One Click to Build An On Demand Virtual Cluster in Cloud Web-based Operating System with Dynamic Loading Prediction Scheduling Algorithm

Chang-Hsing Wu, Yi-Lun Pan, Hsi-En Yu, Hui-Shan Chen, and Ching-Wen Yu
Software Technology Division
National Center for High-Performance Computing
Hsinchu, Taiwan
E-mail：hsing@nchc.narl.org.tw, serenapan@nchc.narl.org.tw, yun@nchc.narl.org.tw, chwhs@nchc.narl.org.tw, cwyu@nchc.narl.org.tw

Abstract—**As virtualization technologies become more prevalent, Cloud users usually encounter the problem of how to build his/her own virtual cluster with a friendly user interface for virtual resource management. To help resolving this problem, an On Demand Virtual Cluster system in Cloud Web-Based OS has been developed by the Pervasive Computing Team at the National Center for High-Performance Computing (NCHC). Through the On Demand Virtual Cluster system, with a click, Cloud users can customize and configure the specified virtual environment. We embedded the On Demand Virtual Cluster system into the Cloud WebOS, an extremely lightweight approach helping users to access virtual computing resources. The Cloud WebOS leverages virtualization techniques and cluster scheduling policy, which is the proposed dynamic loading prediction scheduling algorithm.**

Keywords - Virtualization Techniques; WebOS; Virtual Cluster; Cluster Scheduling Policy.

I. INTRODUCTION

In Cloud computing environment, there are various important issues, including information security, virtual computing resource management, routing, fault tolerance, and so on. Among these issues, the virtual computing resource management has emerged as one of the most important ones in the past few years. Currently, Cloud users have to manually build specified virtual cluster with the commend line mode in order to manage or generate virtual resources. To improve this condition, an On Demand Virtual Cluster system in Cloud WebOS (Web-Based Operating System) platform has been developed by the Pervasive Computing (PerComp) Team at the National Center for High-Performance Computing (NCHC). On this platform, Cloud users can build on demand virtual clusters with one click.

The Cloud WebOS platform provides a new service paradigm [1]. The WebOS infrastructure offers a seamless and unified access to geographical distributed resources connected via Internet, and it can supply most basic operating system services [2]. The proposed Cloud WebOS platform adopts the Asynchronous JavaScript and XML (AJAX) as a base. The major feature of this Cloud WebOS platform embedded with the On Demand Virtual Cluster system is that users can easily customize and configure their virtual environment according to their needs. It also can seek, diagnose, and monitor Cloud computing resources automatically. Meanwhile, the PerComp Team developed several Cloud widgets on the Cloud WebOS platform to control virtual clusters and virtual machines.

An efficient scheduling policy is indispensable, especially for distributed computing and Cloud computing. We designed an efficient scheduling policy, a dynamic loading prediction scheduling (DLPS) algorithm. It predicts loading of computing resources and makes the most adaptive resources allocation. The PerComp Team not only built the Cloud WebOS platform with the eyeOS [3] framework, but also incorporated the mechanism of scheduling algorithm.

In conclusion, the ultimate target of this research is to find a solution for scientists/researchers to painlessly run their jobs on Clouds. This research focuses on the development of friendly user interface, automatically dynamic allocation technique, integrated heterogeneous computing resources, and computing results visualization.

The rest of the paper is organized as follows. Section II presents related works. In Sections III, we proposed the On Demand Virtual Cluster system in Cloud WebOS and the dynamic loading prediction scheduling algorithm (DLPS). In Section IV, Cloud Widgets and experimental results are presented. Finally, the conclusion and future research directions are presented in Section V.

II. RELATED WORKS

A. Existing Web-based Operating System (Web OS) Projects

Recently, a famous WebOS - Chrome OS, developed based on AJAX technique [4]. It can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. The developments of Cloud WebOS platform via AJAX technique become practicable. However, Chrome OS does not provide on-demand applications and computing services to users in Clouds.

