Phoenix Cloud: Consolidating Different Computing Loads on Shared Cluster System for Large Organization

Jianfeng Zhan[*], Lei Wang[*], Bibo Tu[*], Yong Li[*], Peng Wang[*], Wei Zhou[*] and Dan Meng[*]
[*]Institute of Computing Technology,Chinese Academy of Sciences Beijing, China 100190
[†] Email: jfzhan@ncic.ac.cn

***Abstract*—Different departments of a large organization often run dedicated cluster systems for different computing loads, like HPC (high performance computing) jobs or Web service applications. In this paper, we have designed and implemented a cloud management system software *Phoenix Cloud* to consolidate heterogeneous workloads from different departments affiliated to the same organization on the shared cluster system. We have also proposed cooperative resource provisioning and management policies for a large organization and its affiliated departments, running HPC jobs and Web service applications, to share the consolidated cluster system. The experiments show that in comparison with the case that each department operates its dedicated cluster system, *Phoenix Cloud* significantly decreases the scale of the required cluster system for a large organization, improves the benefit of the scientific computing department, and at the same time provisions enough resources to the other department running Web services with varying loads.** [1]

I. Introduction

Since 2007, a client from a large organization, which we keep anonymous at its request, requires us to build the system software for managing a shared infrastructure. This large organization has two representative departments: *one running a batch queuing system for HPC jobs*, and *the other one responsible of providing Web services*, of which the ratios of peak loads to normal loads are high. So two representative departments from this big organization have operated two cluster systems with independent administration staffs and found many annoying problems: first, resource utilization rates of two cluster systems are varying. For peak loads, dedicated cluster systems can not provision enough resources, while for normal loads lots of resources are idle; second, the number of administration staffs for two separated cluster systems is high. The client inquired us whether it is possible to help them consolidate two cluster systems on one shared system.

At same time, we have noticed that many famous IT companies are advocating and experiencing cloud computing. For example, Amazon [1] has provided cloud computing services like *elastic computing cloud (EC2)* and *simple storage service (S3)* to end users. What is the link between services provided by Amazon and the requirement of our anonymous client? In our opinion, driven by the cost, cloud computing is a new wave of reconstructing and consolidating data centers. Traditional cluster system software is self-containing [2], inadequate for adapting to this change. EC2 and S3 are big efforts to provide virtualized hosting environments for end users, but it can not provide the customized system stack software to consolidate heterogeneous loads on shared cluster systems for large organizations. In fact, there lies no one-fit-all solution.

In this paper, we focus on developing cloud computing management software that enables the consolidation of heterogeneous workloads on shared cluster systems for large organizations, and we stress that *we do not target the design of capability-oriented system software stack* [3]. To the best of our knowledge, this is the first paper to propose the layered architecture of cloud computing management software for large organizations that intend to consolidate heterogeneous workloads from different departments on shared cluster systems. [4] proposes the utility computing service framework to facilitate code reuse in the context of traditional data centers, but do not consider how to enable consolidating different types of workloads. [5] [6] propose *Cluster on Demand*(COD) as a new mechanism for dynamical cluster resource management in the contexts of Internet hosting center [5] or scientific computing [6], but their works mainly focus on dynamic resource provisioning in respective computing contexts.

The distinguished differences of our system and architecture from others are that: first, we develop a common service framework as a foundation for cloud computing system software; second, with the support of a common service framework, we create cloud management services respectively for scientific computing (HPC jobs) and web service applications; third, we propose optimal resource management and provision policies for heterogeneous workloads to cooperatively share cluster resources. The contribution of this paper can be concluded as:

- We have designed and implemented a cloud management system software *Phoenix Cloud* with the *layered architecture* to consolidated HPC jobs and Web service applications on shared cluster systems.
- We have proposed cooperative resource provisioning and management policies for a large organization and its

[1]This document is dated from August 13, 2008 and contains an early summary of experiences of this project, which also available from the web site of *the First Workshop of Cloud Computing and its Application (http://www.cca08.org/papers/Poster-8-Jianfeng-Zhan.pdf)*. The extended version with the title of *PhoenixCloud: Provisioning Resources for Heterogeneous Workloads in Cloud Computing* can be downloaded from http://arxiv.org/abs/1006.1401.

affiliated departments to share the cluster system.

