Cost estimation of advance reservations over queued jobs: a quantitative study

Chunjiang Zhao[1], Junwei Cao[2,3*†], Huarui Wu[1,4] and Fan Zhang[3,5]

[1]*National Engineering and Research Center for Information Technology for Agriculture, Beijing 100097, P. R. China*
[2]*Research Institute of Information Technology, Tsinghua University, Beijing 100084, P. R. China*
[3]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, P. R. China*
[4]*School of Computer Science, Beijing University of Technology, Beijing 100022, P. R. China*
[5]*Department of Automation, Tsinghua University, Beijing 100084, P. R. China*

Abstract

A grid is a geographically distributed resource sharing environment across multiple organizations. The most typical grid resources are clusters with the high performance/cost ratio. In general, these clusters are shared as non-dedicated grid resources since local users may run their jobs simultaneously. Local jobs are usually queued and processed in a batch mode with uncertain waiting time while grid jobs always require advance reservations with guaranteed resource allocation.

In this paper, we provide quantitative analysis on the impact of advance reservations over queued jobs, in terms of job waiting time and resource utilization, respectively. It is observed that advance reservations will lead to longer job waiting time and lower resource utilization. That is to say, advance reservations should cost more than queued jobs. In this work, based on quantitative experimental results, an empirical formula for cost estimation of advance reservations over queued jobs is presented. It is suggested that compared with queued jobs, advance reservations should be doubly charged to compensate resource utilization loss. If the notice time of an advance reservation is short below a threshold, additional cost should be applied further since queue waiting time is increased.

Key Words: grid computing; cluster computing; job scheduling; advance reservations; cost estimation

1 Introduction

A grid environment provides uniform access to distributed computing resources that belong to multiple organizations [13]. The grid enables cross-domain resource sharing so that users from different organizations can collaborate with each other as a virtual organization (VO) [15]. Clusters are the most

* Correspondence to: Junwei Cao, FIT Building 3-415, Tsinghua University, Beijing 100084, P. R. China
† E-mail: jcao@tsinghua.edu.cn

popular computational grid resources that can achieve high performance with relatively low cost [3].

Traditional job scheduling on clusters is dominated by batch queuing mechanisms. Different local users can submit their jobs simultaneously. One job can be allocated with a number of required processors and a fixed period of time. Jobs are queued first-come-first-serve or according to different priorities. Some backfilling mechanisms are usually applied for small jobs arriving late or with lower priorities to start earlier so that better resource utilization can be achieved.

Grid computing brings new challenges on job scheduling issues. If a cluster is shared within a grid environment, it allows remote users to launch jobs locally. Most of these clusters are not fully dedicated to the grid, which means local users still want to use their own resources. On the other hand, remote users usually require some level of quality of services (QoS) support. Instead of queuing with uncertain waiting time as local users, grid users prefer to obtain guaranteed grid resources, which can be implemented using advance reservation mechanisms [22].

When local queues are combined with advance reservations for grid jobs, previous experimental results show that advance reservations will increase queue waiting time and decrease resource utilization [31]. That is to say, an advance reservation should cost higher than traditional batch queuing jobs. No previous work is focused on exactly how much higher an advance reservation required by grid users should cost. Our work presented in this paper is an initial effort to address this issue in a quantitative way and provide an empirical formula for cost estimation of advance reservations over queued jobs.

The rest of the paper is organized as follows. A summary of related work is included in Section 2; Section 3 provides a detailed description of job scheduling scenarios; In Section 4, the cost estimation method for advance reservations over queued jobs is proposed so that quantitative analysis can be carried out; Experimental results are illustrated in Section 5 with an empirical cost model proposed; The paper concludes in Section 6 with a brief future research plan.

2 Related work

There are many existing job scheduling systems for clusters, e.g. Condor [21], EASY [20], LoadLeveler [29], LSF [35], Maui [18], PBS [17], and Titan [32]. Batch queuing is the most basic job management approach that is supported by all these systems. For example, the Maui cluster scheduler is an open source job scheduler, capable of supporting multiple scheduling policies, dynamic priorities, reservations, re-negotiation and fairshare capabilities. Condor allocates free CPU cycles to queued jobs so that high throughput computing can be achieved. Titan utilizes the Genetic Algorithm to achieve more fine-grained queue scheduling for parallel jobs. All these systems have been developed for over a decade and some of them have been extended with commercial versions, e.g. PBS.