A Web-based Operating System (WebOS) project started at the University of California, Berkeley in 1996 as part of Network of Workstations [4]. So far, there are several typical commercial projects of WebOS, such as FlyakiteOSX [6], Glide OS [7], XIN [8], and so on. All of these systems are online OS with Ajax and PHP techniques. However, these projects are not open source and lack of the management of distributed computing resources. To meet the demand of distributed computing resource management, the Cloud WebOS platform is developed. This development follows the spirit of open source, open standard, and GNU/GPL license.

B. Virtualization Technologies

Our research enhances the efficiency of job scheduling and retains the execution of parallel computing jobs via virtual technique. To implement virtualization technology, an additional software layer, called a virtual machine monitor or a hypervisor, has to be inserted between the existing operating system and hardware to manage the resources and virtual machines. The characteristics of virtualization technology are described as the following:

- Utilization – better utilization means various services run on one physical machine with multiple virtual machines (VMs);
- Isolation - better isolation means a VM can halt and catch fire without affecting the real host or other running VMs;
- Flexibility - the ability of the virtualization technologies to run platforms and operating systems that are different from the host, good flexibility means more choices for VM platforms and the ability to run VMs with minimal modification;
- Manageability - availability of tools and APIs for starting, stopping and moving VMs.

Generally, modern hypervisor implementations are divided into two categories, including Host-based and Bare-metal approaches. The host-based approach uses modified operating systems to provide virtual machine monitoring, such as Linux-VServer [9], Solaris Zones [10], and Kernel-based Virtual Machine (KVM) [11]. On the other hand, the bare-metal approach employs small-dedicated hypervisors to directly run on physical machines. The VMware ESX server [12], and the XenServer [13] are the famous examples of the bare-metal approach.

With success of the virtual technologies, we integrate virtualization technology – KVM and WebOS. This research comes up with a new and lightweight approach to access virtual computing services via the On Demand Virtual Cluster system.

III. PROPOSED ON DEMAND VIRTUAL CLUSTER SYSTEM IN CLOUD WEBOS

A. Research Objective

The key idea of Cloud Computing lies in its component-based nature, which are reusability, substitutability and user friendly. By integrating virtualization technologies and WebOS, we provided a web environment to access Cloud services via Cloud Widgets in the Cloud WebOS. This progress helps to lower the barrier for using Clouds. In order to develop an autonomic virtual computing resources management system based on decentralized resource discovery architecture, we implemented the On Demand Virtual Cluster system in Cloud WebOS. At the same time, an efficient scheduling policy is also important. Therefore, the dynamic loading prediction scheduling (DLPS) algorithm is used for the scheduling of virtual cluster and physical cluster. It predicts loading of computing resources and makes the most adaptive resources allocation.

As the Figure 1, it shows a high level overview of the Cloud WebOS. In the middle of this figure, when the Cloud WebOS receives a Cloud job request from the users via the web browser, and then the job will be sent to the fittest virtual cluster in the backend to process via the On Demand Virtual Cluster system. The system will help users to generate the fittest virtual cluster and choose/allocate the most adaptive physical resources with a graphical interface.