- Our experiments show that in comparison with the case that each department maintains its dedicated cluster system, consolidating HPC jobs and Web service applications with cooperative resource provisioning and management policies can significantly decrease the scale of the required cluster system for a large organization, at the same time improve the benefit of the scientific computing department while provisioning enough resources to the department who runs Web service applications with varying loads.

Our paper includes four sections. In Section II, we explain the design and implementation issue of *Phoenix Cloud*. In Section III, we evaluate our system. In Section IV, we draw a conclusion.

II. Design and Implementation Issues

In Section II-A, we introduce the layered architecture of *Phoenix Cloud*. In Section II-B, we propose cooperative resource provision and management policies for *Phoenix Cloud*.

A. The Layered Architecture of Phoenix Cloud

We divide our cloud computing management software into three independent layers: *shared infrastructure for the resource provider*, *cloud management services for service providers who are different departments affiliated to the same large organization*, and *client tools for end user*. Fig. 1 shows the macro-level architecture of our innovative system as follows:

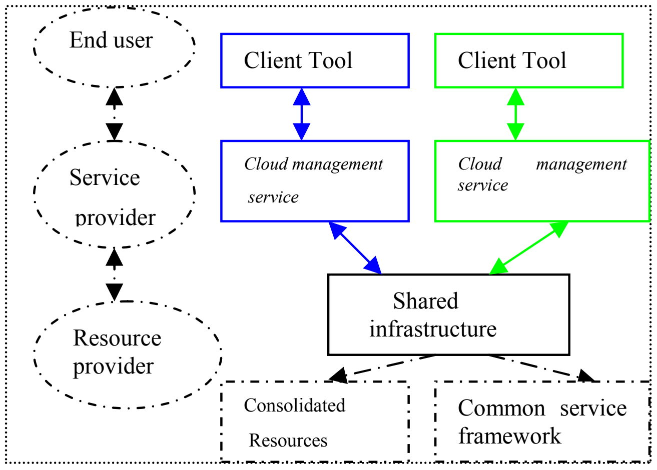

Fig. 1. The layered architecture of *Phoenix Cloud*.

- The resource provider is responsible for operating *the shared infrastructure*, including *the shared cluster resources* and *the common service framework*. The shared cluster resources include hardware resources, e.g. CPU, memory, and system software like host operating systems. *The common service framework* provides a set of services that manage, monitor the shared cluster resources and provision resources to *cloud management services* for different service providers.
- *The cloud management service* (CMS) is a management service for a specific computing load, the implementation detail of which is seen in Fig. **??**.
- Client tools: end users use client tools to access services or submit jobs.

Fig. 2 shows the micro-level architecture of *Phoenix Cloud* when two *cloud management services* share a cluster system and reuse *the common service framework*.

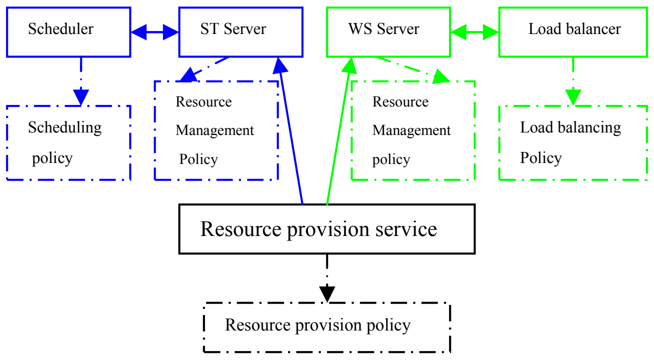
Scheduler
ST Server
WS Server
Load balancer
Scheduling policy
Resource Management Policy
Resource Management policy
Load balancing Policy
Resource provision service
Resource provision policy

Fig. 2. The micro-level architecture of *Phoenix Cloud*.

