International Journal of Hybrid Information Technology
Vol.6, No.6 (2013), pp.51-64
http://dx.doi.org/10.14257/ijhit.2013.6.6.05

An Adaptive Redundant Reservation Strategy in Distributed High-performance Computing Environments

Peng Xiao[1*], Peixin Qu[2] and Xilong Qu[1]

[1]*College of Computer and Information Science, Hunan Institute of Engineering*
[2]*School of Information and Engineering, Henan Institute of Science and Technology*

[*]*xpeng4623@gmail.com, quxilong@sina.com, qupeixin@163.com*

Abstract

In distributed high-performance computing environments, resource reservation mechanism is an effective approach to provide desirable quality of service for large-scale applications. However, conventional reservation service might result in lower resource utilization and higher rejection rate if it is excessively applied. Furthermore, redundant reservation policy has been widely applied in many practical systems with aiming to improve the reliability of application execution at runtime. In this paper, we proposed an adaptive redundant reservation strategy, which uses overlapping technique to implement reservation admission and enable resource providers dynamically determine the redundant degree at runtime. By overlapping a new reservation with an existing one, a request whose reservation requirements can not be satisfied in traditional way might be accepted. Also, by dynamically determining the redundant degree, our strategy can obtain optimal tradeoff between performance and reliability for distributed high-performance computing systems. Experimental results show that the strategy can bring about remarkably higher resource utilization and lower rejection rate when using redundant reservation service at the price of a slightly increasing of reservation violations.

Keywords: *Advance Reservation; Quality of service; Redundant Policy; Co-allocation*

1. Introduction

In distributed high-performance computing environments, end-to-end QoS provision is often required by high-end applications [1]. Resource reservation, as an effective technique to support end-to-end QoS guarantees, has been incorporated into many famous middleware [2, 3], which allows applications to gain concurrent access to adequate resources, and guarantees the availability of resources to applications at the required times. Although reservation has been proven effective in many situations, it also brings several negative effects on resources sharing and scheduling. Studies in [4-16, 22-25] have shown that fixed-capability reservation will result in low resource utilization, and excessive reservation can lead to high rejection rate of requests. These inevitably have significant influences on utility-based computing environment [10], where systems wish to fully utilize their resources to obtain maximal profits with constraints of users' QoS requirements. Hence, how to mitigate the negative effects brought about by advance reservation becomes an important issue needed to be solved.

Traditionally, advance reservation is defined as a promise from systems that a subsequent resource allocation request will succeed [2]. Two key properties of a reservation are start time and deadline. Resource management systems promise and

ISSN: 1738-9968 IJHIT

only promise that the reserved resources will be accessible between start time and deadline. However, for many applications precisely prediction of these two parameters is difficult to achieve, if not impossible. As a result, applications tend to overestimate these parameters, especially the reservation deadline, to ensure their successful completion. This behavior results in two consequences: first, resource capability is underutilized; second, requests might be rejected due to their overestimation of deadline. In addition, redundant reservation request has been widely applied in many real-world systems for improving the reliability of resources. This mechanism tends to put extra overload on distributed resources. In addition, when the system applies redundant reservation policy for improving reliability, it will add more overloads on the target systems.

Motivated by aforementioned facts, in this paper, we propose a more flexible reservation strategy, namely Adaptive Redundant Reservation with Overlapping Strategy (ARROS), which allowing two reservations overlapping each other under certain conditions and enable systems can adaptively select the optimal redundant degree based on the dynamic workload. The objective of ARROS is to increase resource utilization and reduce rejection rate when using advance reservation. More importantly, we provide an adaptive mechanism by which the system can dynamically determine the redundant degree (often note as K) on per-application basis.

The rest of this paper is organized as follows. Section 2 presents the related work. In Section 3, we describe the traditional reservation model and analyze its limitations. In Section 4, we introduce our reservation strategy and analyze it using probability theory. In Section 5, massive experiments are conducted to verify the performance of this strategy. Finally, Section 6 concludes the paper with a brief discussion of future work.

2. Related Work

Since advance reservation was introduced into distributed resource management and scheduling, its effects on system performance have been widely studied. In [5], the authors investigated the impacts of advance reservation on performance of Grid scheduler. Three system metrics, including Mean Waiting Time (MWT), Mean Offset Time (MOT), and Request Rejection Rate (RRR) were used to quantitatively evaluate the impacts. Their experimental results showed: (1) using advance reservation will increase MWT and RRR; (2) Moderate reservation can lower down MOT, but excessive reservation will increase MOT too. In [7], the stimulation results also confirmed the conclusions in [5]. In [9], the authors studied the effects of advance reservation on remote jobs as well as local jobs in non-dedicated environment. A measure metric relative slowdown (the ratio of the mean waiting time with reservation and that without reservation) was introduced to quantify the impacts caused by reservation on local and remote jobs. By modeling distributed resources as M/G/1 FCFS queuing system, they formally proved that excessive reservation would prolong the waiting time of both local and remote jobs. All the above studies have shown that inappropriate reservation might result in low resource utilization and high rejection rate.