The grid is proposed in mid 1990s for cross-domain sharing of computational resources. Since a grid environment allows users to seamlessly access to multiple clusters, new mechanisms are developed for multiple cluster schedulers to work together. Globus [12] is the de facto standard implementation of grid computing. Globus resource management can integrate various cluster schedulers, e.g. Condor and PBS, and support advance reservations and co-allocation [14] so that a certain level of QoS can be achieved. Legion [16] is another famous software implementation for wide-area computing, supporting both system-level resource management and adaptability for user-level scheduling policies [8]. UNICORE [27] is developed in Europe as a Java grid implementation that provides client/server software packages for seamless access to distributed computing and data resources [26]. Titan is integrated with ARMS [5], an agent-based resource management system for grid computing, to implement grid load balancing [6], though no QoS support can be provided to individual jobs. A good summary of grid resource management can be found in [24].

Most clusters shared in a grid are not fully dedicated to grid users, since local users with queued jobs

may also compete for CPU cycles with grid users. In general, QoS support for grid jobs is implemented using advance reservations, which become essential for cluster schedulers. In [30] and [31] the impact of advance reservations on queue scheduling is investigated. There is no system implementation associated with the simulation study included in [30], while experimental results included in [31] is achieved using the Maui cluster scheduler. The work concludes that as the percentage of advance reservations increases, the overall resource utilization declines, which is also observed in our work. GridSim [33] is another grid simulation environment that supports advance reservations for repeatable and controlled evaluations, since evaluating various scenarios can not feasibly be carried out on a real grid environment due to its dynamic nature. Advance reservation can also be carried out in a negotiation-based way, as shown in [28], optimizing resource utilization and QoS constraints and generating contention-free solutions. Other research issues related to grid resource reservations include protocol design [19], scheduling algorithms [23], and virtual implementation [25].

In this work, we focus on cost estimation of advance reservations to achieve grid job QoS support, compared with local queued jobs. This is somehow related to the research topic, so-called grid economy [4], a metaphor for effective management of resources and job scheduling. For example, a market based resource reservation system is developed in [2] that utilize a trustworthy Vickrey auction to make combinatorial allocations of resources. In the work described in [34], revenue management is proposed to determine pricing of advance reservations in order to increase profits, where prices are periodically updated according to resource demands. A latest summary of similar work and future trends is included in [10].

Instead of market driven as mentioned above, we use a different approach for cost estimation of advance reservations. Several basic principles are concluded from quantitative experimental results, regarding on the impact of advance reservations on queued jobs, in terms of both resource utilization and job waiting time. This is described in Sections 3 and 4 and investigated in Section 5 using a lightweight implementation of a cluster scheduler and a simulated workload.

3 Job scheduling

In this section, several job scheduling mechanisms are illustrated. These are all involved in cost estimation of advance reservations described in Section 4.

3.1 Queue scheduling

Queue scheduling considered in this work is based on the first-come-first-served policy. In general, jobs are parallel programs using MPI or PVM interfaces, which require multiple CPU resources to execute. Local users submit jobs for queuing by specifying the required number of CPU nodes and time duration. The system queues jobs one-by-one according to arrival time. This is illustrated in Fig. 1 with 4 jobs and 6 CPUs.

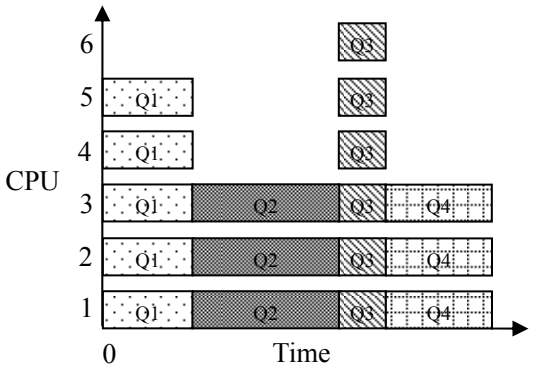

Fig. 1 First-come-first-served queue scheduling

In Fig. 1, the job *Q1* is the currently running job using 5 CPUs. As no enough CPU resources are available, the job *Q2* has to wait until *Q1* is finished. Jobs *Q3* and *Q4* also have to wait in the queue for future resource allocation. There could be additional support to the queue scheduling scenario described in Fig. 1. Users may want to select nodes by providing a list of preferred or disallowed nodes when submitting a job. Especially when nodes are heterogeneous, they can be classified into different node sets so that users are allowed to specify preferred node sets. Also job scheduling can support multiple queues with different priorities. There can also be job priorities that are associated with corresponding users. A user can have different priorities in different queues. For example, a grid user may be deferred in day and preferred at night. In our work, cost estimation of advance reservations are investigated using several statistics where job and queue priorities are not major concerns, these features are not included in our implementation. Another mechanism, called backfilling, which is very essential to improve resource utilization, is considered in our work and discussed below.

