RESOURCE RESERVATION SERVICE ON PORTABLE BATCH SYSTEM

Tran Ngoc Minh[(1),] Nguyen Tuan Anh[(1),] Do Quoc Tuan[(2),] Bui Dung Anh Tuan[(3)]
(1) University of Technology, VNU-HCM
(2) Renasas Company
(3) TMA Company

1.INTRODUCTION

Low-cost PC-based clusters with high-speed communication network have become the mainstream of parallel and distributed platforms. Parallel applications running on these clusters often require high quality of services. Resource reservation is a common service ensuring the availability of resources when applications are submitted to the system.

Users could reserve resources necessary for the execution of their applications in advance through the reservation service. The reservation can be on processors, network bandwidth, memory, etc. Users must specify their resource requirements such as number of processors, size of memory, or computing power of processors, etc to the reservation service. The service then will check whether free or non-reserved resources could satisfy user requirement. Our current system could support users to reserve compute nodes, each contains one or more processors, in advance. Users submit their jobs with reservation identifiers which confirms their successful reservations to the system. The system then verifies the reservation and schedules jobs on the reserved resources.

Reservation is particularly important for parallel applications which run across several clusters. Unfortunately, current open source systems such as Load Leveler (LL) [7], Portable Batch System (PBS) [1][2][3], etc do not support users to express these kinds of demands. In this paper we present our advanced reservation service for end-users together with the way we built and integrated it into OpenPBS. The rest of this paper is divided into four sections. Section 2 describes related work. Our advanced reservation service is presented in section 3. Then the implementation is shown in section 4. Finally, we conclude the paper with some discussions of our future work in section 5.

2.RELATED WORK

We will discuss in this section some well-known systems together with their support for the advanced reservation.

2.1.Portable Batch System

Portable Batch System (PBS) [1][2][3] is a workload management system developed by Veridian Systems. PBS operates in UNIX environments and a network of heterogeneous workstations. The purpose of PBS is to provide additional controls over initiating or scheduling execution of batch jobs [10]. The advanced reservation service is currently supported in Professional PBS [8] but not in OpenPBS. However, the advanced reservation service in Professional PBS does not support users to modify booked reservations [8] as in our system.

2.2.Maui Scheduler

Maui [9][4] is an advanced job scheduler for use on clusters and supercomputers. It is implemented as a plugin scheduler to PBS. The primary purpose of Maui scheduler is to provide an advanced reservation infrastructure allowing sites to control exactly when, how, and by whom resources are used while still assuring fairness and fairshare policies. It is, however, difficult to implement scheduling algorithms with Maui while this problem is done easily with our system.

2.3.Catalina

Catalina [8][5] is freely available for educational, research and non-profit purposes. It is very similar to Maui in design, but written in Python. Catalina lacks Maui's fairshare, workload profiling, multiple jobs/node. Catalina has short pool, more flexible jobs filtering by reservations. It is designed to support parts of the GridForum Advance Reservation API. Like Maui, changing the scheduling algorithm is also a problem of Catalina. In addition, Catalina does not support users to alter booked reservations [8] as our system.

2.4.Load Sharing Facility

Load Sharing Facility (LSF) [6] is a software for managing and accelerating batch workload processing for compute-and data-intensive applications. With LSF, users can intelligently schedule and guarantee completion of batch workload across a distributed, virtualized IT environment regardless of operating system, including desktops, servers and mainframes. Like Catalina and Professional PBS, the advanced reservation service in LSF also does not support users to alter booked reservations [8].

3.INTEGRATING RESERVATION TO PORTABLE BATCH SYSTEM

3.1.Job Supporting

Our system could support two kinds of jobs: jobs with and without advanced reservation. While the executions of reserved jobs are assured by the system, executions of non-reserved jobs depend completely on the cluster load.

3.2.System Architecture

Our current reservation system could support users to reserve compute nodes in advance. The reservation architecture is shown in Figure. 1. The process of creating a reservation is as follows:

1. The user sends a reservation request consisting of numbers of compute nodes that his job needs for its execution to the reservation module.

2. The reservation module requests information about all compute nodes from the server.

3. The server queries all compute nodes in the system for their current status which could be marked as free, reserved or job-exclusive (the node is serving jobs). This step is necessary because executions of non-reserved jobs could change the status of compute nodes.

4. The server returns node information to the reservation module.

5. The reservation module makes decision based on this information. If number of free nodes is not adequate, the user will receive an error message and his reservation request is rejected. Otherwise, the reservation identifier is returned back to the user.

The process of submitting a job is illustrated in Figure. 2. The user submits his job with the reservation identifier to the server. Upon receiving a job submission, the server creates a job identifier and pass the control to the reservation module. The reservation module verifies the job submission and return the job identifier to the user.