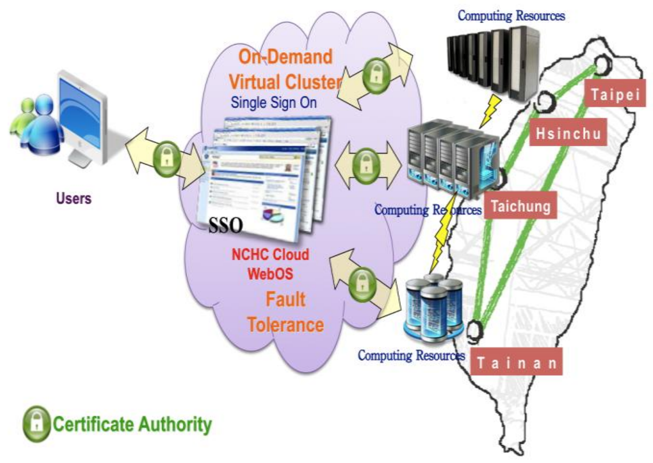


Figure 1. The Overview of Cloud WebOS Platform

B. Implementation and System Architecture

In this project, we combine the WebOS platform with Cloud computing resources to offer users a friendlier Cloud environment. The system architecture of the On Demand Virtual Cluster system in Cloud WebOS is sketched in the Figure 2. In the Cloud WebOS, upon receiving a Cloud job request from the end users via Web Browser, the system acquires Cloud Services via Cloud Widgets, which in turn connect the Image Creator Widget, Virtual Machine (VM) Creator Widget, VM Monitor Widget, and VM Control Widget, within On Demand Virtual Cluster system. Each of Cloud Widgets is described in Section IV. The system helps selecting the most adaptive computing resources to create virtual clusters automatically based on the demands from the end users. These Widgets of On Demand Virtual Cluster system in Cloud WebOS also drive the Cloud middleware to operate physical computing resources and storages.

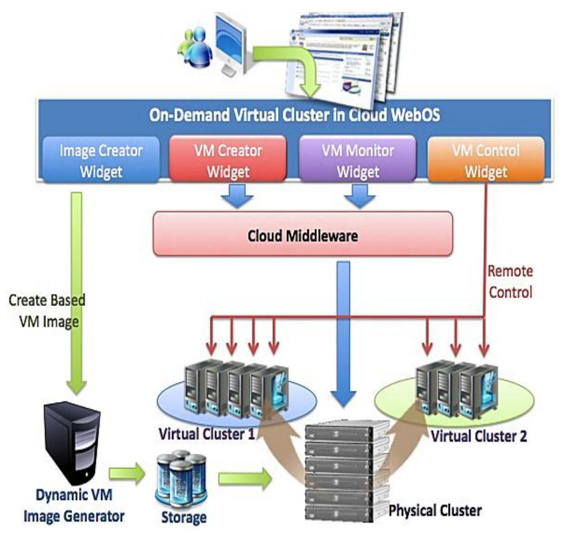


Figure 2. The System Architecture of On Demand Virtual Cluster System in Cloud WebOS

Meanwhile, as the Figure 3 is shown, the end users connect the Application Pool to get the software services, such as information security and Bio simulation via Cloud WebOS easily. After connecting Application Pool, the Cloud WebOS also can integrate the public service provider, such as Amazon EC2 [14] and so on. Upon receiving Cloud job request via Cloud WebOS, the On Demand Virtual Cluster system makes communication with Cloud Middlewares, which are Data Broker, Monitoring & Reporting, and Dynamic Provisioning. The Data Broker collects data from the distributed physical sensors. The Monitoring & Reporting takes responsible for monitoring the status of physical machines and virtual machines. Finally, the Dynamic Provisioning provides the capability of resource allocation automatically, and the feature of DLPS algorithm is activated at the same time. The DLPS algorithm can improve the performance of the dynamic scheduling over conventional scheduling policies.

