

# 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, Dan Meng

Institute of Computing Technology  
Chinese Academy of Sciences, Beijing 100190, China

jfzhan@ncic.ac.cn

## ABSTRACT

Different departments of large organization 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 heterogeneous computing loads on shared cluster system. Furthermore, we have proposed a cooperative resource provision and management policy for large organization and its affiliated departments to share the aggregated cluster system. The experiments show: comparing with the case that each department maintains its dedicated cluster system, Phoenix Cloud enables the consolidation of scientific computing and information service; Moreover, Phoenix Cloud significantly decreases the scale of required cluster system for large organization, and improves the benefit of scientific computing department and its end users while provisioning enough resource to information service with varying load.

## 1. INTRODUCTION

Since 2007, a client from a large organization, which we keep anonymous at its request, requires us to build system software for managing a shared infrastructure. This large organization has two representative departments: one maintaining a batch queuing system for scientific computing, and the other one responsible of providing information service, of which the ratio of the peak load to normal load is high. So two representative departments from this big organization have operated two clusters with independent administration staffs and found many annoying problems: first, the resource utilization rates of two cluster systems are varying. For peak load, the dedicated cluster can not provision enough resource, while for normal load 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 the service provided by Amazon and the requirement of our anonymous client? In our opinion, driven by the cost, the cloud computing is a new wave of reconstructing and consolidating data center. 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 the shared cluster system for large organization. In fact, there lies no one-fit-all solution.

In this paper, we focus on developing cloud computing management software which enables the consolidation of heterogeneous computing loads on shared cluster system for large organization, 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 layered architecture of cloud computing management software for large organization to consolidate heterogeneous computing loads from different departments on shared cluster system. [4] proposes the utility computing service framework to facilitate the code reuse in the context of traditional data center, but do not consider how to enable consolidating of different kinds of computing. [5] [6] propose the COD (Cluster on Demand) as new mechanism for dynamical cluster resource management in the context of Internet hosting center [5] or scientific computing [6], but their works mainly focus on the dynamic resource provision in respective computing contexts.

The distinguished difference of our system and architecture from others is that: first, we develop *common service framework* as the foundation of cloud computing system software; second, with the support of common service framework, we create *cloud management services* respectively for scientific computing and information service; third, we propose optimal resource management and provision policy for heterogeneous computing loads to cooperatively share cluster resource. The contribution of this paper can be concluded as:

(1) We have designed and implemented a cloud management system software-Phoenix Cloud with layered architecture to consolidated scientific computing and information service on shared cluster system.

(2) We have proposed a cooperative resource provision and management policy for large organization and its affiliated departments to share the cluster system.

(3) Our experiments show: comparing with the case that each department maintains its dedicated cluster system, the consolidation of scientific computing and information service with cooperative resource provision and management policy can significantly decrease the scale of required cluster system for large organization, at the same time improve the benefit of scientific computing department and its end users while provisioning enough resource to information service with varying load.

This structure of our paper includes four sections. In the section 2, we explain the design and implementation issue of Phoenix Cloud. In the section 3, we evaluate the new system. In section 4, we draw a conclusion.

## 2. THE DESIGN and IMPELMENTATION ISSUE

In section 2.1, we introduce the layered architecture of Phoenix Cloud. In section 2.2, we propose cooperative resource provision and management policy of Phoenix Cloud.

### 2.1. Layered Architecture of Phoenix Cloud

We divide the cloud computing management software into three independent layers: shared infrastructure for resource provider-large organization, cloud management services for service provider-different departments, and client tool for end user. Figure 1 shows the macro-level architecture of our new system as follows:

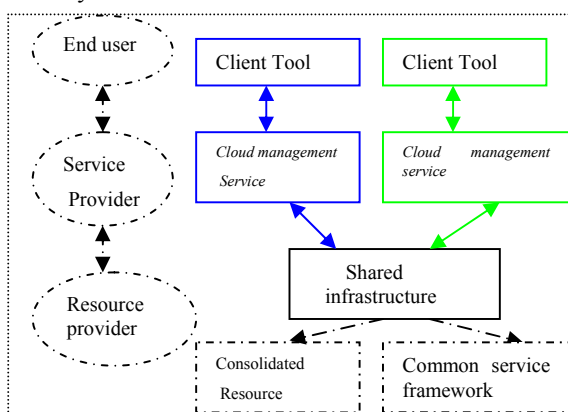


Figure 1. Layered architecture of Phoenix Cloud

(1) The *shared infrastructure* is for resource provider, which includes *shared resource* and *common services framework*. *Shared resource* includes hardware resources, e.g. CPU, memory, and system software,

e.g. host operating system.