Besides above studies, many effective techniques have been proposed to overcome the limitations of advance reservation. In [4], the authors proposed an extended reservation architecture, which combining advance reservation and application adaptation together, to overcome the limitations of fixed-capability reservation. In this architecture, resources are enhanced with online control interfaces, sensors, decision procedures, so as to provide more efficient resource usages and deliver more robust application performance for high-end Grid applications. In [8], the authors incorporated

gang scheduling and adaptive resource allocation into SCOJO scheduler [11] to mitigate the negative effects brought about by advance reservation. In [12], the authors introduced several techniques, including re-arranging subtask, interweaving task graphs, backfilling, into advance reservation based scheduling in cluster environment with aiming to improve resource utilization. To lower reservation rejection rate and increase resources utilization, a flexible reservation window scheme is proposed in [18]. By conducting extensive simulations, the authors concluded that when the size of reservation window equal to the average waiting time in on-demand queue, the reservation blocking probability (rejection rate) can be minimized near to zero. In [19], a flexible reservation framework is implemented, in which slacking reservation mechanism is incorporated in a modular middleware. In [20], the authors developed an algorithm that allows service consumers to execute business workflows of interdependent services in a dependable manner within tight time-constraints. In [21], the authors studied an extension of the EMLM in order to ensure QoS guarantee per service-class in the heterogeneous environment of telecom networks.

3. Resource Reservation Model

At first, we describe the traditional reservation model, which conforms to GARA specification [2]. In resource reservation model, heterogeneous Grid resources are managed by *Reservation Manager* (RM), which performs admission control and tracks the reservations on all resources that under its control. All reservation requests are sent to RM. Each reservation request can be characterized by a 4–tuple: $< ts_l, ts_u, d_l, d_u >$, where ts_l and ts_u are the lower and upper bound of reservation start time, d_l and d_u is the lower and upper bound of relative deadline (the period between start time and absolute deadline). A reservation request is valid if $t \le ts_l \le ts_u$ and $0 < d_l \le d_u$, where t is the arrival time of the request.

On receiving a request, RM will try to find a time-slot, which is capable of meeting the request's requirement $< ts_l, ts_u, d_l, d_u >$. If a feasible time-slot is found, RM will respond the request with a 3-tuple: $< ts_i, te_i, d_i >$, where ts_i is start time of the reservation, te_i is the absolution deadline, and $d_i = te_i - ts_i$ is relative deadline. If the response is confirmed by the request, we say that a *reservation contract* has been successfully signed between the request and RM. Otherwise, we say that the reservation request is *rejected*. Given a reservation contract $< ts_i, te_i, d_i >$ between request r_i and RM, if the latter does not make the resource accessible for r_i at time ts_i, or fail to keep the resource accessible for request r_i until time te_i, we say that a *reservation violation* occurs. With the loss of generality, we make some assumptions of above reservation model as follows.

- Resource offers service to requests in the order of reservation start time.
- No resource is capable of performing service for two or more requests in parallel.
- If the job of a request has completed before its deadline, it will release the resource immediately. RM will preempt the resource from a request if its absolute deadline expires.

- Reservation violations caused by system crash, software fault, network disconnection, etc. are not taken into account in this paper.

4. Adaptive Redundant Reservation with Overlapping Strategy

4.1. Overlapping Reservation Strategy

Given current time is t_0, a time slot table of reservation for a resource is shown in Figure 1. Every reservation contract is represented by a 3-tuple: $< ts_l, te_l, d_l >$, which is illustrated by a rectangle with texture.

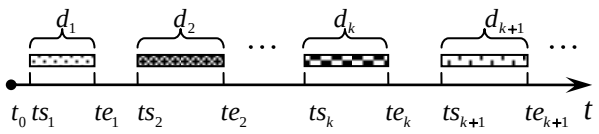


Figure 1. An example of time slot of reservation

As we can see that, the reservation split the entire time table into a lot of time slots, which can be categories into two groups: *reserved slots* and *free slots*. With arrival of new requests, some free slots may be allocated to those requests if the slot can meet their reservation requirements. However, there are still many free slots would never be allocated, which causing underutilization of resource. As mentioned before, requests usually tend to overestimate their deadline to ensure their successful completion. Our strategy takes this overestimation into account, and tries to insert some requests whose reservation requirements can not be met in traditional way into free slots. Obviously, this overlapped strategy will take some risks of violating those reservation contracts. So, our work is to analyze the benefits and risks of this strategy.

Following the scenario depicted in Figure 1, we hypothesize that RM adopts overlapping strategy. When request r_i arrives, RM finds that no available free slot can meet the requirements of r_i strictly. However, RM notices that a free slot between te_k and ts_{k+1} seems to be a good candidate, except that this candidate slot overlaps with a reserved slot of r_k, which means that reservation start time of r_i can not be guaranteed. If request r_k completes its job before ts_i, then RM will not violate reservation contract of r_i. The time slot table is depicted in Figure 2.