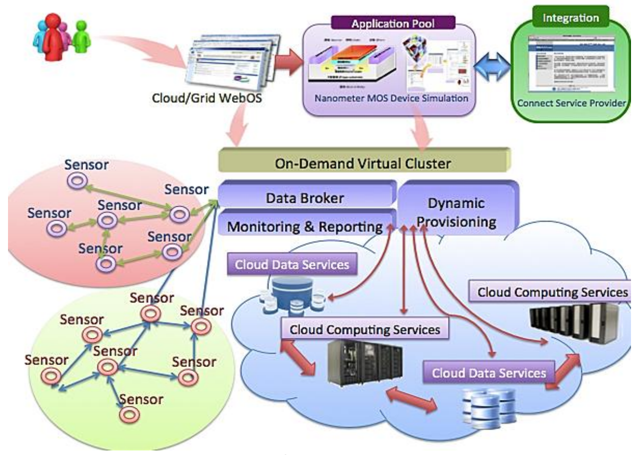


Figure 3. The Application of On Demand Virtual Cluster System

With On Demand Virtual Cluster system in Cloud WebOS, users can create a dynamic HPC cluster consisting of VMs. The scale of each virtual cluster can be determined by user's criteria. When user needs virtual cluster no longer, the virtual cluster can be destroyed. Then the computing resources are released. The whole operation can be manipulated via web browser, because we use XML-RPC based Application Programming Interface (API). Moreover, there are two middlewares embedded into the proposed Cloud WebOS, as the following shown: 1) *Integration with OpenNebula* – OpenNebula is used as central cloud management [15]. It is responsible for finding available computing resources, creating VMs based on a selected image, and deploying the image into the physical computing resources. It also manages unique MAC address, IP address, and virtual network (vNet) ID. Therefore, each user's cluster lives on its own vNet, in order to isolate the various virtual clusters. 2) *Batch System with Torque* – It is used for the scheduling of virtual cluster and physical cluster. The DLPS algorithm is embedded with this resource manager, Torque [16] in the Figure 4. This development not only makes users submit job as usual via PBS_SERVER, but also makes resource manager have additional capabilities of loading prediction and job scheduling with virtual technology. The detail explanation is in Section III-C.

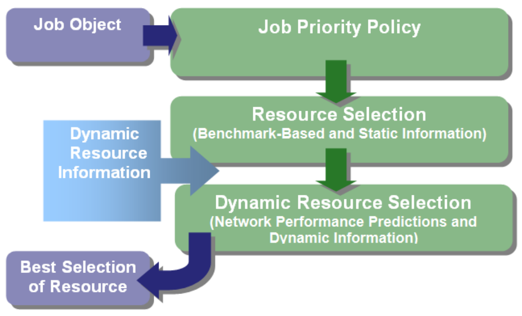

Job Object
Job Priority Policy
Resource Selection
(Benchmark-Based and Static Information)
Dynamic Resource Information
Dynamic Resource Selection
(Network Performance Predictions and Dynamic Information)
Best Selection of Resource

Figure 4. The Scheduling Policy - DLPS in Cloud WebOS

C. *Scheduling Policy - Dynamic Loading Prediction Scheduling (DLPS) Algorithm in Cloud WebOS*

The presented scheduling policy is called Dynamic Loading Prediction Scheduling (DLPS) algorithm. It can schedule the computing resources in Clouds and even multiple clusters. The objective function of DLPS is achieving the minimized makespan (defined in Definition 1). Thus, we designed the following equation to describe the objective function, as in (1).

$$M^* = Min[\max(d_k) - \min(s_k)] \quad (1)$$

The above equation (1) is defined in ***Definition 2***.

Definition 1: The completion time is defined as the time from the job being assigned to one machine until the time the job is finished. The complete time is also called makespan time.

Definition 2: M^* means the minimized makespan. In order to predict precisely, there are two parameters - d_k and S_k. d_k is the maximum *job ending time* of the *k*th job, which means the end time of job completed. And S_k is the minimum *job submitting time* of the *k*th job, which means the time stamp when users submit the *k*th job.

The each step of DLPS algorithm is descried as the following pseudo code in Figure 5.

```
Input: Cloud WebOS form-specification(s) of jobs requirements and users' criteria
Objective Function: $M^* = Min[\max(d_k) - \min(s_k)]$
 It means the time of total deliver or response time is minimum.
Output: The most appropriate resource candidate and available resources list

1: Process users' criteria and job requirements from Cloud
   WebOS, High-Performance Linpack Benchmark, data
   set, execution programs, and Queue type, etc.
2: Make communicate with each Cloud resource for
   getting the static and dynamic resource information.
3: For each job {
     i.  Store the features and status of each cluster and
         Cloud resource into Database
     ii. Call the adaptive resource allocation function to
         compare the free nodes with requirement nodes,
         and then filter out unsuitable resources.

4:     Y = (Tem_value); initial value
5:     if (free nodes $\geq$ requirement nodes){
6:           With Definition 3, it calculates higher weight;}
7:     else (free nodes < requirement nodes){
8:                 call dynamic loading prediction function
                   with predict EstBacklog (Definition 4)
                   and Job Expansion Factor (Definition 5)
                   methods to calculate final weight
                   through Definition 6;}
9: Calculate the $Min(Y) = M^*$;
10: }
```

Figure. 5 The Pseudo of DLPS

The logical flow chart of the DLPS is illustrated as in the Figure 6. First, the DLPS retrieves the information of computing resources from the local queuing system, and then filters out unsuitable resources with the adaptive resource allocation function. After using adaptive resource allocation function, DLPS compares free nodes with required nodes. If current free nodes are enough, DLPS will give higher weight (defined in Definition 3). Otherwise, the following step enters the dynamic loading prediction function with EstBacklog and minimum Job Expansion Factor (defined in Definition 4 and Definition 5) methods to predict which computing resources respond and execute job quickly, and then calculate the weight (defined in Definition 6). Finally, the DLPS ranks all available resources and selects the most appropriate resources to dispatch job and generate virtual machines.