3.2 Backfilling

While the basic queue scheduling policy is first-come-first-served, the backfilling mechanism can be utilized to improve resource utilization and decrease queue waiting time, as illustrated in Fig. 2.

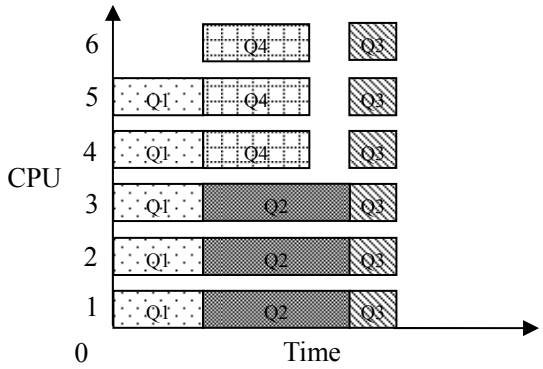

Fig. 2 Queue scheduling with backfilling

In the scenario described in Fig. 2, though the job *Q4* arrives later than *Q3*, there exists an opportunity that if *Q4* is queued before *Q3*, *Q3* would not be delayed and meanwhile some previously idle CPU resources are utilized. Later jobs can be executed first as long as enough resources are available and no other queued jobs are delayed. This backfilling mechanism is obviously essential to improve resource utilization without increasing queue waiting time. In our work, backfilling is applied to queue scheduling when we investigate performance impact of advance reservations.

3.3 Advance reservations

Queued jobs do not have a fixed starting time that can be expected, which has difficulties in providing QoS supports. A grid job usually requires a certain level of QoS supports, for example, a fixed starting time, instead of waiting in a queue, to coordinate multiple related jobs at multiple sites. Advance reservations can be applied to guarantee resource allocation.

Compared with jobs shown in Fig. 2, an additional advance reservation *R5* arrives after *Q4*, as shown in the scenario of Fig. 3.

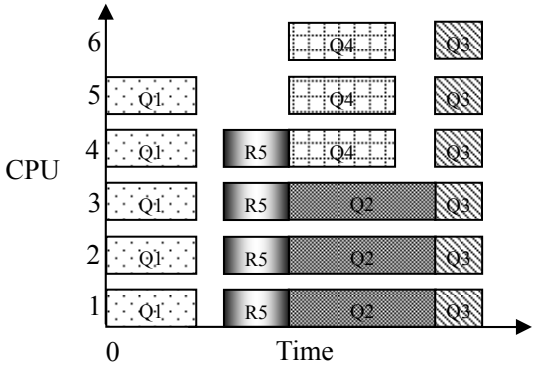

Fig. 3 Impact of advance reservations on queued jobs

R5 requires 4 CPUs with a fixed starting time, which has significant impact on queued jobs *Q2-4*. *Q2-4* have to be queued after *R5*, which increases queue waiting time. While there is still a time slot between *Q1* and *R5*, no jobs could fit in. Instinctively, advance reservations would decrease resource utilization.

In our work, we guarantee an accepted advance reservation to be started exactly at the scheduled starting time. This means a scheduled reservation will not be changed due to queued jobs. Sometimes users may be required to meet a deadline before which the job must be finished. In this case, advance reservation is scheduled to be completed exactly at the deadline (rejected if not possible) and later may be rescheduled to earlier possible CPUs. These are all implemented and deployed in the experiments of this work. Also some applications in a grid computing environment include QoS negotiation processes. An application may book resources for a job at multiple sites simultaneously and finally confirm one of them with the best QoS support. In this situation, cluster schedulers are required to support the two phase commitment. In a default mode, an advance reservation is booked and confirmed immediately if accepted. With additional supports, users can choose to book a reservation and confirm it later. If a reservation is not confirmed within a certain period of time, say 5 minutes, the schedule can be released automatically without actual execution.

As qualitatively shown in Fig. 3, advance reservations should cost more than queued jobs. In this work, we estimate cost of advance reservations using a quantitative method so that a empirical cost model can be proposed for advance reservations in comparison with queued jobs.

4 Cost estimation

In our opinion, main reasons that advance reservations should cost higher than queued jobs lie in the following two aspects:

- Advance reservations cause resource fragmentations that decrease queue scheduling efficiency and lead to lower overall resource utilization
- An advance reservation may require a very short notice time and take starting time advantages over

queued jobs.

