Enabling Personal Clusters on Demand for Batch Resources Using Commodity Software

Yang-Suk Kee, Carl Kesselman
Information Sciences Institute, University of Southern California
{yskee,carl}@isi.edu
Daniel Nurmi, Rich Wolski
University of California, Santa Barbara
{dan,rich}@cs.ucsb.edu

Abstract

Providing QoS (Quality of Service) in batch resources against the uncertainty of resource availability due to the space-sharing nature of scheduling policies is a critical capability required for high-performance computing. This paper introduces a technique called personal cluster which reserves a partition of batch resources on user's demand in a best-effort manner. A personal cluster provides a private cluster dedicated to the user during a user-specified time period by installing a user-level resource manager on the resource partition. This technique not only enables cost-effective resource utilization and efficient task management but also provides the user a uniform interface to heterogeneous resources regardless of local resource management software. A prototype implementation using a PBS batch resource manager and Globus Toolkits based on Web Services shows that the overhead of instantiating a personal cluster of medium size is small, which is just about 1 minute for a personal cluster having 32 processors.

1. Introduction

Best-effort batch queuing is the most popular resource management paradigm used for high performance and Grid computing. Most clusters in production today employ a variety of batch scheduling software such as PBS (Portable Batch System) [1], Condor [2], LSF (Load Sharing Facility) [3], and so on for efficient resource management and a space-sharing resource management policy for QoS (Quality of Service). The major goal of this resource management paradigm is to achieve high throughput across a cluster system and maximize the system utilization.

By contrast, the common interest of the users who use these systems is to achieve the best performance of their applications in a cost-effective way. However, these systems are unlikely to optimize the turnaround time of a single application especially consisting of multiple tasks against the fair sharing of resources between jobs. For instance, other applications' jobs can intervene between the jobs of the application being submitted to batch queues, which makes the final execution time of the application unpredictable. We understand that batch systems with user-level advance reservation [4] can be a promising alternative to secure performance and being actively studied in the high-performance computing community. However, user-level advance reservation is still neither popular nor cheap in general because it adversely affects the fairness and the efficient resource utilization.

Therefore, we need to lessen conflicts of interest between end users and resource providers. Especially, this paper discusses about QoS and cost-effectiveness in resource allocation. We introduce a technique called personal cluster (PC) which enables private dedicated clusters on demand for best-effort batch resources. We call a partition of batch resources allocated for an application and managed by its own resource manager as personal cluster. Personal cluster makes the resources available for the application's lifetime by automatically installing a placeholder instead of actual jobs. Residing on the allocated resources, the placeholder provides a uniform interface to resources and task management services regardless of local resource management software. Specifically, three major benefits from personal clusters include:

- **Cost-effective resource allocation**: since a personal cluster acquires batch resources via default best-effort schedulers of resource providers and releases them immediately to the host cluster at termination, it requires neither any modifications of local schedulers nor extra cost for reservation.
- **Efficient task/resource management**: a personal cluster provides an exclusive access to a partition of resources and enables efficient task/resource management via commodity software.
- **Uniform interfaces to heterogeneous resources**: a personal cluster can provide a uniform job/resource management environment over heterogeneous resources (e.g., PBS interfaces in our prototype system) regardless of system-level resource management paradigms.

This paper is organized as follows. Firstly, we discuss the high-level idea of personal cluster and its implementation using a PBS installation and Globus Toolkits in Section 2. Then, we analyze the overhead of installing personal clusters on a cluster system in Section 3. We discuss other related studies in Section 4 and finally conclude in Section 5.

2. Personal Cluster

A personal cluster is a private cluster instantiated on demand from batch resources, which gives an illusion to the user as if the instant cluster is dedicated to the user for a certain time period. Personal cluster enables the uniform, cost-effective resource use for batch resources in a simple manner.