One is *a cloud management service for scientific computing like HPC jobs)* (in short, *ST CMS*), including *ST Server* and *Scheduler*, and the other is *a cloud management service for Web services* (*in short, WS CMS*), including *WS Server* and *Load balancer*:

- Among *the common service framework*, a service named *Resource Provision Service* with *the customized resource provisioning policy* acts as the proxy of a large organization, responsible for managing and provisioning resource to *different cloud management services*.
- *The resource provision policy* determines when *Resource Provision Service* will provision how many resources to different *cloud management services* in what priority.
- The *cloud management service* with *a customized resource management policy* and *a scheduling/load balancing policy* behaves as the representative of a service provider, responsible for managing resources, scheduling jobs or distributing requests for load balancing.
- *The resource management policy* of a service provider determines when *ST Server* or *WS Server*obtains or returns how many resources to *Resource Provision Service* according to what criteria.
- *The scheduling policy* determines *Scheduler* of *ST CMS* when and how to choose HPC jobs for running. *The load balancing policy* determines *Load Balancer* of *WS CMS* how to distribute requests and adjust Web service instances according to what criteria.

Our innovative system evolves from our previous work *Phoenix*, which is a cluster system software stack [8] [9]. Based on *Phoenix*, we have consolidated two different *cloud management services* respectively for HPC jobs and Web services on the shared cluster system. Fig.3 shows the architecture

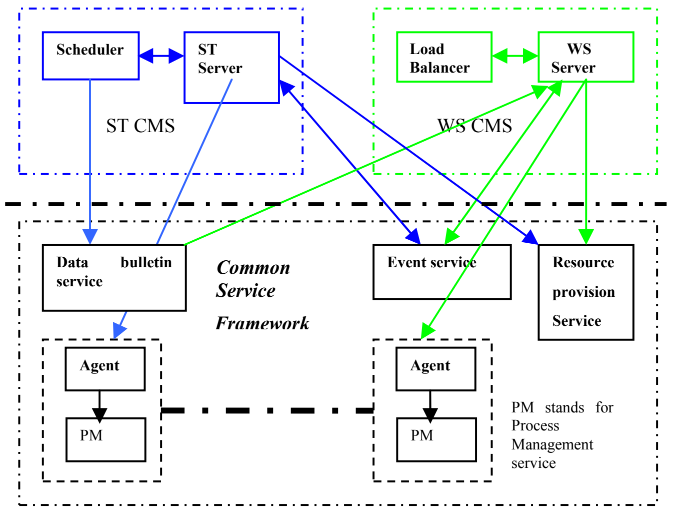

Fig. 3. With *Phoenix Cloud*, two cloud management services reuse and share the common service framework.

of *ST CMS* and *WS CMS* based on *Phoenix*. The function of *ST CMS* is similar to OpenPBS [10], while the function of *WM CMS* is similar to the Oceano [11]. But distinguished differences of our systems are as follows:

- Different heterogeneous workloads can be consolidated on the shared system;
- Different cloud management services can reuse the common service framework;
- Different cloud management services can cooperatively share resources under varying loads according to the cooperative policies proposed in section II-B.

B. Cooperative Resource Provision and Management Policies

In this section, we propose cooperative resource provisioning and management policies for a large organization and its affiliated departments. As shown in Fig.2, we could specify *the resource provisioning policy* for *Resource Provision Service*, and different resource management policies for *ST Server* and *WS Server*. *The resource provisioning policy* is as follows:

- The resource demands from *WS Server* have higher priority than that of *ST Server*.
- If there are idle resources for *Resource Provision Service*, it will provision all idle resources to *ST Server*.
- If *WS Server* claims urgent resources, *Resource Provision Service* will force *ST Server* to return resources with the size claimed by *WS Server* and then reallocate to *WS Server*.

The resource management policy of *ST server* is as follows:

- *ST Server* passively receives resources provisioned by *Resource Provision Service*.
- If *Resource Provision Service* forces *ST Server* to return resources, the latter will release resources immediately with the size demanded by the former.
- If there are no enough idle resources for *ST Server*, it will kill jobs in turn from the beginning of job with minimum size and shortest running time, and release enough resources to *Resource Provision Service*.

The resource management policy of *WS Server* is as follows:

- If *WS Server* owns idle resources, it will release them to *Resource Provision Service* immediately. If *WS Server* needs more resources, it will request enough resources from *Resource Provision Service*.

III. Evaluation and Discussion

In this section, we will demonstrate that with the cooperative resource provision and management policies proposed in Section II-B, consolidating HPC jobs and Web services on the shared cluster system, of which we call *dynamic configuration*, can decrease the cost of a large organization in term of *resource consumption* in comparison with *the static configuration*, of which each department maintains its dedicated cluster system.