3.3.Properties of the Advanced Reservation Module

Our reservation service assures the availability of necessary resources for job executions. The service provides users four functions: Create, Modify, Show status, and Cancel the reservation.

1. Users use Create function to request a reservation and receive the reservation identifier as a ticket for using resources in case of success.

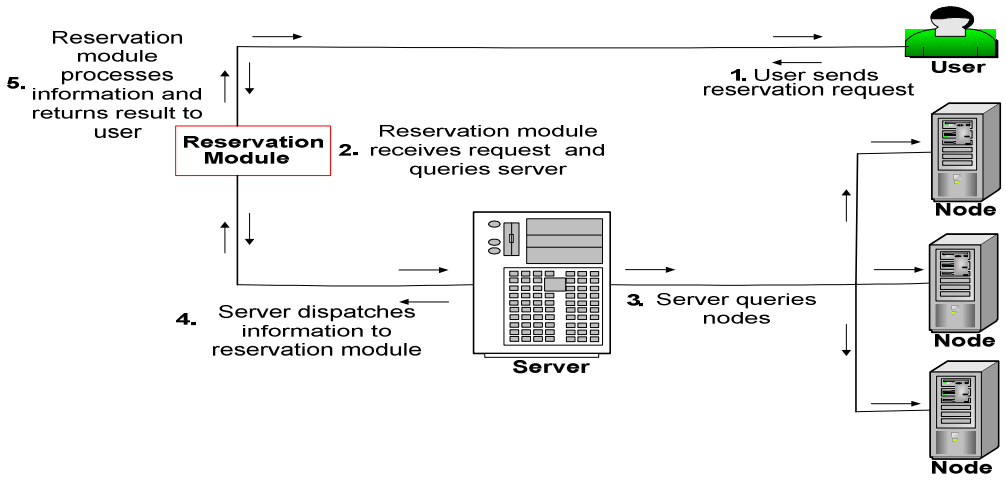

Figure 1. System architecture

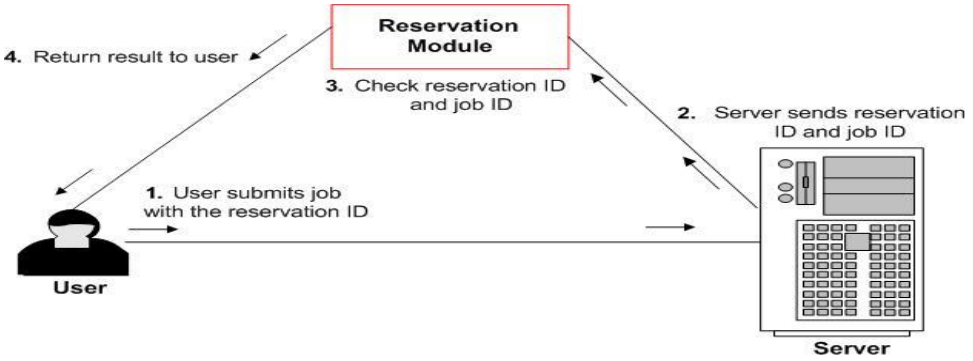

Figure 2. Users submit job with the reservation identifier

2. Users could easily use Modify function with their successful reservation identifier to modify the reservation. This function is ineffective after users submitted a job with that reservation.

3. Besides, our system also supports users to query the current status of their own reservation.

4. The Cancel function is used if users desire to delete their reservation. When this function is called, the status of all reserved compute nodes is set to free. Jobs running on these nodes, if any, will be terminated immediately.

4.IMPLEMENTATION

In order to integrate the reservation module to PBS, we had solved two problems: accepting reservation requests from users and allowing job submissions through PBS with legal reservation identifiers to be executed. In addition, our implementation also allows people to change the scheduling algorithm easily as PBS.

4.1.Accepting Reservation Requests from Users

To accept a reservation request from a user we built a reservation module and defined a new user command named qreserve.

The reservation module will receive from the user number of compute nodes he desires for his job. We refer this one rsv_node_request parameter. The reservation module then asks the server for information about all nodes in the system. If the status of a node is free, that node is

enable to be reserved for user job. In case the status of a node is reserved or job-exclusive, that node could not be reserved. The reservation module makes decision based on this information. If the total number of free nodes in the system is smaller than rsv- _node_request, user request is not legal and rejected. On the contrary, corresponding nodes are updated to be reserved and users will receive a reservation identifier saved in the Reservation structure defined as

struct Reservation

```
{
char* rsv id; // reservation identifier
int rsv time; // time point that a reservation request comes
int rsv node; // number of nodes required
};
```

If the reservation is timed out, corresponding reserved nodes will be freed and become available for other jobs. The reservation module controls this timeout by a system-wide parameter PBS_MAX_WAIT- _TIME.