(2) The *common services framework* is provided by resource provider as a set of services to manage and monitor the shared resource, provision resource to different *cloud management services* for different service providers.

(3) The *cloud management service (CMS)* is the management service for specific computing load, the implementation detail of which is seen in Figure 3.

(4) *Client tool*: the end user uses client tool to access service or submit job.

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

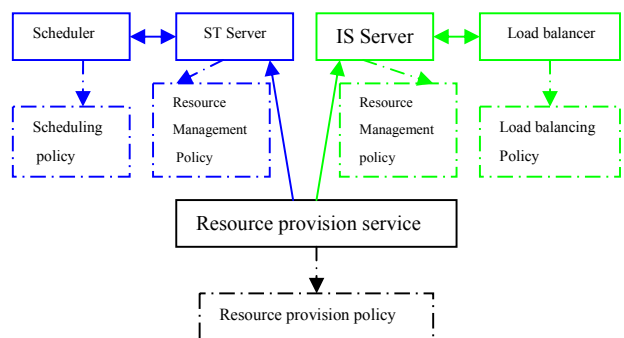


Figure 2. Micro-architecture of Phoenix Cloud.

One is cloud management service for scientific computing (ST CMS), including ST Server and scheduler, and the other is cloud management service for information service (IS CMS), including IS Server and load balancer:

(1) Among common service framework, a service named *resource provision service* with customized *resource provision policy* acts as the proxy of large organization, responsible for managing and provisioning resource to different cloud management services.

(2) *Resource provision policy* determines when the resource provision service will provision how much resource to different cloud management services in what priority.

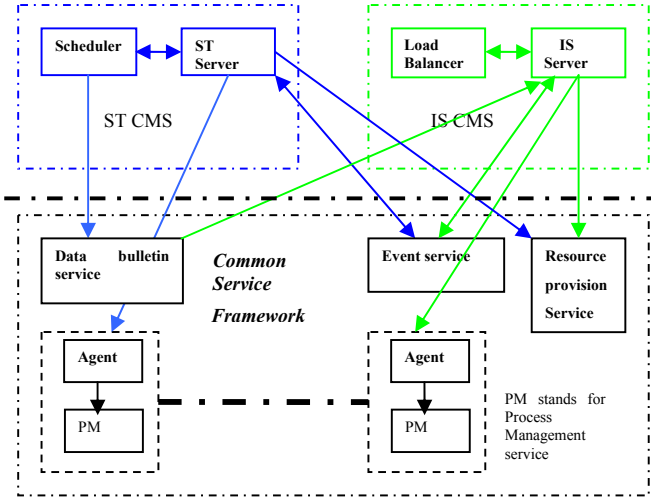
(3) The *cloud management service* with customized *resource management policy* and *scheduling/load balancing policy* behaves as the representative of service provider, responsible for managing resource, scheduling job or distributing requests for load balancing.

(4) The *resource management policy* of service provider determines when Server (ST Server or IS Server) obtains or returns how much resource to the resource provision service according to what criteria.

(5) *Scheduling policy* determines the scheduler of ST CMS when and how to choose scientific computing job for running.

(6) *Load balancing policy* determines the load balancer of

IS CMS how to distribute requests and adjust instances of information service according to what criteria.



**Figure 3. For Phoenix Cloud, two cloud management services reuse and share common service framework.**

Our new system evolves from Phoenix cluster system software stack [8] [9]. Based on Phoenix common service framework, we have consolidated two different cloud management services respectively for scientific computing and information service on shared cluster system. Figure 3 shows the architecture of ST CMS and IS CMS based on the Phoenix common service framework. The function of ST CMS is similar to OpenPBS [10], while the function of IM CMS is similar to the Océano [11]. But the distinguished difference of our new systems is that (1) they are consolidated on the shared system; (2) they reuse common service framework; (3) They could cooperatively share resource under varying load condition according to the cooperative policy proposed in section 2.2.

## 2.2. Cooperative Resource Provision and Management Policy

In this section, we propose a cooperative resource provision and management policy for large organization and its affiliated departments. As shown in Figure 2, we could specify the resource provision policy for resource provision service, different resource management policies for ST (Scientific Computing) Server and IS (Information Service) Server. The resource provision policy is as follows:

- The resource demands from IS Server have higher priority than that of ST Server.
- If there is idle resource for resource provision service, it will provision all idle resource to ST Server.
- If IS Server claims urgent resource, the resource provision service will force ST Server to return resource with the size claimed by IS Server and then reallocate to the IS Server.