A. The benefit and cost models

For a large organization, we use *the size of nodes* to measure the cost of owning a cluster system. For HPC jobs, we use *the number of completed jobs* to measure the benefit of a service provider; at the same time, we use *the reciprocal of the average turnaround time per job* to measure the benefit of end user. For Web service application, we use *the throughput* in term of *request/second* to measure the benefit of a service provider; at the same time we use *the average response time of requests* to measure the benefit of an end user.

B. Experiment method and load traces

Our experiments include two parts: first, in Section III-C we obtain the real resource consumption of a Web service application under varying loads on the testbed. Second, based on the real resource consumption of a Web service application obtained in Section III-C, we use a simulation method to obtain the resource consumption in the case of consolidating different computing loads from different departments of a large organization on the shared cluster system. The synthetic request trace of Web service application is obtained from the real trace of World Cup of two week from June 7 in 1998 [12] with a scaling factor of 2.22. For the World Cup trace, the ratio of the peak load to the normal load is high. HPC trace is the real trace of SDSC BLUE of two weeks from Apr 25 15:00:03 PDT 2000 on the web site of http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.

C. The resource consumption of Web service under varying load

The testbed is as follows: All nodes are connected with a 1 Gb/s switch. Each node has the same configuration: 8 $\times$ Intel(R) Xeon(R) (2.00GHz) CPU and 2G memory with 64 bit Linux with kernel of 2.6.18-xen. On each node, we deploy eight XEN [13] virtual machines. The configuration of XEN virtual machine is: 1 $\times$ Intel(R) Xeon(R)(2.00GHz) CPU; 256M memory; the guest operating system is 64 bit CentOS with kernel version of 2.6.18.

Fig. 4 shows the system deployment diagram. We choose httperf [14] as the load generator, LVS[15] with direct route mode is responsible for distributing requests to the Web service with the least-connection scheduling policy. The DNS server is responsible for distributing connection from each user to one of four LVS with round robin policy. We choose open source software ZAP! [7] as a typical Web service, and each instance of ZAP! is deployed on a virtual machine.

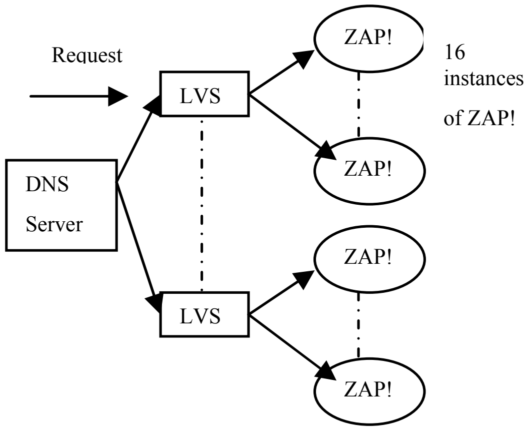

Fig. 4. The system deployment diagram.

WS Server adjusts *the number of instances of Web services* according to the criterion in terms of *the average utilization rate of CPU consumed by Web service instances*. We presume the number of current instances of information service is n. If *the average utilization rate of CPUs consumed by Web service instances* exceeds 80% in the past 20 seconds, *WS Server* will increase one instance. If *the average utilization rate of CPUs consumed by Web service instances* is lower than $80\%(n-1)/n$ in the past 20 seconds, *WS Server* will decrease one instance until the number of the current instances is equal to 1. We use the Web service trace described in Section III-B, and Fig.5 shows the varying resource consumption in two weeks, of which the peak resource demand is 64 virtual machines.

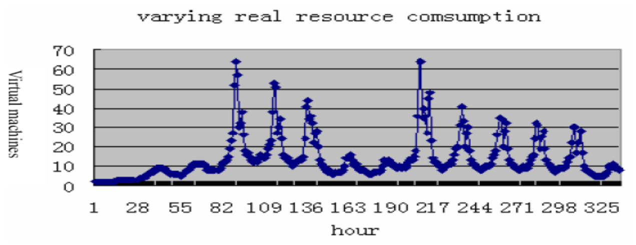

Fig. 5. The resource consumption of Web service trace in two weeks.