At the current prototype, we use the command qreserve to support the user in sending a reservation request to the reservation module together with number of compute nodes he desires for his job.

4.2.Allowing Job Submissions

To allow the submission of the reservation identifier as an input parameter with the job, PBS defines input arguments in two C-header files: src/include/site_job_attr_def.ht and src/include/site_job_attr- _enum.ht. We will add two variables named reservation_id and rsv_valid to these files to support jobs with reservation. The former is the reservation identifier and the latter shows the kind of a job: normal or reserved job. Jobs without reservation are pushed into queues. Server will mark status of these jobs as Hold_Job and postpone their executions until enough resources become available and all reserved jobs get their resources. When a job is terminated, resources will be released and marked as free.

4.3. Changing the Scheduling Algorithm

Figure 3. illustrates the way PBS implementing its scheduler. We could flexibly change the default FIFO scheduling algorithm of PBS by modifying the function scheduling_cycle() which is called directly by the function schedule(). In our system, we still remain this property to allow people to change the scheduling algorithm easily. As shown in Figure 4., the function schedule() will call resv_scheduling_cycle() directly instead of scheduling_cycle() and then resv_scheduling_cycle() will call scheduling_cycle(). The function resv_scheduling_cycle() assures that reserved jobs are executed prior to normal jobs. This was done thank to rsv_valid property of each job. If rsv_valid is true, the job will be executed immediately. If not, it will be requeued and wait until available resources are adequate and all reserved jobs in the queue get their resources. Scheduling algorithms for non-reserved jobs could be implemented easily in scheduling_cycle() as PBS without modifying anything related to the reservation service.

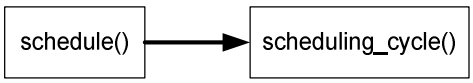

Figure 3. Scheduling implementation in PBS

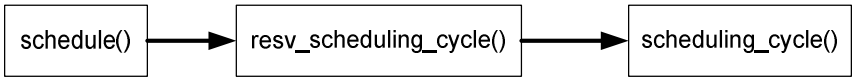

Figure 4. Modify the scheduler in our system

5.CONCLUSIONS AND FURTHER STUDY

Advanced reservation is a very convenient service for users in assuring the successful execution of their jobs. The service provides users a way to assure the availability of all resources necessary for job executions. We have developed and presented in this paper an advanced reservation service which has been integrated into OpenPBS.

Our reservation service can be used by not only an administrator, but also a normal user. In addition, the system also allows users to query, modify or cancel reserved jobs in a user's friendly way. As comparing with other systems which support the advanced reservation without altering booked reservations [8] such as Professional PBS, LSF, and Catalina our system could allow users to modify their successful reservations and allow people to change the scheduling algorithm easily.

At the current prototype, we only support reservations on compute nodes. More types of resources such as network bandwidth, memory, etc will be taken into account for the future development. We also plan to enhance the reservation system by new features such as negotiation, supporting interactive jobs, renegotiated reservation, etc. Finally, we desire the system is used to manage gridnodes in EDAGrid which is being developed by HCMUT and will finish in 2008.

REFERENCES

[1]. Bayucan A., Lesiak C., Mann B., Henderson R. L., Proett T., and Tweten D., *External Reference Specification*, Release 1.1.12, http://www-unix.mcs.anl.gov/openpbs/, August (1998).

[2]. Bayucan A., Lesiak C., Mann B., Henderson R. L., Proett T., and Tweten D., *Internal Design Specification*, Release 1.1.12, http://www-unix.mcs.anl.gov/openpbs/, August (1998).

[3]. Bayucan A., Lesiak C., Mann B., Henderson R. L., Proett T., Tweten D., and Jasinskyj L. T., *Portable Batch System Administrator Guide,* Release 1.1.12, http://www.compsci.wm.edu/SciClone/documentation/software/OpenPBS/, August (1998).

[4]. Bode B., Halstead D. M., Kendall R., and Lei Z., *The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters*, Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, October (2000).

[5]. Catalina, http://www.sdsc.edu/catalina.

[6]. Load Sharing Facility, http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/.

[7]. LoadLeveler, http://www.mhpcc.edu/training/workshop/loadleveler/MAIN.html.

[8]. MacLaren J., Ziegler W., and Fraunhofer, *Advance Reservations: State of the Art*, June (2003).

[9]. Maui, http://www.clusterresources.com/pages/resources/documentation.php.

[10]. Sherwani J., Ali N., Lotia N., Hayat Z., and Buyya R., *Libra: A Computational Economy-Based Job Scheduling System for Clusters*, Software: Practice and Experience, Volume 34, Issue 6, p573-590, May (2004).