Actually if ample notice time were given, it would be possible for a cluster to incorporate advance reservations without sacrificing queue efficiency. Our strategy for cost estimation of advance reservations is to identify the shortest notice time that should be given by advance reservations to avoid impact on queued jobs. Then cost of advance reservations with shorter notice time should be increased accordingly. Also even advance reservations with enough notice time that do not delay queued jobs can also decrease resource utilization. This has to be also taken into account for cost estimation.

In our work, cost estimation of advance reservations follow several principles concluded from quantitative results of queue scheduling combined with advance reservations. Two aspects of performance evaluation are considered in this work using different system configurations. Experiments are carried out using a lightweight implementation under a standard workload model. These are introduced in details below.

4.1 Performance metrics

Cost estimation of advance reservations is associated with the following performance metrics in our work:

- Mean waiting time, $w(h)$. The average amount of time that jobs have to wait before being executed.
- Resource utilization, $u(\%)$. The average percentage of CPU time that is utilized by jobs.

The mean waiting time can be applied to both queued jobs and advance reservations and we are particularly interested in the interaction between them. Resource utilization is a common performance metrics that is also used in other job scheduling research.

4.2 The workload model

A representative workload is important to evaluate job scheduling algorithms and policies. Too light or heavy workload may result into a situation where impact of different policies cannot be observed.

In this work, the workload used for all the experiments is the same and generated by the Cirne and Berman archive [9] of parallel workload models included in [11]. The Argonne National Laboratory (ANL) arrival pattern is applied to generate the workload file. The resource model is a 32-CPU cluster. For each experiment, 100 requests are generated that represent over one week's actual operation time. The submission time, required time and number of CPUs are retrieved from the workload file for each request. Experiments are also simplified by assuming that only one queue is supported, all users have the same priority, two phase commitment for advance reservations is not involved, and all cluster nodes are homogeneous and belong to one node set.

4.3 System implementation

System implementation includes a front-end scheduling daemon and node daemons. Daemons are implemented using C++ that perform job scheduling and execution. Command line user interfaces are provided to submit both queuing and reservation requests.

When the scheduling daemon is started up, basic configurations on queues and resources are loaded from a predefined XML file. Resource monitoring is carried out periodically to check CPU availability so that the scheduler can adapt to the scheduling space dynamically. Scheduling processes are logged and all information is restored in database. If the scheduling daemon crashes, errors can be traced and user requests can be recovered from log and database supports.

Job execution is guided by the scheduling information and responsible for actual allocating / releasing CPU access to the corresponding user when a job is started / finished. The system takes care of job execution by releasing resources for early completed jobs and terminating uncompleted jobs when

scheduled end time arrives.

4.4 System configurations

The main purpose of our experiments is to identify the shortest notice time for reservation requests so that impact on queue efficiency can be avoided. Since the impact of advance reservations on queue efficiency is evaluated using the mean waiting time of queued jobs, this statistics can be feedback for the runtime configuration of the notice time constraint for each reservation.

The main advantage is the decision making how much a reservation should cost can adapt to the workload of queued jobs in real time. For example, if the system load is very low, i.e. there are practically no jobs waiting in the queue, notice time constraints for reservation requests would not make much sense. On the other hand, if the queue is very long, advance reservations should be also postponed accordingly to avoid postponing execution of queued jobs. The drawback is that the statistics have to be recalculated every time the scheduling daemon receives a reservation request.

The implementation described above is straightforward. During each experiment, we randomly turn a certain percent of jobs in the workload model into advance reservations. The exact starting time of each reservation request equals to the shortest notice time plus the submission time.

More experiments are configured to investigate whether the mean waiting time of queued jobs can remain the same as if there were no advance reservations by applying a longer notice time for each reservation request. This results that the mean waiting time of advance reservations will be longer than that of queued jobs so that postponing advance reservations can compensate the loss of queue efficiency and result in a shorter queue waiting time.

In summary, each experiment is carried out using the same workload with two parameters configured:

- Percentage of advance reservations, $p(\%)$. The percentage of advance reservations is increased from 5% to 20%.
- Shortest notice time of advance reservations, n. This is represented as a multiple of the mean waiting time of queued jobs. Given a certain p, n is increased to reach a value when the corresponding notice time constraint can lead to the mean waiting time of queued jobs is as short as if there were no advance reservations involved.

After each experiment, two performance metrics included in Section 4.1 are evaluated and corresponding experimental results and cost estimation suggestions according to these results are included below in the next section.

5 Experimental results

The first experiment is carried out without advance reservations (p=0). The mean job waiting time in this situation, which is 46 hours, is taken as a reference value. Also the resource utilization is 74% when no advance reservations are involved. Further results when different percentages of advance reservations are involved are compared with these values. Statistical results are illustrated in Figures 4 and 5. Below we discuss the results in details.