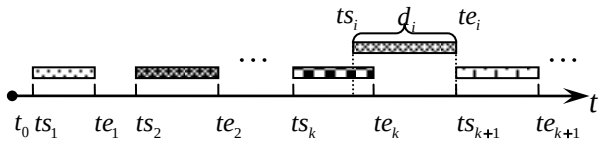


Figure 2. The example of overlapping time slot of reservation

Let random event $\mathbf{E_i}$ denotes no reservation violation occurs for r_i, so the probability of $\mathbf{E_i}$ can be expressed as

$$\Pr\{\mathbf{E_i}\} = \Pr\{TS_i = ts_i \bigcap TE_i \le te_i\} \tag{1}$$

where random variable TS_i represents the time when the resource is accessible to r_i, TE_i is also a random variable representing the completion time of r_i, ts_i and te_i are start time and absolute deadline of r_i. From the Figure 2, it is obviously that

$$\Pr\{TS_i = ts_i\} = \Pr\{TE_k \le ts_i\} \tag{2}$$

For reservation r_k, its start time can be guaranteed, which means $\Pr\{TS_k = ts_k\} = 1$. By using conditional probability theorem, we can obtain that

$$\Pr\{TE_k \le ts_i\} = \Pr\{TE_k \le ts_i \mid TS_k = ts_k\} \tag{3}$$

So, applying conditional probability theorem on formula (1), it can be rewritten as

$$\Pr\{\mathbf{E_i}\} = \Pr\{TE_i \le te_i \mid TS_i = ts_i\} \cdot \Pr\{TE_k \le ts_i \mid TS_k = ts_k\} \tag{4}$$

Formula (4) gives the probability of no reservation violation occurring if r_i is accepted. Clearly, $1 - \Pr\{\mathbf{E_i}\}$ is the risk of reservation violation. To evaluate (4), we only need to know the distribution of service time (running time) of requests. In contrast with predicting service time, obtaining their distribution is easier and more precisely. After knowing the risk of accepting r_i, we can also calculate the profits of doing that. Given C as the resource price under condition that no reservation violation occurs, F is compensation price if reservation violation occurs. If r_i is accepted, the expected profits P_i of RM can be estimated as

$$\begin{aligned} \mathrm{E}(P_i) &= C \cdot \Pr\{\mathbf{E_i}\} - F \cdot (1 - \Pr\{\mathbf{E_i}\}) \\ &= (C + F) \cdot \Pr\{\mathbf{E_i}\} - F \end{aligned} \tag{5}$$

The relationship between $\Pr\{\mathbf{E_i}\}$ and $\mathrm{E}(P_i)$ is described in Figure 3.

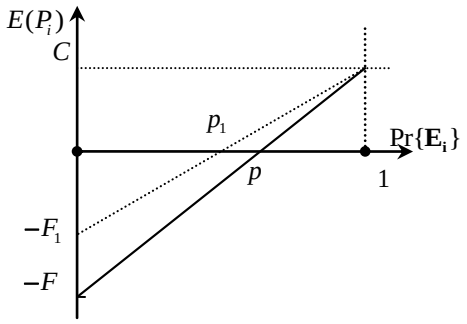


Figure 3. Relationship between $\Pr\{\mathbf{E_i}\}$ and $\mathrm{E}(P_i)$

As Shown in Figure 3, if $\Pr\{\mathbf{E_i}\} > p$, then the expected profits $E(P_i)$ is positive with maximal value C. Otherwise, $E(P_i)$ would be negative. Hence, p is an indicative measure, above which the acceptance of r_i may be profitable. It is noteworthy that $\Pr\{\mathbf{E_i}\} < p$ only means that accepting r_i is very risky. Consider that RM uses $p_*(0 < p_* < 1)$ as its criterion to decide whether or not to accept requests r_i. For example, if $\Pr\{\mathbf{E_i}\} \geq p_*$ RM accepts r_i, otherwise rejects it. A higher value of p_* indicates that RM is conservative; a lower p_* shows that it is willing to take more risks to improve utilization of its resources.

In another scenario, if RM can flexibly adjust its compensation price F, the relationship between $\Pr\{\mathbf{E_i}\}$ and $\mathrm{E}(P_i)$ will be changed correspondingly. In Figure 3, we illustrate this case that RM chooses a lower compensation price F_1. Correspondingly, p changes into p_1, and $p_1 < p$. It means that to get the same $\mathrm{E}(P_i)$, a lower compensation price leads to lower risks. This is consistent with our common sense.

It is noteworthy that our analysis does not assume that service time on different resources follows an identical distribution. Nor do we assume that service time on a resource follows certain distribution. We believe that those distributions should be estimated based on historical information recorded by RM. For example, RM may find that service time on a certain resource follows exponential distribution with a mean value $1/\lambda$, then its cumulative distribution function can be expressed as $F(t) = 1 - e^{-\lambda t}$. Only here, we can get follows expressions

$$\begin{cases} \Pr\{TE_i \leq te_i \mid TS_i = ts_i\} = F(te_i - ts_i) \\ \Pr\{TE_k \leq ts_i \mid TS_k = ts_k\} = F(ts_i - ts_k) \end{cases} \tag{6}$$

Then, Formula (4) can be rewritten as

$$\begin{aligned} \Pr\{\mathbf{E_i}\} &= F(te_i - ts_i) \cdot F(ts_i - ts_k) \\ &= (1 - e^{-\lambda(te_i - ts_i)}) \cdot (1 - e^{-\lambda(ts_i - ts_k)}) \end{aligned} \tag{7}$$

In summary, with $\Pr\{\mathbf{E_i}\}$ and $\mathrm{E}(P_i)$, RM can take diverse policies. The optimal value of C, F, and p_* should be determined by system according to its own objectives, such issues are out of the scope of this paper.