Firstly, personal clusters enable efficient job and resource management for best-effort batch resources. Under best-effort resource environment, the tasks submitted for an application have to compete for resources with other applications. For instance, a common use pattern of the users who conduct parameter studies with best-effort batch resources is to submit a series of jobs into the queues, anticipating short turnaround time. However, the jobs can be interrupted by other applications' tasks because any job can be presented to the system while the series of jobs are being submitted. If an application is interrupted by a long-running job, the overall turnaround time of the application can be delayed significantly. In order to prevent the performance degradation due to such interruptions, the user can cluster the tasks together and submit a single script that runs the actual tasks when the script is executed. However, this clustering technique cannot be benefited by the common capabilities for efficient scheduling such as backfilling provided by resource management systems.

By contrast, a personal cluster holds resources for the application lifetime on the behalf of the application and provides an execution environment similar to the host cluster. In other words, a resource manager is installed as a placeholder on the allocated resources and plays as a gateway, taking care of resource-specific access mechanisms as well as task launching and scheduling. In consequence, the users can have a dedicated cluster under the control of a private resource manager. However, it is redundant and unnecessary to implement a new job/resource manager for this purpose. As an alternative, we utilize commodity tools for job and resource management. The commodity tools provide a vehicle for efficient resource management and make the application development simple.

In the sense that a resource partition is dedicated for the application, a promising solution is to have batch scheduler support user-level advance reservation. However, user-level advance reservation is not common yet and it can be cost-ineffective because the users have to pay for the entire reservation period regardless of whether they use the resources or not. Furthermore, the resource provider may charge more on the users for reservation since reservation can be adverse to efficient resource utilization of the entire system and the fairness between jobs. By contrast, personal clusters can have the same benefits without the host cluster having any special scheduler as the batch systems with advance reservation. Personal clusters do not cause any surcharge for reservation since the resources are allocated in a best-effort manner. Moreover, they can terminate at any time without any penalty because the allocated resources will be returned to the host cluster immediately at termination.

To execute a job on batch resources, the users have to write a job submission script. If the users want to run their applications on heterogeneous resources such as TeraGrid [5], they have to write multiple job descriptions for each resource management software. Personal clusters lessen this kind of burdens from the users by providing a uniform runtime environment regardless of local resource management software. That is, the commodity batch scheduler installed for the allocated resources makes the execution environment homogeneous and consistent bypassing the system-level resource managers. For instance, we use a PBS job manager as a placeholder in our prototype system. As such, the users only need to write a PBS script regardless of the kinds of the system-level job managers (e.g., LSF, Condor, PBS, SGE).

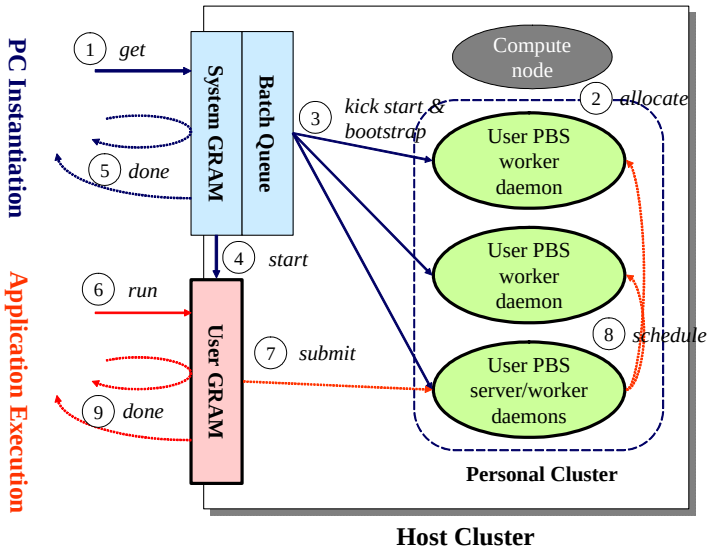

Figure 1. Instantiating a personal cluster using a user-level PBS installation and a WS-GRAM container

2.1 Implementation

We have implemented personal clusters for best-effort batch systems, using WS-based Globus Toolkits [6] and a PBS installation. We assume a conservative cluster configuration where a remote user can access the cluster via public gateway machines while the individual nodes behind the PBS system are private and the accesses to the allocated resources are allowed only during the time period of resource allocation. Then, the batch scheduler allocates a resource partition and launches placeholders on the resources via remote launching tools such as rsh, ssh, pbsdsh, mpiexec, etc, depending on the administrative preference.