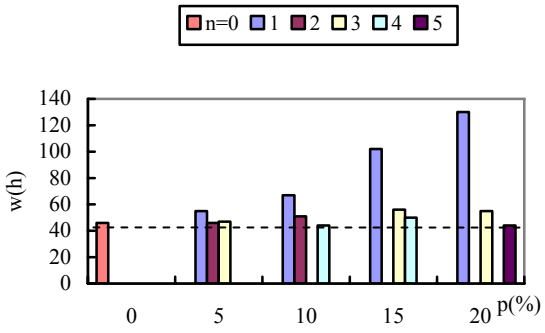

Fig. 4. Experimental result 1: mean waiting time of queued jobs (*w*) against different percentages of advance reservations (*p*) with different shortest notice time of advance reservations (*n*) configured

5.1 Mean waiting time

The mean waiting time of queued jobs for each experiment is illustrated in Fig. 4.

It is apparent that the queue waiting time increases with the number of involved advance reservations. This effect is strongly related to the added constraints at the time dimension so that the queue scheduling efficiency is decreased. It can be also noticed that in the case where *n* equals to 1, which means advance reservations do not take start time advantages, the values of mean waiting time *w* are greater than that of *n*=0, given any percentage of advance reservations involved, which means queue scheduling efficiency is decreased in any case. For example, the mean waiting time of queued jobs is increased by almost 20% in the experiment where *n*=1 and *p*=5% and over 100% in experiments when *p* is increased to over 15%.

We are mainly interested in how much longer the shortest notice time should be applied to advance reservations for better queue efficiency. It is apparent that as *n* increases, *w* does decline in any case. On the one hand, when advance reservations are configured with longer notice time constraints and, thus, have to be started later, queued jobs may start earlier. On the other hand, longer notice time constraints somehow enforce that advance reservations are scheduled with larger intervals, which provides more opportunities for queued jobs to be scheduled between these reservation intervals. When more reservation requests are involved, it will be more difficult to improve the queue efficiency. For example, when 5% requests are advance reservations and the notice time constraint is configured with 2 times of mean waiting time of queued jobs, queue scheduling is already as efficient as if there were no advance reservations. But as shown in Fig. 4, in order to reach a similar efficiency when 15% requests are reservations, the notice time constraint has to be configured as long as 4 times of mean waiting time of queued jobs.

5.2 Resource utilization

As illustrated in Fig. 5, when more advance reservations are involved, the average resource utilization rate decreases. This is also observed in the work described in [31].

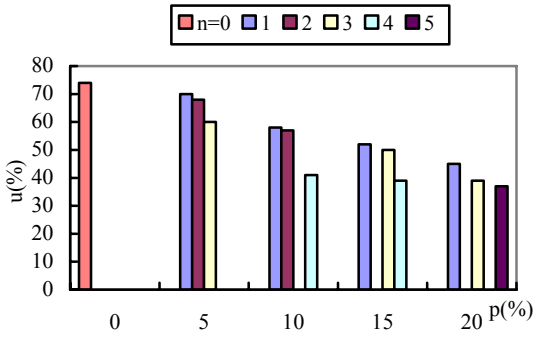

Fig. 5 Experimental result 2: resource utilization (*u*) against different percentages of advance reservations (*p*) with different shortest notice time of advance reservations (*n*) configured

It seems postponing advance reservations does not lead to an improvement of resource utilization but slightly reduces it, especially in the situation when advance reservations are largely postponed (e.g. $n>3$). However further investigation of the raw data of these experiments prove that the statistics cannot reflect situations of real world systems. Since experiments have an explicit end point in time, in situations when advance reservations are postponing largely, some of the advance reservations are scheduled to be started very late when all queued jobs are already finished. Thus no queue jobs are left that could utilize the last idle resources caused by these reservations. Since real world systems are continuously operational, there will be more requests that can continue to utilize these last idle resources. This issue is also discussed in the simulation study described in [30].

5.3 An empirical cost model

According to experimental results achieved in the above section, we provide suggestion on cost estimation of advance reservations.

According to results shown in Fig. 5, we find that overall resource utilization decrease ($\bar{u}$) is almost linear to the increase of the percentage of advance reservations, as illustrated in Fig. 6. For example, when there are 10% advance reservations involved, overall resource utilization will be decreased by 10%. And when the percentage of advance reservations increases to 20%, overall resource utilization will also decreased by 20%.