4.2. Adaptive Redundant Reservation Policy

Let $\boldsymbol{J} = \{t_1, t_2, \ldots, t_n\}$ represent the application's subtask set, and each subtask is noted as $<a_i, c_i, v_i>$, where a_i is the execution time, c_i is the resource demands, v_i is the reservation quantity. The system's resource site is noted as set $\boldsymbol{R}=\{r_1, r_2, \ldots, r_m\}$, and each site's time slot set is noted as $Slot(r_i)=\{s_{i,1}, s_{i,2}, \ldots, s_{i,k}\}$. Based on the above definitions, we can note the resource mapping scheme as $\boldsymbol{S}: \boldsymbol{J} \times \boldsymbol{R} \rightarrow \{0, 1\}$, which means that the co-reservation scheme is $n \times m$ matrix.

Let random event $\Psi_{i,j}$ present that subtask t_i has successfully reserved resource on r_j, and its probability is noted as $\Pr\{\Psi_{i,j}\}$. Therefore, the probability that the reservation scheme $\boldsymbol{S}$ can be accepted can be noted as $\Pr(\boldsymbol{\Psi}, \boldsymbol{S})$, where $\boldsymbol{\Psi}$ is the random event matrix. When using adaptive redundant policy, we hope that the redundant degree (noted as K)

can be dynamically adjusted according the runtime workload. So the problem can be described as following:

$$\begin{aligned} &\min \quad K(\boldsymbol{\Psi}, \boldsymbol{S}) \\ &s.t. \quad \boldsymbol{S} = \boldsymbol{J} \times \boldsymbol{R} \to \{0,1\} \\ &\qquad \Pr\{\boldsymbol{\Psi}, \boldsymbol{S}\} = \prod_{i=1}^{n} \Pr\{\bigcup_{j=1}^{m} \Psi_{i,j}\} \end{aligned} \tag{8}$$

To solve the problem (8), we give the following theorems and conclusions.

Theorem 1. The probability that resource site r_j can satisfy the start time of subtask t_i is

$$P^{\alpha}(s_{j,k}, t_i) = \prod_{l \in \Omega^*} \Pr\{T_l \le ts_l\} \tag{9}$$

where T_l is practical completing time of t_l, set Ω^* must satisfy the following condition:

$$\begin{cases} \Omega^* = \{\Psi \mid \Psi \subseteq \alpha_j(t_i), \max\{\prod_{l \in \Psi} \Pr\{T_l \le ts_i\}\}\} \\ v_i \le c_{j,k} + \sum_{l \in \Omega^*} v_l \end{cases} \tag{10}$$

Proof. According the descriptions in Section IV.1, it is clear that if the subtask in $\alpha_j(t_i)$ can be finished before the start time of t_i, the set Ψ must satisfy

$$v_i \le c_{j,k} + \sum_{l \in \psi} v_l \tag{11}$$

Under this condition, we can use probability theory to calculate the total probability as

$$P^{\alpha}(s_{j,k}, t_i) = \prod_{l \in \psi} \Pr\{T_l \le ts_i\} \tag{12}$$

Unfortunately, Ψ is not the only set that can satisfy condition (11). Therefore, we must find an optimal set that can maximize the probability of (12). It is clear that condition (10) is the requirements that such a set should be satisfied, and we note it as Ω^*.

■

Theorem 2. The probability that time slot $s_{j,k}$ can fully satisfy the t_i's reservation requirement is

$$P^{\beta}(s_{j,k}, t_i) = \Pr\{T_i \le ts_l\} \tag{13}$$

where $\Pr\{T_i \le ts_l\}$ is the probability that t_i is finished before $t_l \in \beta_j(t_i)$, and t_l should satisfy the following condition:

$$\sum_{j=1}^{l-1} v_j \le C^{max} - v_i < \sum_{j=1}^{l} v_j \tag{14}$$

Proof. Firstly, we sort the requests in $\beta_j(t_i)$ by ascendant order of their start time. Then, it is clear that if the practical finishing time of t_i is earlier than ts_1 then this subtask will not interfere any request in $\beta_j(t_i)$, and the probability calculating equation is noted as (13). On the other side, we must make sure that the resource site has enough

available resources when t_i's start time is coming. So, the condition in (14) is used for this cause, where C^{max} is the total resources quantity on r_i.

■

Based on the conclusions of Theorem 1 and Theorem 2, we can calculate the probability of successful reservation for the whole application by the following equation:

$$P(\bigcup_{j=1}^{m} \Psi_{i,j}) = 1 - \prod_{j=1}^{k} [1 - P^{\alpha}(s_j, t_i) \cdot P^{\beta}(s_j, t_i)] \tag{15}$$

By (15), we can further draw the following conclusion.

Corollary 1. When using adaptive redundant reservation, if we want to make sure that the probability of successful reservation is higher than W^*, then the redundant degree K should be bounded in:

$$\begin{cases} K_{\min} = \left\lceil \dfrac{\ln(1-W^*)}{\ln(1-\min\{P^{\alpha}(s_j,t_i)P^{\beta}(s_j,t_i)\})} \right\rceil \\ K_{\max} = \left\lceil \dfrac{\ln(1-W^*)}{\ln(1-\max\{P^{\alpha}(s_j,t_i)P^{\beta}(s_j,t_i)\})} \right\rceil \end{cases} \tag{16}$$

The above corollary can be applied in practical systems for calculating the runtime redundant degree.