A client component called PC factory instantiates personal clusters on the behalf of users, submitting requests to the system-level batch schedulers, monitoring the progress status, and installing software components. In essence, the actual job the factory submits sets up a private, temporary version of PBS on a per application basis. This user-level PBS installation has access to the resources and accepts actual jobs from the user. As foundation software, we use the most recent open source Torque package [7] and made several source level modifications to enable a user-level execution. We can use any resource managers that can run at the user-level.

Figure 1 illustrates how to configure a personal cluster using user-level PBS and WS-GRAM service when the resources are under the control of a batch system and Globus Toolkits based on Web Services provide the access mechanisms. We preinstalled a user-level GRAM server and a user-level PBS on the cluster and configured the user-level GRAM-PBS adaptor to communicate with the user-level PBS. The coordinator first launches a kick-start script to identify allocated hosts and then invokes a bootstrap script for configuring PBS daemons on each node. The kick-start script assigns an ID for each node, not each processor, and identifies the number of processors allocated for each node. For batch resources, the system batch scheduler will launch this kick-start script on the resources via a system-level GRAM adaptor (e.g., GRAM-PBS, GRAM-LSF). Once the kick-start script has started successfully, the system resource manager retreats and the factory has control of the allocated resources. At last, the bootstrap script configures a user-level PBS for the resources on a per-node basis. The node with the ID 0 hosts a PBS server (i.e., pbs_server) and a PBS scheduler (i.e., pbs_sched) while the others do the PBS workers (i.e., pbs_mom). The bootstrap script creates default directories for log, configuration files, and so on; generates a file for the communication with the personal GRAM-PBS adaptor (i.e., globus-pbs.conf), configures queue management options; and starts the daemon executables, based on its role. Finally, the factory starts a personal WS-GRAM server via the system-level GRAM-FORK adaptor on a gateway node of the resources.

Once a user-level PBS and GRAM are in service, the user can bypass the system-level resource management and utilize the resources as if a dedicated cluster is available. Now a personal cluster is ready and the user can submit multiple jobs via this private,

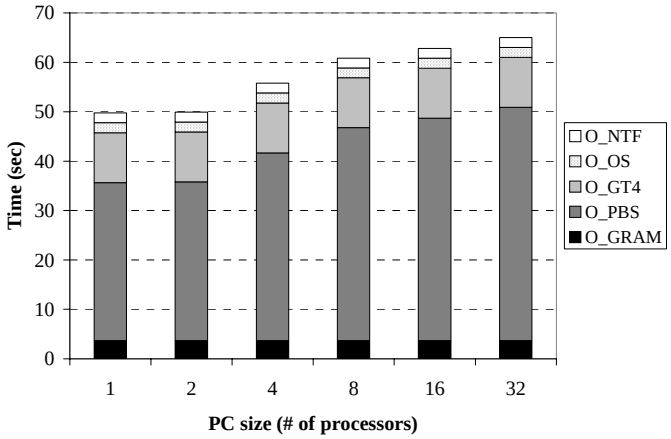

Figure 2. The overhead of installing personal clusters based on a PBS installation and a WS-GRAM container on a cluster consisting of 16 nodes with dual Xeon processors connected by a Gigabit Ethernet

temporary WS-GRAM service using the standard WS-GRAM schema or directly submit them to the private PBS, leveraging a variety of PBS features for managing the allocation of jobs to resources.