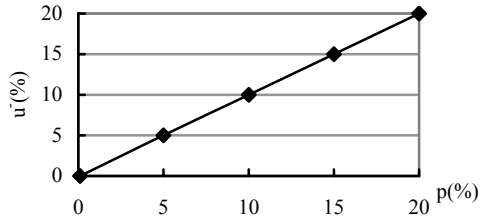

Fig. 6 Overall resource utilization decrease ($\bar{u}$) with different percentages of advance reservations (*p*) involved

From resource utilization point of view, the cost of an advance reservation is doubled, compared with a local queued job. A double charge has to be applied to an advance reservation so that the loss of resource utilization caused by the advance reservation can be compensated. For example, if a normal queued job costs 1, 100 such jobs would cost 100. If 20 of such queued jobs are turned to be reservations, since the utilization is decreased by 20% according to Fig. 6, the total cost would become 120. If 80 queued jobs is still charged 1 as normal, each of 20 reservations has to be charged 2 so that the total cost 120 can be

covered.

The cost of advance reservations should be also related to its impact on queue waiting time. When a reservation request arrives, there exists a shortest notice time (n^*) so that queue waiting is not prolonged. As explicitly illustrated in Fig. 7, the notice time threshold for advance reservations are represented as a multiple of mean waiting time of queued jobs and increase linearly from 1 to 5 as the percentage of advance reservations increases from 0% (not inclusive) to 20%.

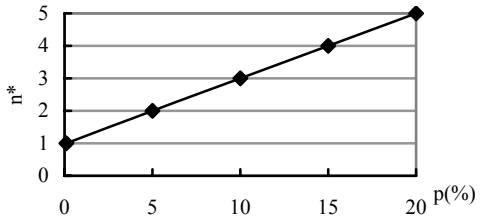

Fig. 7 Notice time thresholds for advance reservations (n^*) with different percentages of advance reservations (p) involved

For example, when 10% jobs are advance reservations, these reservations should give at least a notice time of 3 times of current mean waiting time of queued jobs so that queue waiting is not prolonged. In this case, no additional charge should be applied. *In terms of job waiting time, there exists a shortest notice time for an advance reservation with which the reservation does not prolong queue waiting and thus no additional charge should be applied. Otherwise if a shorter notice time is required, additional charge has to be applied.*

An empirical formula is provided below for cost estimation of advance reservations, c. Detailed results are also illustrated in Fig. 8

$$c = \begin{cases} \dfrac{0.4p+2}{n} & n \le n^* \\ 2 & n > n^* \end{cases} \qquad (1)$$

For each p, there exists a notice time threshold n^*, as shown in Fig. 7. If the required notice time of an advance reservation is larger than n^*, the cost would be double of that of similar queued jobs to compensate resource utilization loss. If a shorter notice time less than n^* is given, the cost would be higher.

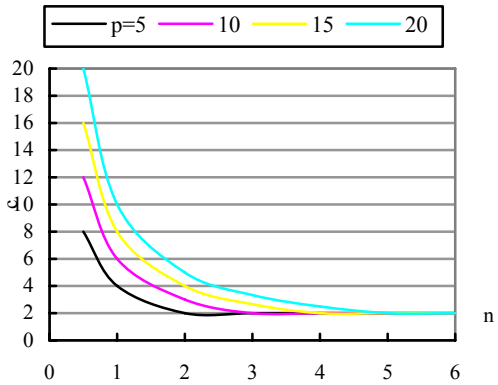

Fig. 8 Cost estimation of advance reservations: an empirical model

As shown in Fig. 8, the cost increases dramatically especially when notice time of advance reservations is shorter than current mean queue waiting time ($n^*<1$). For example, in a busy scenario where 20%

reservation requests are involved, a request with a notice time of half of mean queue waiting time is charged as highly as 20 times of a similar queued job. Since the reservation tries to take start time advantages, it is reasonable a higher charge is applied. In our empirical formula, it is impossible for a reservation to get resource allocation immediately without any notice time (n^*=0).

The empirical model included in the formula (1) and illustrated in Fig. 8 is only one example cost estimation. The resource owner can actually define exactly how much higher of the cost of an advance reservation with very short notice time over queued jobs according to his own preference. But it is suggested basic principles proposed in this work should be still followed, since are concluded using quantitative experimental results with standard scheduling algorithms and workload models.

In general, when the percentage of advance reservations is higher than 20%, it will be very difficult to carry out queue scheduling properly. This can be investigated by looking into experiment raw data. In such an experiment, reservation requests arrive so frequently that some large queue jobs cannot be scheduled in any interval of these advance reservations. In real world systems, these jobs would have no chance to be scheduled and will wait for ever. If many queued jobs starve, it means queue scheduling does not work at all. In the work [7], concepts from computational geometry are utilized to tackle resource fragmentations caused by advance reservations. This is beyond the scope of this paper.