Definition 3: When free nodes fulfill required nodes, the weight of *k*th job is designed as following:

$$Weight_k = (R_{nodes} / f_{nodes}) \times M_{capability}$$

Where R_{nodes} means the number of required nodes, f_{nodes} means the number of free nodes, and $M_{capability}$ means the capability of each computing resources. The capability is based on static information, such as High-Performance Linpack Benchmark results, and HPC Challenge Benchmark.

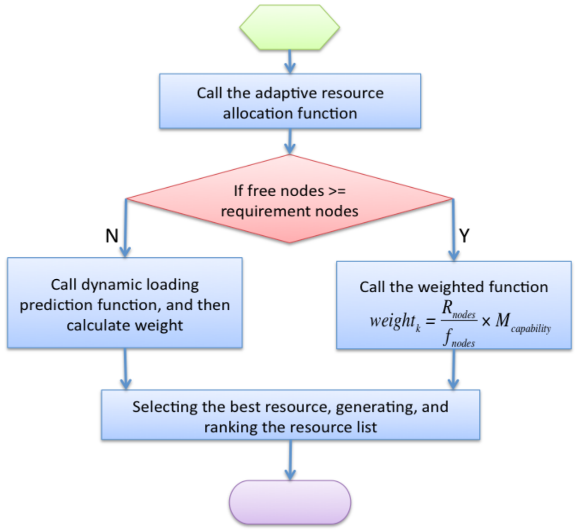

Call the adaptive resource allocation function
If free nodes >= requirement nodes
N
Y
Call dynamic loading prediction function, and then calculate weight
Call the weighted function
$weight_k = \frac{R_{nodes}}{f_{nodes}} \times M_{capability}$
Selecting the best resource, generating, and ranking the resource list

Figure. 6 The Logical Flow Chart of DLPS

Definition 4: The *EstBacklog* means estimated backlog of queued work in hours. The general *EstBacklog* form is shown as the equation (2):

$$EBL = \left(\frac{QueuePS \times CPUAccuracy}{TotalJobsCompleted}\right) \times \left(\frac{TotalProcHours \times 3600 \times AvailableProcHours}{DedicatedProcHours}\right) \quad (2)$$

QueuePS is the idle time of queued jobs. *CPUAccuracy* is the actual run time of job. *TotalJobsCompleted* is the number of jobs completed. The *Toatl ProcHours* is the total number of proc-hours required to complete running jobs. The *Available ProcHours* is the total proc-hours available to the scheduler. The last variable, *Dedicated ProcHours*, is the total proc-hours made available to jobs.

Some of above values are from the system historical statistic values of queuing system loading and the others are from real-time queuing situation. The output is divided into two categories, running and completed. The Running statistics include information about the currently running jobs. The completed statistics are compiled using historical information from both running and completed jobs. Therefore, the *EBL* can forecast the backlog of each computing site with above information.

Definition 5: The *job expansion factor* subcomponent has an effect similar to the queue time factor but favors shorter jobs based on the requested wallclock run time. The job expansion factor metric is calculated by the information from local queuing system as described in the equation (3):

$$JEF = \frac{QueuedTime + RunTime}{WallClockLimit} \qquad (3)$$

Definition 6: After getting *EstBacklog* and *job expansion factor*, the metric is calculated by the following equation (4):

$$Weight_k = \lambda \times \frac{JEF_k}{TotalJEF} + (1-\lambda) \times \frac{EBL_k}{TotalEBL} \qquad (4)$$

The $Weight_k$ means the weight of the *k*th job. $\frac{JEF_k}{TotalJEF}$ means the *JEF* of *k*th job divided by the Total *JEF*. $\frac{EBL_k}{TotalEBL}$ means the *EBL* of *k*th job divided by the Total *EBL*. Where λ is the modulated parameter of system, which can be obtained from numerous trials. The EstBacklog can be respected the dynamic situation of queuing system generally. Therefore, it always uses the higher λ value. Consequently, we can use above parameters to calculate the minimum time of total deliver and response time.