3. Evaluation

We estimate the overhead of installing personal clusters by measuring the set up time of user-level PBS and WS-GRAM service on a host cluster that consists of 16 compute nodes with 3.2 GHz dual Xeon processors. Each node is connected to a 1.0 Gigabit Ethernet network and the users' home directories are mounted via NFS. Globus Toolkits 4.0.1 and an OpenPBS 1.2.0 are installed for system-level resource management while Globus Toolkits 4.0.3 and a Torque 2.1.2 package including a PBS installation are used for personal clusters.

The overhead of personal clusters based on Globus and PBS consists of 5 major factors: O_GRAM, O_PBS, O_GT4, O_OS, and O_NTF. O_GRAM represents the processing time for GRAM service that the GT4 container spends until it submits a PBS job request for configuring personal clusters to the system PBS, which include parsing WS-GRAM schema, delegating credentials, generating scripts for PBS job submission, invoking the GRAM-PBS scheduling adaptor, and so on. O_PBS denotes the time to make a user-level PBS system ready on the allocated resources, which includes launching the kick-start script, starting pbs_server and pbs_moms on compute nodes, and synchronizing resource status across the PBS system. O_GT4 is the time spent to launch a user-level Globus container. O_OS is the operating system overhead mostly for file state synchronization via NFS. Finally, O_NTF represents the event notification overhead of the system Globus Container.

The experimental results are shown in Figure 2. Overall, the overhead of small scale personal clusters increases in proportion to the number of processors approximately in a log scale. The overhead of personal clusters with 2 processors is almost same to that of personal clusters with single processor because a PBS worker daemon is spawned per node, not per processor. The only additional delay that personal clusters with 2 processors experience is due to launching the kick-start script on the other processor on the same node through ssh.

The overhead of personal clusters with single processor is about 50 seconds and the overhead increases with the PC size. The most dominant factor is O_PBS, which explains more than 64% of the overhead when the PC size is 1 and increases as the PC size is bigger. Precisely, O_PBS spends most of its time in waiting until the PBS worker daemons report their states as free while launching the PBS server and workers themselves takes less than 1 second. By contrast, the other factors are constant regardless of the PC size. The overhead of processing a GRAM service request is about 2 seconds while launching a Globus container takes about 3 seconds. The OS-related overhead is about 2.1 seconds which consists of 2 second sleep for retrial and the net OS overhead of one hundred milliseconds. Finally, the delay due to notification of Globus container is about 2 seconds.

```
Cluster host = new Cluster ("bowhead.cs.ucsb.edu",   // front-end hostname
                            "PBS"          // host resource manager name
                            32,            // maximum number of processors
                            16,            // maximum number of hosts
                            60 * 12        // maximum duration
                    );
Cluster guest = new Cluster ("bowhead.cs.ucsb.edu", // front-end hostname
                             "PBS",         // my job manager name
                             16,            // # of processors
                             8,             // # of hosts
                             60             // duration
                    );
PersonalClusterFactory factory = new PersonalClusterFactory ();
factory.registerHostCluster (host);
PersonalCluster pc = factory.createPersonalCluster (guest);

if (pc.isReady ()) {
   // do something with this personal cluster
}
```

Figure 3. An example of creating a personal cluster using JAVA APIs

4. Related Work

The virtual cluster or COD (Cluster on Demand) technology [8] shares the goal with our study in the sense that it provides an illusion to the user as if a set of processors is dedicated on demand. We understand that virtual cluster based on VM (Virtual Machine) technology is promising and provide a variety of sophisticated features for deployment, security, fault-tolerance, reproducibility, and so on. However, overcoming the performance degradation due to virtualization overhead is critical especially for high performance computing [9]. By contrast, personal cluster can exploit full performance of resources since it is simple and is directly implemented on top of physical resources. Note that personal clusters can be deployed to virtual clusters or COD as well.