5. Experimental Results and Analysis

5.1. Experiment Settings

In this section, massive simulations are conducted to verify the efficiency of our *Adaptive Redundant Reservation with Overlapping Strategy* (ARROS). We focus on the effectiveness of ARROS comparing with traditional reservation mechanism, and what price we should pay when applying ARROS. In simulations, we choose Lublin-Feitelson Model [13], which is derived from existing workload logs, to generate experimental workload (reservation requests). Each request in the workload is characterized by its arrival time, number of nodes, and running time. As the model is based on long-term jobs on supercomputer, we divided the arrival times and running times by 60 to reduce the overall time to run the experiments. To reflect the overestimation of relative reservation deadline, we multiply running time of each request with a random factor k_{over} ($k_{over} \geq 1$). The resource model consists of 16 resource site and a RM. The RM is designed to capable of enforcing either traditional or overlapped reservation strategies.

5.2. Comparison on Resource Utilization and Rejection Rate

In the first experiment, we investigate the resource utilization and rejection rate when using ARROS. The basic workload used consists of 8000 requests, and it is modified into four different workloads with 5%, 10%, 15%, 20% requests using reservation, respectively. In this experiment, we set $p_* = 0.8$, which means that RM will overlap a reservation request only when the probability of reservation violation for this request is

less than 20%; and the factor k_{over} is set to be uniformly distributed in the interval [1.2, 1.5], which means reservation requests tend to overestimate their relative deadline with mean value 35%. The experimental results are shown in Figure 4 and Figure 5.

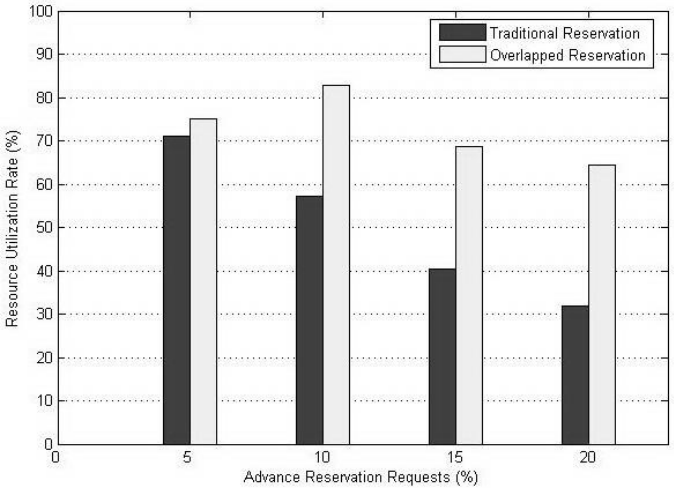


Figure 4. Comparison of resource utilization rate

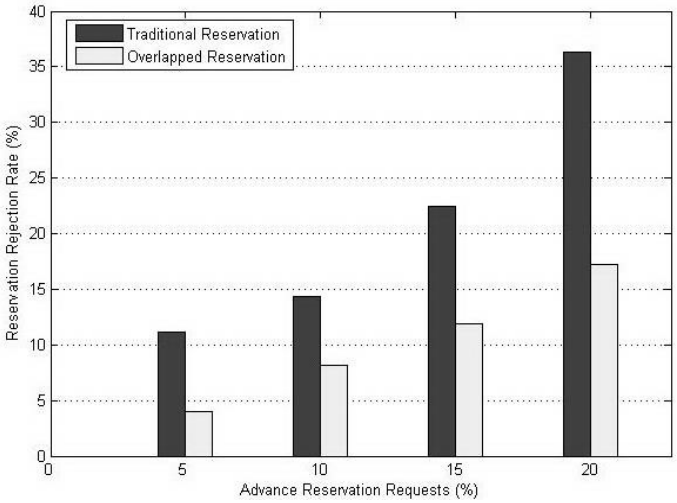


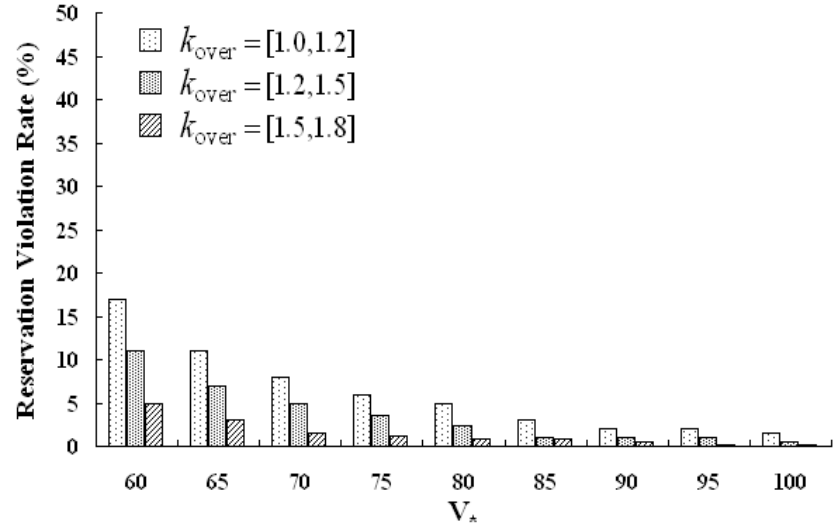
Figure 5. Comparison of reservation rejection rate