D. The simulation experiments of consolidating computing loads

We use a simulation method to verify the advantage of consolidating different computing loads from different departments of a large organization on the shared cluster system. Fig. 6 shows the architecture of our simulation system, which includes *one cloud management service for scientific computing* (*ST CMS*) and *one cloud management service for Web service* (*WS CMS*). In comparison with the real *Phoenix Cloud* system, our simulated system maintains *Resource Provision Service*, *WS Server*, *ST Server* and *Scheduler*, while other services are removed or substituted.

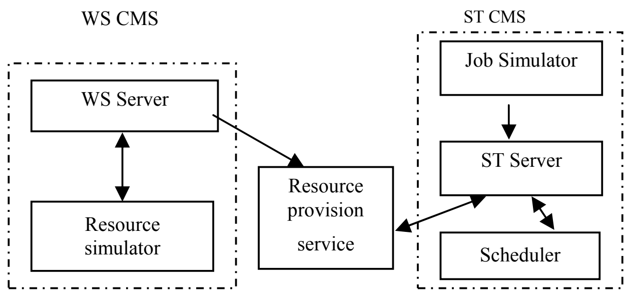

Fig. 6. The architecture of the simulated system.

For *WS CMS*, a daemon named *Resource Simulator* will simulate the varying resource demand of *WS CMS* and drive *WS Server* to obtain or release resources from and to *Resource Provision Service*. We use the real resource consumption in Fig. III-C as the input to *Resource Simulator*. For *ST CMS*, *Scheduler* is specified with *the First-Fit* scheduling policy, and *Job Simulator* is used to simulate the process of submitting jobs. To accelerate the experiment, we speed up the submission and completion of jobs by a factor of 100. This speedup allows two weeks trace to complete in about three hours. The HPC trace is introduced in Section III-B. We presume that the software package of Web service are pre-deployed on those reallocated nodes, so the time of reallocating nodes from *ST Server* to *WS server* is only seconds, includes *the time of killing jobs* and *the time of communicating among* WS Server, ST Server *and* Resource Provision Service.

In our simulation system, for *static configuration* (in short *SC*), of which each department of a large organization maintains its own cluster system, the minimum scale of the cluster system for HPC trace introduced in Section III-B is 144 nodes, because the real SDSC trace is also collected from the same 144 nodes; the minimum scale of the cluster system for Web service is 64 nodes, because the peak resource demand in Fig. 5 is 64 virtual machines. So *the size of the cluster configuration* allocated to Web services and HPC jobs for *SC* is 208. For *the dynamic configuration* (in short *DC*), we respectively set *the size of the cluster configuration* allocated to Web services and HPC jobs as 200, 190, 180, 170, 160 and 150. Fig. 7 shows *the number of completed jobs* and *the average turnaround time per job in term of seconds* for HPC trace in two weeks when we set different *size of the cluster configuration*.

For HPC trace, 2672 jobs are submitted to *ST Server*. For *dynamic configuration*, when the cost of the large organization in term of *the size of the cluster configuration* decreases to

160, only 76.9% of that of *static configuration*, the benefit of scientific computing department in term of *the number of completed jobs* in two weeks is still higher than that of *static configuration*; while the benefit of end user in term of *the reciprocal of the average turnaround time per job* is still higher than that of *static configuration*. With *the size of the cluster configuration* decreases, *the number of killed jobs* increases in general. Only the exception is the number of killed jobs when *the size of the cluster configuration* is 170, which is higher than that when *the size of the cluster configuration* is 160.

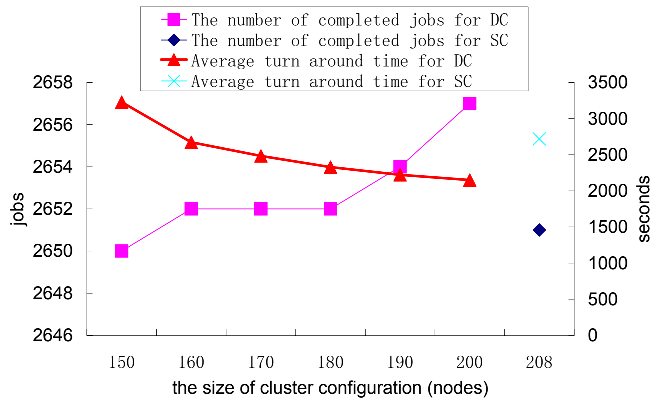

Fig. 7. For HPC trace, the number of completed jobs and the average turnaround time per jobs in two weeks with different size of the cluster configuration.