There are of course many other factors that should be considered for advance reservation pricing, e.g. job execution time and market demands, etc. In the work described in [1], data parallelism is also considered for scheduling with advance reservations. Our work described in this paper only suggests a relative cost estimation method for advance reservations in comparison with queued jobs. Future work will incorporate more factors for cost estimation and pricing.

6 Conclusions

It is widely accepted that advance reservation of grid resources is a feasible way to implement QoS supports for grid jobs, especially some grid workflows may require to co-allocate related jobs on geographically distributed grid sites. It is also widely known that such reservations cost higher than traditional batch queuing jobs. No previous work is focused on exactly how much higher a advance reservation should cost than a local queued job. Our work try to address this issue in a quantitative way and provide an empirical formula for cost estimation of advance reservations.

In this work, quantitative study of impact of advance reservations on local job queuing is investigated first using a representative workload model and a lightweight implementation of cluster schedulers, in terms of both resource utilization and job waiting time.

From resource utilization point of view, a double charge has to be applied to an advance reservation so that the loss of resource utilization caused by the reservation can be compensated. In terms of job waiting time, there exists a shortest notice time for an advance reservation with which the reservation does not prolong queue waiting and thus no additional charge should be applied. Otherwise if a shorter notice time is required, additional charge has to be applied. Following these principles, an empirical formula is provided in this work.

Future work includes larger scale experiments for quantitative analysis and incorporating more factors in advance reservation pricing, using existing open source or commercial implementation of cluster schedulers.

Acknowledgement

This work is supported by National Science Foundation of China (grant No. 60803017) and Ministry of Science and Technology of China under the national 863 high-tech R&D program (grants No. 2008AA01Z118 and No. 2008BAH32B03).

References

1. Aida, K., Casanova, H.: Scheduling mixed-parallel applications with advance reservations. In: HPDC '08: Proceedings of the 17th International Symposium on High Performance Distributed Computing, Boston, pp. 65-74 (2008)
2. Bubendorfer, K.: Fine grained resource reservation in open grid economies. In: e-Science '06: Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam, pp. 81 (2006)
3. Buyya, R: High Performance Cluster Computing. Prentice Hall, New Jersey (1999)
4. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. Proceedings of the IEEE, 93(3), 698-714 (2005)
5. Cao, J., Jarvis, S.A., Saini, S., Kerbyson, D.J., Nudd, G.R.: ARMS: an agent-based resource management system for grid computing. Scientific Programming, Special Issue on Grid Computing, 10(2), 135-148 (2002)
6. Cao, J., Spooner, D.P., Jarvis, S.A., Nudd, G.R.: Grid load balancing using intelligent agents. Future Generation Computer Systems, 21(1), 135-149 (2005)
7. Castillo, C., Rouskas, G.N., Harfoush, K.: On the design of online scheduling algorithms for advance reservations and QoS in grids. In: IPDPS '07: Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, Long Beach, pp. 1-10 (2007)
8. Chapin, S.J., Katramatos, D., Karpovich, J., Grimshaw, A.: Resource management in Legion. Future Generation Computer Systems, 15(5), pp. 583-594 (1999)
9. Cirne, W., Berman, F.: A comprehensive model of the supercomputer Workload. In: Proceedings of the 4th IEEE Annual Workshop on Workload Characterization, 2001.
10. Dube, N., Parizeau, M.: Utility computing and market-based scheduling: shortcomings for grid resources sharing and the next steps. In: HPCS '08: Proceedings of the 22nd International Symposium on High Performance Computing Systems and Applications, Quebec City, pp. 59-68 (2008)
11. Feitelson, D.: Parallel workload models. http://www.cs.huji.ac.il/labs/parallel/workload/models.html.
12. Foster, I., Kesselman, C.: Globus: a metacomputing infrastructure toolkit. International Journal on Supercomputer Applications, 11(2), 115-128 (1997)
13. Foster, I., Kesselman, C.: The GRID: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1998)
14. Foster, I., Kesselman, C., Lee, C., Lindell, R., Nahrstedt, K., Roy, A.: A distributed resource management architecture that supports advance reservations and co-allocation. In: IWQoS '99: Proceedings of the 7th IEEE International Workshop on Quality of Service, London, UK, pp. 27-36 (1999)
15. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: enabling scalable virtual organizations. International Journal of High performance Computing Applications, 15(3), 200-222 (2001)
16. Grimshaw, A., Ferrari, A., Knabe, F., Humphrey, M.: Wide-area computing: resource sharing on a large scale. IEEE Computer, 32(5), 29-37 (1999)
17. Henderson, R.L.: Job scheduling under the portable batch system. In: JSSPP '95: Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing, 9th IEEE International Parallel Processing Symposium, Santa Barbara, California, USA, Lecture Notes in Computer Science Vol. 949, pp. 279-294 (1995)
18. Jackson, D., Snell, Q., Clement, M.: Core algorithms of the Maui scheduler. In: JSSPP '01: Proceedings of 7th Job Scheduling Strategies for Parallel Processing, ACM SIGMETRICS 2001, Cambridge, Massachusetts, USA, Lecture Notes Computer Science Vol. 2221, pp 87-102 (2001)
19. Kuo, D., Mckeown, K.M.: Advance reservation and co-allocation protocol for grid computing. In: e-Science '05: Proceedings of the 1st International Conference on e-Science and Grid Computing, Melbourne, Australia., pp. 164-171 (2005)
20. Lifka, D.: The ANL/IBM SP scheduling system. In: JSSPP '95: Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing, 9th IEEE International Parallel Processing Symposium, Santa Barbara, California, USA, Lecture Notes in Computer Science Vol. 949, pp. 187-191 (1995)
21. Litzkow, M.J., Livny, M., Mutka, M.W.: Condor – a hunter of idle workstations. In: ICDCS '88: Proceedings of the 8th International Conference on Distributed Computing Systems, pp. 104–111 (1988)
22. MacLaren, J., Sander, V., Ziegler, W: Advance reservations: state of the art. Grid Resource Allocation Agreement Protocol Working Group, Global Grid Forum (2002)
23. Min, R., Maheswaran, M.: Scheduling advance reservations with priorities in grid computing systems. In: PDCS '01: Proceedings of the 13th IASTED International Conference on Parallel and Distributed Computing Systems, pp. 172-176 (2001)
24. Nabrzyski, J., Schopf, J.M., Węglarz, J.: Grid Resource Management: State of the Art and Future Trends. Kluwer Academic Publishers, Boston (2004)