IV. Cloud Widgets/Experimental Results

A. *Cloud Widgets*

In addition to the basic Widgets, more advanced Cloud Computing Widgets are attempted as well. We have developed many Cloud Widgets with friendly graphical user interface in WebOS. The kernel of the On Demand Virtual Cluster system architecture consists of four Widgets, including Image Creator Widget, VM Creator Widget, VM Monitor Widget, and VM Control Widget. Users without much learning effort can easily manage all of these widgets.

The Image Creator Widget, in the Figure 7, is to generate the customized base image and the on-demand/specified applications from the end users' requirements. This Widget provides a complete and integrated HPC software stack that consists of operating system, management tools, resource monitor, and even commercial package, such as the Matlab. The VM Creator Widget - with the profile of virtual cluster demanded by the user provided, it will generate a specification, shown in the Figure 8, which in turn is parsed by the VM Creator engine to create specified virtual cluster on the physical computing resources. Thus, after completing the profile of virtual cluster, with a click, Cloud users can customize and configure the specified virtual environment in real time.

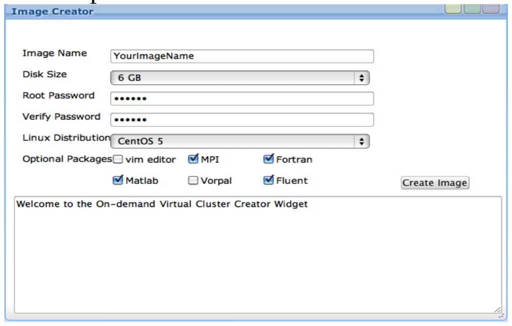


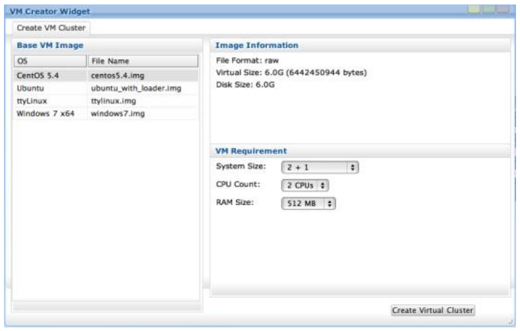
Image Creator

Image Name	YourImageName
Disk Size	6 GB
Root Password	••••••
Verify Password	••••••
Linux Distribution	CentOS 5
Optional Packages	☐ vim editor ☑ MPI ☑ Fortran ☑ Matlab ☐ Vorpal ☑ Fluent

Create Image

Welcome to the On-demand Virtual Cluster Creator Widget

Figure 7. Image Creator Widget



VM Creator Widget

Create VM Cluster

Base VM Image

OS	File Name
CentOS 5.4	centos5.4.img
Ubuntu	ubuntu_with_loader.img
ttyLinux	ttylinux.img
Windows 7 x64	windows7.img