The resource management policy of ST server is as follows:

- The ST Server silently receives the resource provisioned by resource provision service.
- If the resource provision service forces the ST Server to return resource, it will return resource immediately with the size demanded by resource provision service.
- If there is no enough idle resource for ST Server, it will kill jobs in turn from the beginning of job with minimum size and shortest running time, and return enough resource to resource provision service.

The resource management policy of IS server is as follows:

- If there is idle resource for IS Server, it returns to resource provision service immediately. If IS Server needs more resource, it will ask for enough resource from the resource provision service.

## 3. EVALUATION AND DISCUSSION

In this section, we will demonstrate that with the cooperative resource provision and management policies proposed in section 2.2, the consolidation of scientific computing and information service on shared cluster system, of which we call *dynamic configuration*, can decrease the cost in term of resource consumption of large organization, comparing with the *static configuration*, of which each department maintains its own cluster system.

### 3.1. The benefit and cost models

For large organization, we briefly use the *size of nodes* to measure the cost of owning cluster system.

For scientific computing, we use *number of completed jobs* to measure the *benefit* of service provider; at the same time, we use the *reciprocal of average turnaround time per job* to measure the benefit of end user.

For information service, we briefly use *throughput in term of request/second* to measure the benefit of service provider; at the same time we use *average response time of requests* to measure the *benefit* of end user.

### 3.2. Experiment method and load traces

Our experiments include two parts: first, in section 3.3 we obtain the real resource consumption of information service under varying load on the testbed. Second, based on the real resource consumption of information service obtained in section 3.3, we use the simulation method to obtain the real resource consumption when consolidating different computing loads from different departments on shared cluster system.

The synthetic request *trace of information service* is obtained from the real trace of World Cup load of two week from June 7 to June 20 in 1998[12] with a scaling factor of 2.22, of which the ratio of peak load to normal load is high.

*Scientific computing 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>.

### 3.3. The resource consumption of information service under varying load

The testbed is as follows: All nodes are connected with a 1 Gb/s switch. Each node has same configuration: CPU-8×Intel(R) Xeon(R) (2.00GHz); memory- 2G; OS- 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: CPU-1×Intel(R) Xeon(R)(2.00GHz); memory- 256M; the guest operating system is 64-bit CentOS with kernel version of 2.6.18.

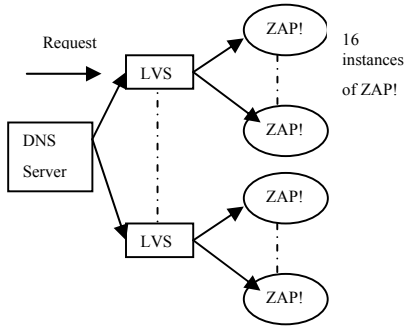


Figure 4 the system deployment diagram

Figure 4 shows the system deployment diagram. We choose httpperf [14] as load generator, LVS [15] with direct route mode is responsible for distributing requests to information service with 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 information service, and each instance of ZAP! is deployed on a virtual machine.

The IS Server adjusts the number of instances of information service according to the criterion of average utilization rate of CPU consumed by instances of information service. We presume the number of current instances of information service is  $n$ . If the average utilization rate of CPUs consumed by instances of information service exceeds 80% in the past 20 seconds, the IS Service will increase one instance. If the average utilization rate of CPUs consumed by instances of information service is lower than 80%  $(n-1)/n$  in the past 20 seconds, the IS Service will decrease one instance until the number of current instances is equal to 1.

We use request *trace of information service* described in section 3.2. Figure 5 shows the varying real resource consumption in two weeks for information service trace, of which the peak resource demand is 64 virtual machines.

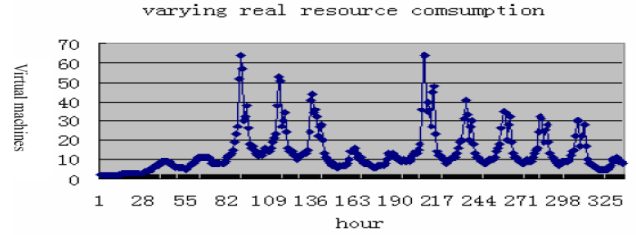


Figure 5 the resource consumption of information service trace in two weeks.

### 3.4. The simulation experiments of consolidating computing loads

We use simulation method to verify the advantage of consolidating different computing loads from different departments on shared cluster system. Figure 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 information service (IS CMS). Comparing with the real Phoenix Cloud system, our simulated system maintains resource provision service, IS (information service) Server, ST (scientific computing) Server and scheduler, while other services are removed or substituted.