As we can see from Figure 4, for traditional reservation, as the percentage of reservation requests increases from 5% to 20%, resource utilization drops from 71% to about 30%. While applying ARROS, resource utilization keeps relatively steady and do not drop dramatically like traditional reservation. On the contrary, we notice that when the reservation percentage increases from 5% to 10%, resource utilization increases about 7%. The reason is that more free time slots can be allocated as reservation requests increases. However, such increasing can not be sustained when the percentage of reservation requests increases to 15% and more. Reservation rejection rates are depicted in Figure 5. Like the resources utilization, when using traditional reservation, the rejection rate increases sharply with the increasing of reservation requests. By applying ARROS, the rejection rate is only about 50% of traditional reservation.

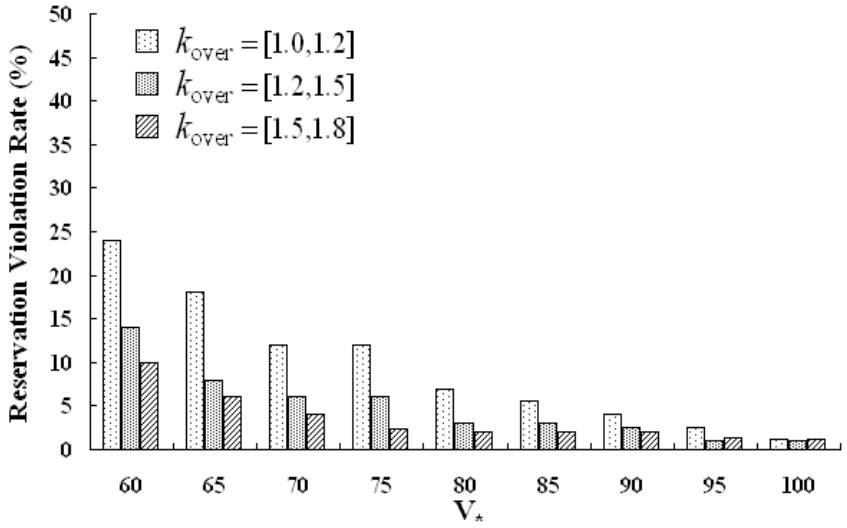
Combing the experimental results of resource utilization and rejection rate, we conclude that, in addition to improve resource utilization and lower down rejection rate, overlapped reservation strategy is more robust in presence of higher percentage of reservation requests.

5.3. Comparison on Reservation Violation Rate

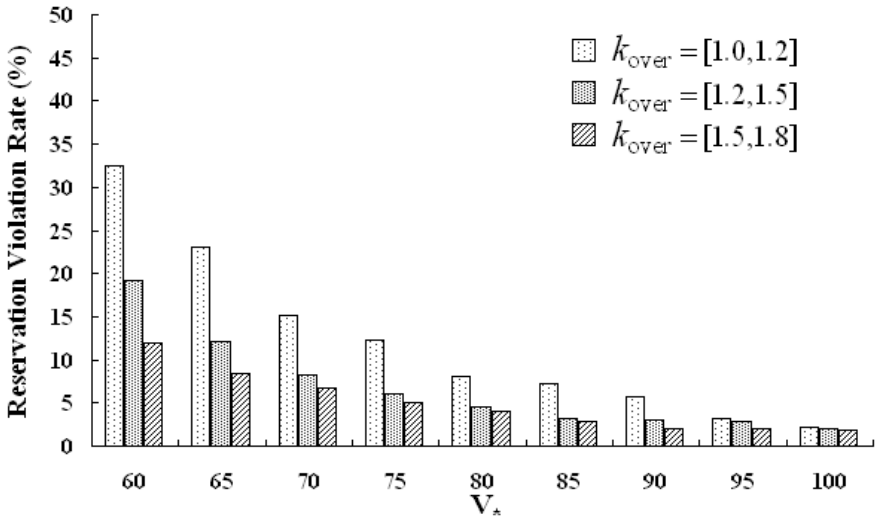
In this section, we focus on reservation violation, which is caused by using ARROS. As our experiments are conducted on simulator, violations cased by network disconnection, system crash and etc. are all ignored. So, we assume that the violation rate is zero when using conventional reservation policy, and only investigate the violation when ARROS is used. In this experiment, the effects of v_* and k_{over} on the performance of ARROS is extensively investigated. In Figure 6 reservation violations for four reservation rates are shown respectively.