Image Information

File Format: raw
Virtual Size: 6.0G (6442450944 bytes)
Disk Size: 6.0G

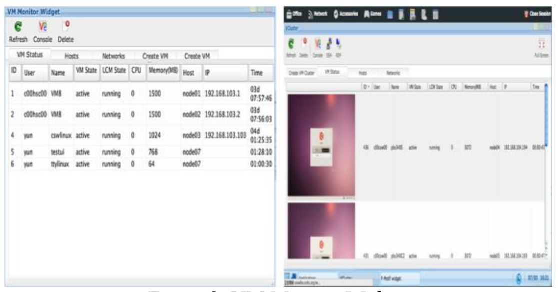
VM Requirement

System Size:	2 + 1
CPU Count:	2 CPUs
RAM Size:	512 MB

Create Virtual Cluster

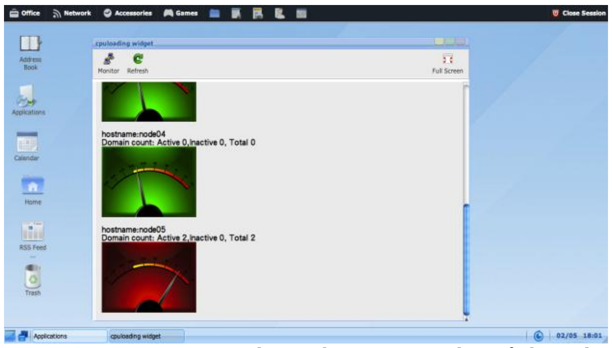
Figure 8. VM Creator Widget

In the Figure 9, the main task of the VM Monitor Widget is to monitor the all the status of virtual machines, Networks, and the physical hardware. The VM Monitor also can show the current loading of physical machines, in the Figure 10. The VM Control Widget provides users to access, ssh, or operate virtual machines through Cloud visualizer, as shown in the Figure 11 and Figure 12. We used above Cloud Widgets to implement the following two customized applications for biological simulation and information security simulation. The F-motif Simulation Widget provides specialized Cloud services to search and analyze the sequence of gene in real time, in Figure 13. The other customized Cloud Widget about information security is called ICAS (IDS-log Cloud Analysis System) Widget. As long as user selects the ICAS base image, the Hadoop DB and virtual cluster are constructed automatically, and then users can analyze the IDS-log as the Figure 14 shown.



ID	User	Name	VM State	LCM State	CPU	Memory(MB)	Host	IP	Time
1	c00hsc00	VM8	active	running	0	1500	node01	192.168.103.1	03d 07:57:46
2	c00hsc00	VM8	active	running	0	1500	node02	192.168.103.2	03d 07:56:03
4	yun	cswlinux	active	running	0	1024	node03	192.168.103.103	04d 01:25:35
5	yun	testui	active	running	0	768	node07		01:28:10
6	yun	ttylinux	active	running	0	64	node07		01:00:30

Figure 9. VM Monitor Widget

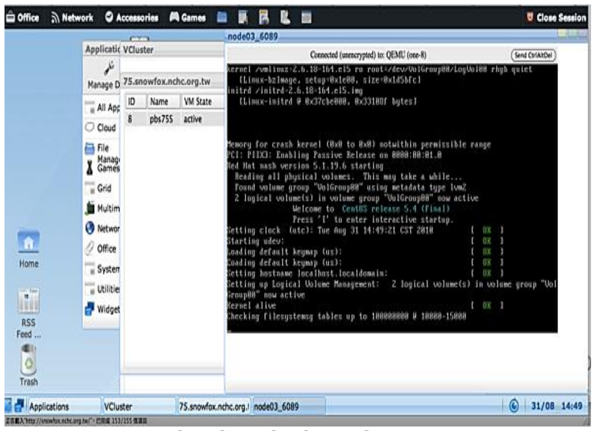


cpuloading widget

hostname:node04
Domain count: Active 0,Inactive 0, Total 0

hostname:node05
Domain count: Active 2,Inactive 0, Total 2

Figure 10. VM Monitor Widget – The Current Loading of Physical Machines



Connected (unencrypted) to: QEMU (one-8)

```
kernel /vmlinuz-2.6.18-164.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
   [Linux-bzImage, setup=0x1e00, size=0x1d5bfc]
initrd /initrd-2.6.18-164.el5.img
   [Linux-initrd @ 0x37cbe000, 0x33100f bytes]


Memory for crash kernel (0x0 to 0x0) notwithin permissible range
PCI: PIIX3: Enabling Passive Release on 0000:00:01.0
Red Hat nash version 5.1.19.6 starting
  Reading all physical volumes.  This may take a while...
  Found volume group "VolGroup00" using metadata type lvm2
  2 logical volume(s) in volume group "VolGroup00" now active
                Welcome to  CentOS release 5.4 (Final)
                Press 'I' to enter interactive startup.
Setting clock  (utc): Tue Aug 31 14:49:21 CST 2010         [  OK  ]
Starting udev:                                             [  OK  ]
Loading default keymap (us):                               [  OK  ]
Loading default keymap (us):                               [  OK  ]
Setting hostname localhost.localdomain:                    [  OK  ]
Setting up Logical Volume Management:   2 logical volume(s) in volume group "Vol
Group00" now active
Kernel alive                                               [  OK  ]
Checking filesystemsg tables up to 100000000 @ 10000-15000
```

