V-C where increased capacity from using HA-JES made the market more stable.

In this experiment, we did not consider the cost of holding an auction. If this is considered, then replication-in-time-space can be disadvantaged for that.

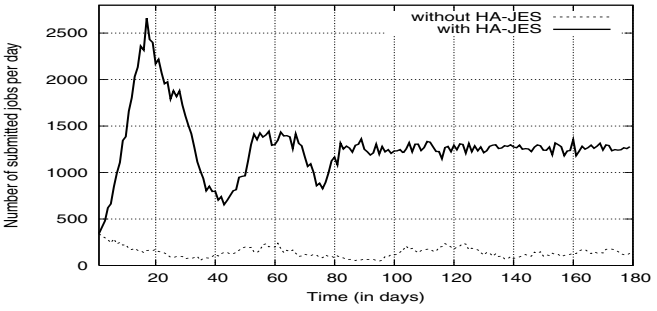


Figure 13. The number of job submission.

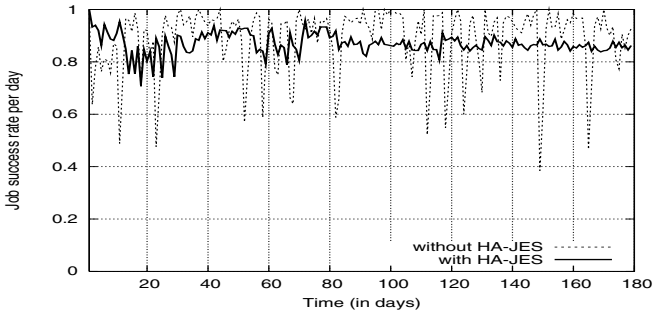


Figure 14. Job success rate.

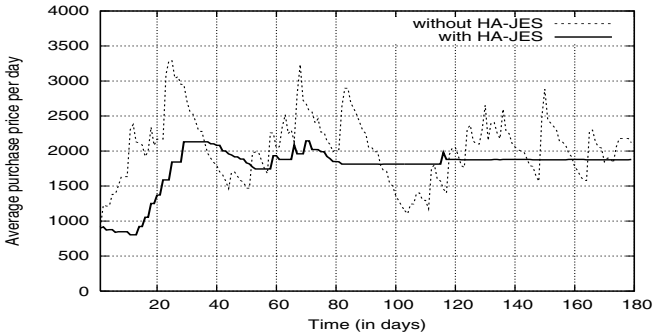


Figure 15. Purchase price.

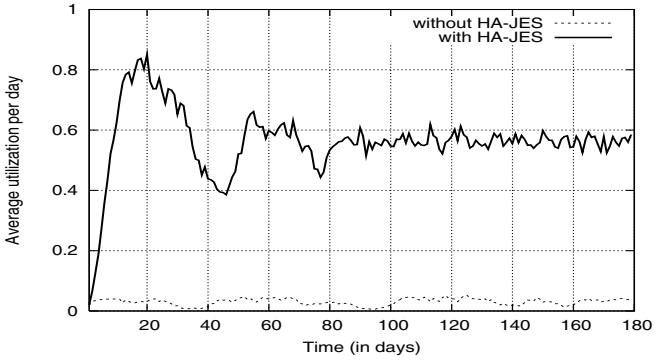


Figure 16. Utilization of low quality resources.

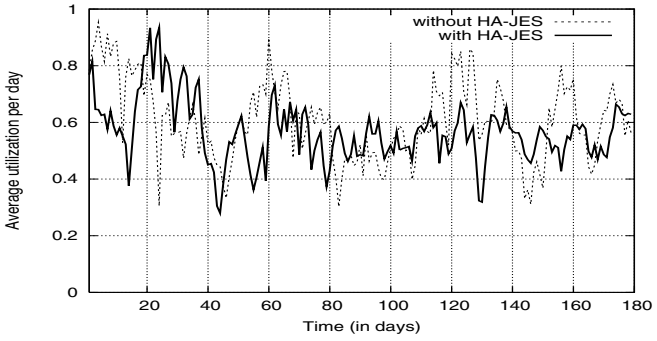


Figure 17. Utilization of high quality resources.

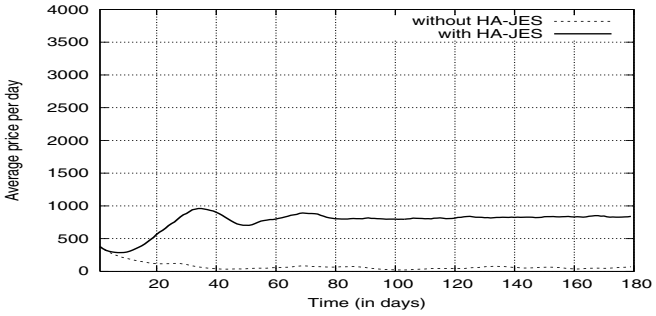


Figure 18. Price of low quality resources.

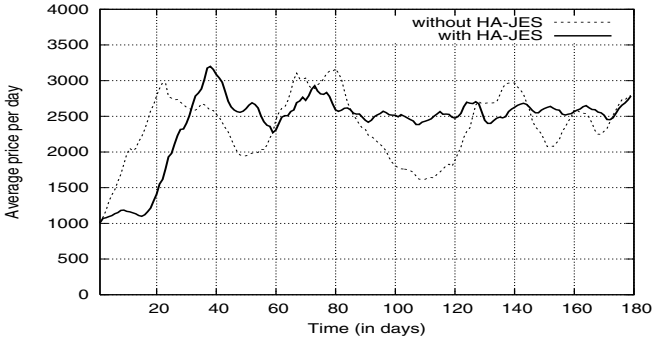


Figure 19. Price of high quality resources.