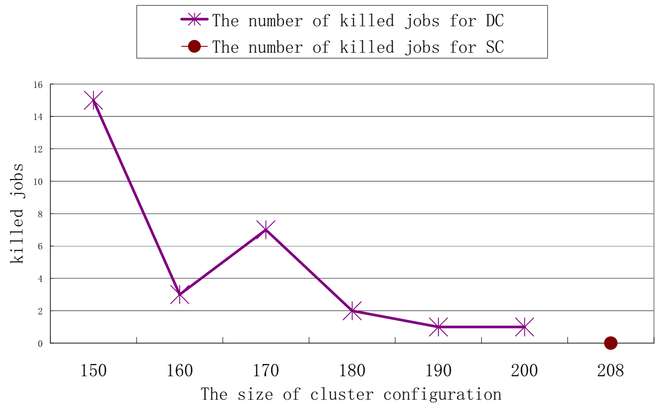

Fig. 8. For HPC trace, the number of killed jobs in two weeks with different size of the cluster configuration.

For Web service, the benefits of service providers and end users are unchanging, since we just use the same resource consumption collected from Section III-C as the input to *Resource Simulator*.

IV. Conclusions

Different departments of large organizations often maintain dedicated cluster systems for different computing loads. In this paper, we have designed and implemented a cloud management system software *Phoenix Cloud* to consolidate HPC jobs and Web service application on the shared cluster system. We have also proposed cooperative resource provisioning and management policies of large organizations and their affiliated departments to share the consolidated cluster systems. Our experiments show that in comparison with the case that each department of the same large organization runs its dedicated cluster system, consolidating HPC jobs and Web service applications from different departments with *cooperative resource provisioning and management policies* not only significantly decreases the scale of the required cluster system for a large organization, but also improves the benefit of the scientific computing departments while provisioning enough resources to the other department running Web service applications with varying loads.

V. Acknowledgements

This paper is supported by the National Science Foundation for Young Scientists of China (Grant No. 60703020).

References

[1] Amazon: http://aws.amazon.com/
[2] Raghavan, B., Vishwanath etc, . Cloud control with distributed rate limiting. In Proceedings of SIGCOMM ’07. ACM, New York, NY, 337-348.
[3] Jean-Charles Tournier, Patrick G. Bridges etc, Towards a Framework for Dedicated Operating Systems Development in High-End Computing Systems, ACM SIGOPS Operating Systems Review, Volume 40 , Issue 2, April 2006
[4] Eilam, T., Appleby etc, Using a utility computing framework to develop utility systems. IBM Syst. J. 43, 1 (Jan. 2004), 97-120.
[5] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. 2001. Managing energy and server resources in hosting centers. SOSP ’01. ACM, New York, NY, 103-116.
[6] Chase, J. S., Irwin etc,. Dynamic Virtual Clusters in a Grid Site Manager. HPDC 03.
[7] http://www.indexdata.dk/
[8] Jianfeng Zhan, Ninghui Sun, Fire Phoenix Cluster Operating System Kernel and its Evaluation, Proceeding of IEEE Cluster 2005, Boston, MA, USA.
[9] Jianfeng Zhan, Lei Wang etc, The Design Methodology of Phoenix System Software Stack, Workshop on High Performance Computing in China: Solution Approaches to Impediments for High Performance Computing in conjunction with SC 07.
[10] OpenPBS: http://www-unix.mcs.anl.gov/openpbs/
[11] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. P. Pazel, J. Pershing, B. Rochwerger, ”Ocano–SLA Based Management of a Computing Utility,” Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management, IEEE, New York (2001).
[12] Martin Arlitt, Tai Jin, Workload Characterization of the 1998 World Cup Web Site, Copyright Hewlett-Packard Company, 1999
[13] XEN: http://www.xen.org
[14] http://www.hpl.hp.com/research/linux/httperf/
[15] LVS: http://www.linuxvirtualserver.org/