In the meantime, Condor-G [10] allows the users to dynamically add resources via the Condor GlideIn mechanism to its resource pool on demand when Condor-G is integrated with Condor-controlled resources. In terms of mechanism, personal cluster is similar to Condor GlideIn since a job manager slides into the allocated resources. Different from Condor GlideIn, however, personal clusters can be arbitrarily coupled with a variety of batch systems including PBS, LSF, Condor, SGE, and Loadleveler. Moreover, an application has a dedicated resource manager while jobs submitted to Condor-G still go through shared Condor queues for Condor-G.

5. Conclusion

In this paper, we presented a personal cluster technique that provides a homogeneous dedicated computing environment over heterogeneous clusters. This technique not only provides the user a uniform interface to the allocated resources but also enables efficient task management in a cost-effective manner, exploiting the features of commodity tools. Personal cluster can not only be flexibly coupled with popular batch systems but also exploit full performance of resources without performance degradation. A prototype implementation based on Globus GRAM service and a PBS installation shows that personal clusters can be instantiated with small overhead.

We are currently using this mechanism to actualize Virtual Grid [11-13]. Java APIs for embedding this feature into users' programs and a script package for command line tool are available for Globus-based personal cluster while a script package is available for secure shell based personal cluster. Figure 3 presents an example of code segment that creates personal clusters using Java APIs. With these simple APIs, for instance, the users can easily embed the feature of personal cluster into their application managers. More details about the personal cluster packages will be available at http://csag.ucsd.edu/individual/yskee.

6. Reference

[1] R. L. Henderson, "Job Scheduling Under the Portable Batch System," in *Lecture Notes in Computer Science*, vol. 949, *IPPS '95 Workshop on Job Scheduling Strategies for Parallel Processing*: Springer, 1995, pp. 279-294.

[2] M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," in *IEEE International Conference on Distributed Computing Systems (ICDCS-8)*: IEEE, 1988, pp. 104-111.

[3] S. Zhou, "LSF: Load sharing in large-scale heterogeneous distributed systems," in *International Workshop on Cluster Computing*: IEEE, 1992.

[4] K. Yoshimoto, P. Kovatch, and P. Andrews, "Co-Scheduling with User-Settable Reservations," in *Lecture Notes in Computer Science*, vol. 3834, *Workshop on Job Scheduling Strategies for Parallel Processing*: Springer, 2005, pp. 146-156.

[5] F. Berman, "Viewpoint: From TeraGrid to Knowledge Grid," *Communications of the ACM*, vol. 44, pp. 27-28, 2001.

[6] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in *Lecture Notes in Computer Science*, vol. 3779, *IFIP International Conference on Network and Parallel Computing*: Springer, 2005, pp. 2-13.

[7] C. R. Inc., "TORQUE v2.0 Admin Manual."

[8] J. Chase, D. Irwin, L. Grit, J. Moore, and S. Sprenkle, "Dynamic Virtual Clusters in a Grid Site Manager," in *IEEE Symposium on High Performance Distributed Computing (HPDC-12)*. Seattle, Washington: IEEE, 2003.

[9] Y. Dong, S. Li, A. Mallick, J. Nakajima, K. Tian, X. Xu, F. Yang, and W. Yu, "Extending Xen with Intel Virtualization Technology," *Intel Technology Journal*, vol. 10, pp. 193-204, 2006.

[10] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," in *IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*: IEEE, 2001, pp. 55-63.

[11] Y.-S. Kee, D. Logothetis, R. Huang, H. Casanova, and A. A. Chien, "Efficient Resource Description and High Quality Selection for Virtual Grids," in *ACM/IEEE International Symposium on Cluster Computing and the Grid (CCGRID'05)*. Cardiff, United Kingdom: IEEE, 2005, pp. 598-606.

[12] Y.-S. Kee, K. Yocum, A. A. Chien, and H. Casanova, "Improving Grid Resource Allocation via Integrated Selection and Binding," in *ACM/IEEE International Conference on High Performance Computing and Communication (SC'06)*. Tampa, United States: IEEE, 2006.