V. SUMMARY AND FUTURE WORK

Job execution service has been the basic building block of high throughput computing platforms such as Condor [15]. Even though inexpensive and low-quality resources are dominant constituent of those platforms, they are underutilized because they do not give enough guarantees on its quality. Economy-based markets are expected to foster the utilization of those underutilized low quality resource via supply-and-demand. However, our experiment shows that price has its limitation in controlling the supply-and-demand in computational markets. This situation necessitates the HA-JES which fosters the balanced resource consumption by dynamically and transparently replicating jobs to underutilized and under-priced resources.

Our experiment shows several benefits of having HA-JES in the job execution service market. For clients, HA-JES gives more choices in resource selection and gives a higher chance of successfully finding a resource satisfying their QoS requirements; for resource providers, more revenues are earned with otherwise wasted resources; and finally for Grid market as a whole, HA-JES increases the computational capacity of the whole Grid and this increased capacity results in desirable market characteristics such as price stability.

We next plan to integrate HA-JES into the Genesis II [1] grid system being developed at the University of Virginia. HA-JES will be integrated into the Genesis II implementation of the OGSA Basic Execution Services (BES)[10] activity factory. BES activity factories take activity documents as parameters. Each activity document contains a JDSL [2] document that describes the job (including optionally the amount of time the job will consume). Activity documents may also contain other sub-documents as extensibility elements. We will extend the BES activity document to contain a dependability document that specifies both the "price" the user is willing to pay as well as the reliability they require. HA-JES will allow us to make those guarantees.

REFERENCES

[1] *http://vcgr.cs.virginia.edu/genesisII/, Genesis II project homepage*, 2006.
[2] A. Anjomshoaa, F. Brisard, M. Dresher, D. Fellows, A. Ly, S. McGough, D. Pulsipher and A. Savva, *Job Submission Description Language(JSDL) Specification, Version 1.0*, Open Grid Forum, 2005.
[3] A. Beardsmore, K. Hartley, S. Hawkins, S. Laws, J. Magowan and A. Twigg, *GSAX:Grid Service Accounting Extensions*, GGF, 2002.
[4] R. W. a. J. S. P. a. J. B. a. T. Bryan, *Analyzing Market-Based Resource Allocation Strategies for the Computational Grid*, Int. J. High Perform. Comput. Appl., 15 (2001), pp. 258-281.
[5] R. Buyya, D. Abramson and J. Giddy, *Nimrod/G: an architecture for a resource management and schedulingsystem in a global computational grid*, *The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region*, 2000, pp. 283-289.
[6] R. Buyya, D. Abramson and S. Venugopal, *The Grid Economy*, Proceedings of the IEEE, 93 (2005), pp. 698-714.
[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Second Edition*, MIT Press, 2001.
[8] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, *Grid Information Services for Distributed Resource Sharing*, *HPDC*, San Francisco, CA, USA, 2001, pp. 181-194.
[9] I. Foster, C. Kesslman, J. Nick and S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Draft of 6/22/2002*.
[10] A. Grimshaw, S. Newhouse, D. Pulsipher and M. Morgan, *OGSA Basic Execution Service Version 1.0*, Open Grid Forum, 2006.
[11] H. Huang, J. F. Karpovic and A. S. Grimshaw, *A Feasibility Study of a Virtual Storage System for Large Organizations*, *1st IEEE/ACM International Workshop on Virtualization Technologies in Distributed Computing (held in conjunction with SC06)*, Tampa, Florida, 2006.
[12] D. E. Irwin, L. E. Grid and J. S. Chase, *Balancing Risk and Reward in a Market-based Task Service*, *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*, 2004.
[13] W. Kang and A. Grimshaw, *Failure Prediction in Computational Grids*, *40th Annual Simulation Symposium*, IEEE, Norfolk, USA, 2007, pp. 275-282.
[14] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang and B. A. Huberman, *Tycoon: an Implementation of a Distributed, Market-based Resource Allocation System*, HP-Labs, 2004.
[15] M. Livny and J. Raman, *High Throughput Computing Resource Management*, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Faufmann, San Francisco, California, 1999.
[16] J. v. Neumann, *Probabilistic logics and the synthesis of reliable organisms from unreliable components*, in C. E. S. a. J. McCarthy, ed., *Automata Studies*, Princeton University Press, 1956, pp. 43-98.
[17] M. Stonbreaker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah and C. Staelin, *An economic paradigm for query processing and data migration in Mariposa*, *3rd International Conference on Parallel and Distributed Information Systems*, 1994, pp. 58-67.
[18] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, John Wiley and Sons, 2001.
[19] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart and W. S. Stornetta, *Spawn: A Distributed Computational Economy*, IEEE Tranactions on Software Engineering, 18 (1992), pp. 103-117.
[20] F. Wang, R. Krithi and J. A. Stankovic, *Determining redundancy levels for fault tolerant Real-Time Systems*, IEEE Tranactions on Computers, 44 (1995), pp. 292-301.