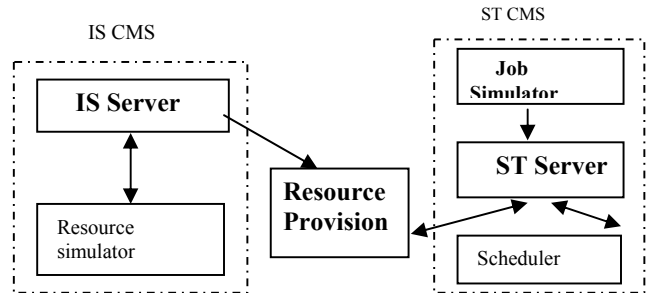


Figure 6 the hybrid experiment system

For IS CMS, a daemon named *resource simulator* will simulate varying resource demand of IS CMS and drive the IS Server to obtain or return resource from and to the resource provision service. We use the real resource consumption of Figure 5 as input to *resource simulator*.

For ST CMS, the scheduler is specified with First-Fit scheduling policy, and a *job simulator* is used to simulate the process of submitting job. 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 scientific computing trace is introduced in section 3.2.

We presume that the software package of information service are pre-deployed on those reallocated nodes, so the time of reallocating nodes from ST Server to IS server is only seconds, includes the time of killing jobs and the time of communicating between IS Server, ST Server and resource provision service.

In our simulation system, for *static configuration (SC for short)*, of which each department of large organization maintains its own cluster system, the minimum scale of cluster system for scientific computing trace introduced in section 3.2 is 144 nodes, because the real SDSC trace is also collected from the same 144 nodes; the minimum scale of cluster system for information service is 64 nodes, because the peak resource demand in Figure 5 is 64 virtual machines. So the *size of initial resource* (in short, *SIR*) allocated to information service and scientific computing for SC is 208. For *dynamic configuration (DC for short)*, we respectively set the *SIR* allocated to information service and scientific computing as 200, 190, 180, 170, 160 and 150. Figure 7 shows the number of completed jobs and average turnaround time per job in term of seconds for scientific computing trace in two weeks when we set different value of SIR.

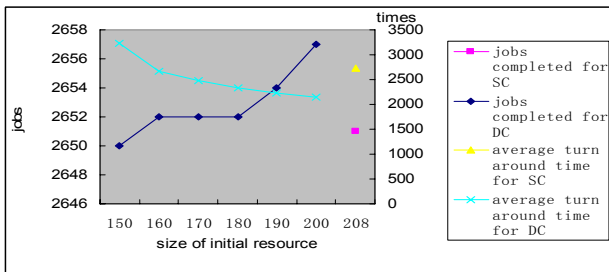


Figure 7. For scientific computing trace, number of jobs completed and average turnaround time per job in two weeks with different setting value of SIR

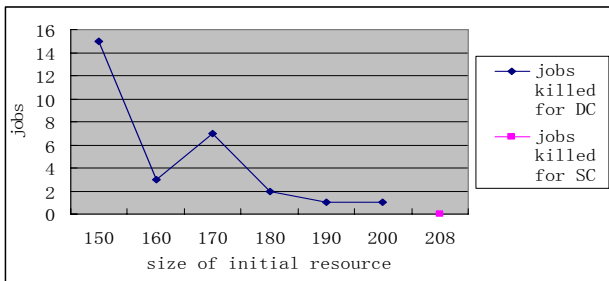


Figure 8. For scientific computing trace, number of killed jobs in two weeks with different value of SIR

For scientific computing trace, 2672 jobs are submitted to ST Server. For DC, when the *cost* of large organization in term of *value of SIR* decreases to 160, only 76.9% of that of static configuration, the *benefit of scientific computing department* in term of *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 average turnaround time per job* is still higher than that of static configuration.

With the *value of SIR* decreases, the number of killed jobs increases in general. Only the exception is the number of

killed jobs at the point when SIR =170, which is higher than that at the point when SIR=160.

For information service, the benefits of service provider and end user are unchanging, since we just use the real resource consumption collected from section 3.3 as input of resource simulator.

## 4. CONCLUSION

Different departments of large organization 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 scientific computing and information service on shared cluster system. Besides, we have proposed a cooperative resource provision and management policy of large organization and its affiliated departments to share the centralized system.

Our experiments show: comparing with the case that each department maintains its dedicated cluster system, the consolidation of scientific computing and information service with cooperative resource provision and management policy can significantly decrease the scale of required cluster system of large organization, at the same time improve the benefit of scientific computing department and its end users while provisioning enough resource to information service with varying load.

## 5. ACKNOWLEDGMENTS

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

## 6. 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, "Océano--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/httpperf/>
- [15] LVS: <http://www.linuxvirtualserver.org/>