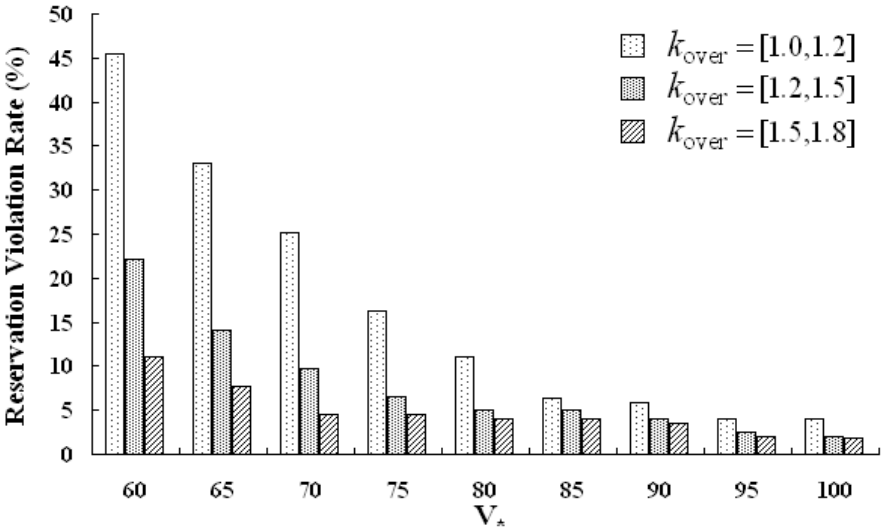
(a) Reservation Rate = 5%



(b) Reservation Rate = 10%



(c) Reservation Rate = 15%



(d) Reservation Rate = 20%

Figure 6. Comparison of Reservation Violation Rate with different v_* and k_{over}

As mentioned before, we multiply the deadline of each reservation with a factor k_{over} ($k_{over} \geq 1$) to reflect the overestimation of deadline. In this experiment, we have conducted simulations with different level of k_{over}. More specifically, we set k_{over} uniformly distributed in the different intervals, such as [1.0, 1.2], [1.2, 1.5], and [1.5, 1.8] respectively. We denote them as k_1, k_2, k_3 for the sake of simplifying representation. When k_{over} is set in level k_1, it means requests in workload tend to overestimate their deadlines with mean value 10%. So, k_2 means 35% overestimation, and k_3 means 65% overestimation. As shown in Figure 6, it is clearly that more overestimation of deadline leads to lower reservation violation rate. That is because many overlapped reservations do not overlap actually in run time, which makes ARROS more effective.

Parameter v_* is the threshold, by which RM decides whether a free time slot can be allocated to an overlapped reservation or not. So, v_* is a strategic parameter of RM, a higher value of v_* indicates that RM is conservative. In this experiment, we increase v_* from 0.6 to 0.95 gradually. The results show that for k_1 and k_2 the violation rate drops quickly when v_* is increased from 0.6 to 0.8, then the decline becomes stable. In all cases tested, we found that, the reservation violation rate can be limited below 10% when $v_* \geq 0.8$. If we set $v_* \geq 0.9$, the violation can be controlled below 5%. In Figure 6(c) and Figure 6(d), the experimental results indicate that low value of v_* is not a good idea when system is in presence of high reservation rate (>15%).

The increasing of reservation violation is the price we have to pay while using ARROS, experimental results indicate that we can limit the violation rate in a relative low level by adjusting parameter v_*. Currently, many systems have support reservation re-negotiation mechanism, such as Maui, EASY, COSY. Combining ARROS and reservation re-negotiation, the RM can take the advantages of ARROS as well as avoiding the risk of reservation violation. It is noteworthy that different resource providers may prefer different v_* in practical systems, and RM may dynamically set v_* for different type of resources based on its resource reservation policies. In our simulations, we set an identical v_* for all resources only for simplicity.

6. Conclusion

In this paper, we studied the negative effects brought about by advance reservation in distributed computing. To mitigate those effects, we proposed a novel reservation strategy based on the fact that applications tend to overestimate their running time to ensure their completion. Extensive simulations based on real workload were conducted to verify the effectiveness of our overlapped reservation strategy. Experimental results show that the strategy can bring about remarkably higher resource utilization and lower rejection rate at the price of a slightly increasing of reservation violations. Furthermore, the overlapped strategy shows robustness in presence of higher percentage of reservation requests. For the future work, we plan to provide an adaptive mechanism for RM to dynamically set parameter p_*, which have significant influences on the performance of ARROS. Furthermore, as parameter v_* have significant influences on the performance of ARROS, we plan to provide an adaptive mechanism for RM to dynamically set optimal v_* based on resource's runtime load.

Acknowledgements

This work is supported by the Provincial Science & Technology plan project of Hunan (No.2012GK3075). Also, it is a project supported by Hunan Provincial Natural Science Foundation of China (No. 13JJ9022).

References

[1] K. Kurowski, A. Oleksiak and J. Weglarz, "Multicriteria, multi-user scheduling in grids with advance reservation", Journal of Scheduling, vol. 13, no. 5, **(2010)**, pp. 493-508.
[2] R. Prodan and M. Wieczorek, "Negotiation-Based Scheduling of Scientific Grid Workflows Through Advance Reservations", Journal of Grid Computing, vol. 8, no. 4, **(2010)**, pp. 493-510.