[13] Y.-S. Kee and C. Kesselman, "Grid Resource Abstraction, Virtualization, and Provisioning for Time-targeted Applications," in *ACM/IEEE International Symposium on Cluster Computing and the Grid (CCGRID'08)*: IEEE, 2008.

Biographies

Yang-Suk Kee is a post-doctoral research associate under the supervision of Dr. Carl Kesselman in the Information Sciences Institute at the University of Southern California. He received a Ph.D. in Electrical Engineering and Computer Science, a Master of Science degree in Computer Engineering, and Bachelors degrees in Computer Engineering from Seoul National University. He was a visiting scholar and post-doctoral researcher at University of California, San Diego under the supervision of Dr. Andrew Chien and Dr. Henri Casanova.

Carl Kesselman is Fellow in the Information Sciences Institute at the University of Southern California. He is the Director of the Center for Grid Technologies at the Information Sciences Institute and a Research Professor of Computer Science at the University of Southern California. He received a Ph.D. in Computer Science from the University of California, Los Angeles, a Master of Science degree in Electrical Engineering from the University of Southern California, and Bachelors degrees in Electrical Engineering and Computer Science from the University at Buffalo. Dr. Kesselman also serves as Chief Scientist of Univa Corporation, a company he founded with Globus co-founders Ian Foster and Steve Tuecke. Dr. Kesselman's current research interests are all aspects of Grid computing, including basic infrastructure, security, resource management, high-level services and Grid applications. He is the author of many significant papers in the field. Together with Dr. Ian Foster, he initiated the Globus Project™, one of the leading Grid research projects. The Globus project has developed the Globus Toolkit®, the de facto standard for Grid computing. Dr. Kesselman received the 1997 Global Information Infrastructure Next Generation Internet award, the 2002 R&D 100 award, the 2002 R&D Editors choice award, the Federal Laboratory Consortium (FLC) Award for Excellence in

Technology Transfer and the 2002 Ada Lovelace Medal from the British Computing Society for significant contributions to information technology. Along with his colleagues Ian Foster and Steve Tuecke, he was named one of the top 10 innovators of 2002 by InfoWorld Magazine. In 2003, he and Dr. Foster were named by MIT Technology Review as the creators of one of the "10 technologies that will change the world." In 2006 Dr. Kesselman received an Honorary Doctorate from the University of Amsterdam.

Dan Nurmi is a Ph.D. candidate at the University of California Santa Barbara under the advisorship of Dr. Rich Wolski. He has studied computer science at the University of Minnesota (Minneapolis Campus), Tulane University and the University of Chicago where he received an M.S. in computer science in 2002. From 1998 to 2002, he held a position as Systems Engineer at the Math and Computer Science department of Argonne National Laboratory near Chicago, where he help design and develop software and management techniques for several large scale HPC systems. His research efforts include large scale system management, resource failure prediction, batch queue delay prediction, distributed workflow scheduling, and virtual resource definition.

Rich Wolski is a Professor in the Computer Science department at the University of California, Santa Barbara (UCSB). Having received his M.S. and Ph.D. degrees from the University of California at Davis (while he held a full-time research position at Lawrence Livermore National Laboratory) he has also held positions at the University of California, San Diego, and the University of Tennessee. He is currently also a strategic advisor to the San Diego Supercomputer Center and an adjunct faculty member at the Lawrence Berkeley National Laboratory. Dr. Wolski heads the Middleware and Applications Yielding Heterogeneous Environments for Metacomputing (MAYHEM) Laboratory which is responsible for several national scale research efforts in the area of high-performance distributed computing and grid computing. These efforts have resulted in nationally supported production-quality software tools such as the Network Weather Service (currently distributed as part of the NSF Middleware Initiative's baseline software distribution) and a supported international user-community in addition to an extensive scholarly corpus. His most recent efforts have focused on both the implementation of nationally distributable tightly-coupled programs as well as the development of statistical predictive techniques for resource-constrained power usage, resource failure prediction, and batch queue delay prediction.