25. Nurmi, D.C., Wolski, R., Brevik, J.: VARQ: virtual advance reservations for queues. In: HPDC ’08: Proceedings of the 17th International Symposium on High Performance Distributed Computing, Boston, pp. 75-86 (2008)
26. Romberg, M.: The UNICORE architecture: seamless access to distributed resources. In: HPDC ’99: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing, pp. 44 (1999)
27. Romberg, M.: The UNICORE Grid infrastructure. Scientific Programming, Special Issue on Grid Computing, 10(2), 149-157 (2002)
28. Siddiqui, M., Villazón, A., Fahringer, T.: Grid capacity planning with negotiation-based advance reservation for optimized QoS. In: SC ’06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Tampa, Florida (2006)
29. Skovira, J., Chan, W., Zhou, H., Lifka, D.: The EASY-Loadleveler API project. In: JSSPP ’96: Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, 10th IEEE International Parallel Processing Symposium, Honolulu, Hawaii, USA, Lecture Notes in Computer Science Vol. 1162, pp. 41-47 (1996)
30. Smith, W., Foster, I., Taylor, V.: Scheduling with advanced reservations. In: IPDPS ’00: Proceedings of the 14th IEEE International Parallel and Distributed Processing Symposium, Cancun, Mexico, pp. 127-132 (2000)
31. Snell, Q., Clement, M., Jackson, D., Gregory, C.: The performance impact of advance reservation meta-scheduling. In: JSSPP ’00: Proceedings of the 6th Job Scheduling Strategies for Parallel Processing, 14th IEEE International Parallel and Distributed Processing Symposium, Cancun, Mexico, Lecture Notes Computer Science Vol. 1911, pp 137-153 (2000)
32. Spooner, D.P., Jarvis, S.A., Cao, J., Saini, S., Nudd, G.R.: Local grid scheduling techniques using performance prediction. IEE Proceedings - Computers and Digital Techniques, 150(2), 87-96 (2003)
33. Sulistio, A., Buyya, R.: A grid simulation infrastructure supporting advance reservation. In PDCS ’04: Proceedings of the 16th IASTED International Conference on Parallel and Distributed Computing and Systems, MIT Cambridge, Boston (2004)
34. Sulistio, A., Kyong, H.K., Buyya, R.: Using revenue management to determine pricing of reservations. In: e-Science ’07: Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing, Bangalore, pp. 396-405 (2007)
35. Zhou, S.: LSF: load sharing in large-scale heterogeneous distributed systems. In: Proceedings of Workshop on Cluster Computing (1992)