[3] A. Roy and V. Sander, "Advance Reservation API", GFD-E.5, Scheduling Working Group, Global Grid Forum (GGF), **(2002)** May.
[4] K. Vanmechelen, W. Depoorter and J. Broeckhove, "Market-based grid resource co-allocation and reservation for applications with hard deadlines", Concurrency and Computation-Practice & Experience, vol. 21, no. 18, **(2009)**, pp. 2270-2297.
[5] W. Smith, I. Foster and V. Taylor, "Scheduling with Advanced Reservations", In Proc. of the 14th IEEE International Symposium on Parallel and Distributed Processing (IPDPS'00), **(2000)**, pp. 127-132.
[6] R. Prodan and M. Wieczorek, "Negotiation-Based Scheduling of Scientific Grid Workflows Through Advance Reservations", Journal of Grid Computing, vol. 8, no. 4, **(2010)**, pp. 493-510.
[7] S. Naiksatam and S. Figueira, "Elastic reservations for efficient bandwidth utilization in LambdaGrids", Future Generation Computer Systems, vol. 23, no. 1, **(2007)**, pp. 1-22.
[8] A. C. Sodan, C. Doshi, L. Barsanti and D. Taylor, "Gang Scheduling and Adaptive Resource Allocation to Mitigate Advance Reservation Impact", Proc. of the 6th International Symposium on Cluster Computing and the Grid, **(2006)**.
[9] E. Elmroth and J. Tordsson, "Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions", Future Generation Computer Systems, vol. 24, no. 6, **(2008)**, pp. 585-593.
[10] E. Elmroth and J. Tordsson, "A standards-based Grid resource brokering service supporting advance reservations, coallocation, and cross-Grid interoperability", Concurrency and Computation-Practice & Experience, vol. 21, no. 18, **(2009)**, pp. 2298-2335.
[11] A. Sodan and X. Huang, "Adaptive Time/Space Scheduling with SCOJO", In Proc. of International Symposium on High Performance Computing Systems, **(2004)**, pp. 165-176.
[12] A. Sulistio, W. Schiffmann and R. Buyya, "Advanced Reservation-based Scheduling of Task Graphs on Clusters", In Proc. of the 13th Annual IEEE International Conference on High Performance Computing (HIPC'06) , **(2006)**, pp. 18-21.
[13] U. Lublin and D. G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs", Journal of Parallel and Distributed Computing, vol. 63, no. 11, **(2003)**, pp. 1105-1122.
[14] J. Luo, Z. Wu, J. Cao, *et al.*, "Dynamic multi-resource advance reservation in grid environment", Journal of Supercomputing, vol. 60, no. 3, **(2012)**, pp. 420-436.
[15] A. D Stefano, M. Fargetta, G. Pappalardo, *et al.*, "Supporting resource reservation and allocation for unaware applications in Grid systems", Concurrency and Computation-Practice & Experience, vol. 18, no. 8, **(2006)**, pp. 851-863, 2006.
[16] A. W. Mu'alem and D. G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling", IEEE Trans. on Parallel and Distributed Systems, vol. 12, no. 6, **(2001)**, pp. 529-543.
[17] R. Yang, S. Bhulai, R. Mei, *et al.*, "Optimal resource allocation for time-reservation systems", Performance Evaluation, vol. 68, no. 5, **(2011)**, pp. 414-428.
[18] N. R. Kaushik, S. M. Figueira and S. A. Chiappari, "Flexible Time-Windows for Advance Reservation Scheduling", In Proc. of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'06), **(2006)**, pp. 218-225.
[19] M. Sojka, P. Píša, D. Faggioli, *et al.*, "Modular software architecture for flexible reservation mechanisms on heterogeneous resources", Journal of Systems Architecture, vol. 57, no. 4, **(2011)**, pp. 366-382.
[20] S. Stein, T. R. Payne and N. R. Jennings, "Robust Execution of Service Workflows Using Redundancy and Advance Reservations", Ieee Transactions on Services Computing, vol. 4, no. 2, **(2011)**, pp. 125-139.
[21] L. D. Moscholios and M. D. Logothetis, "The Erlang multirate loss model with Batched Poisson arrival processes under the bandwidth reservation policy", Computer Communications, vol. 33, **(2010)**, pp. S167-S179.
[22] A. Kovacs, I. Godor, S. Racz, *et al.*, "Cross-layer quality-based resource reservation for scalable multimedia", Computer Communications, vol. 33, no. 3, **(2010)**, pp. 283-292.
[23] C. L. Yu, C. S. Chang and D. -S. Lee, "CR Switch: A Load-Balanced Switch With Contention and Reservation", Ieee-Acm Transactions on Networking, vol. 17, no. 5, **(2009)**, pp. 1659-1671.
[24] C. Castillo, G. N. Rouskas and K. Harfoush, "Online algorithms for advance resource reservations", Journal of Parallel and Distributed Computing, vol. 71, no. 7, **(2011)**, pp. 963-973.
[25] J. S. Lin and K. T. Feng, "QoS-Based Adaptive Contention/Reservation Medium Access Control Protocols for Wireless Local Area Networks", IEEE Transactions on Mobile Computing, vol. 10, no. 12, **(2011)**, pp. 1785-1803.

Authors

Peng Xiao received the Ph.D degree in computer science from the Central South University in 2009. Currently, he is an associate professor in the Hunan Institute of Engineering. Also, he is the advanced network engineer in HP High-performance Network Centre in Hunan. His research interests include, distributed resource management. He is a member of ACM and IEEE.

PeiXin Qu received his Master degree in Henan University. Currently, he works as lecturer in Henan Institute of Science and Technology. His research interests include complex networking deployment, distributed computing, information security technology, fault-tolerance in distributed systems.

Xilong Qu received his master degree in University of Electronic Science and Technology of China, and doctor degree in Southwest Jiaotong University. Currently, he is an associate professor in Hunan Institute of Engineering. His research interests include web service, distributed computing, information security technology.