Figure 11. VM Control Widget Cloud Visualizer – Linux Booting Status

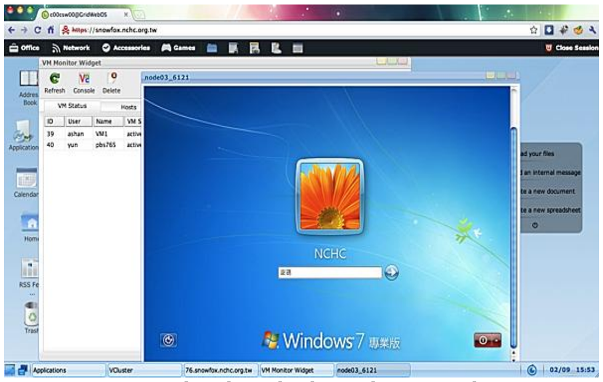


Figure12. VM Control Widget Cloud Visualizer – Windows 7 Booting Status

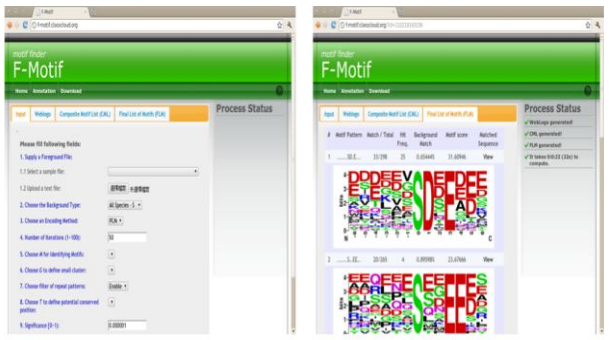
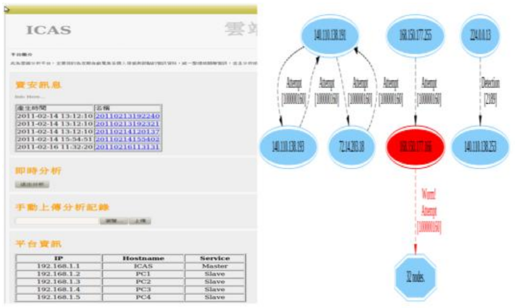


Figure 13. F-motif Widget



ICAS

資安訊息

產生時間	名稱
2011-02-14 13:12:10	20110213192240
2011-02-14 13:12:10	20110213192321
2011-02-14 13:12:10	20110214120137
2011-02-14 15:54:51	20110214155402
2011-02-16 11:32:20	20110216113131

即時分析

手動上傳分析記錄

平台資訊

IP	Hostname	Service
192.168.1.1	ICAS	Master
192.168.1.2	PC1	Slave
192.168.1.3	PC2	Slave
192.168.1.4	PC3	Slave
192.168.1.5	PC4	Slave

Figure 14. ICAS Widget

B. *Experimental Results*

The preliminaries of experiment are needed to set up, including the multi-sites physical computing environment, the virtual machine – KVM, Network Speed Test [17], and Disk I/O test tool - bonnie++ [18]. As shown in Table I, we list the summary environment characteristics of NCHC computing resources.

TABLE I. SUMMARY ENVIRONMENT CHARACTERISTICS OF NCHC COMPUTING RESOURCES

Resource	CPU Model	Memory (GB)	CPU Speed (MHz)	#Cores	Nodes	Job Manager
Snowfox	Intel(R)Xeon(R) CPU 2.5GHz, E5420	16	2500	112	14	Torque
Capri	Intel(R) Xeon(R) CPU E5620 , 2.40GHz	32	2400	232	29	Torque

There are three scenarios, including the performance of Network I/O, Disk I/O, and the DLPS algorithm. Moreover, in order to improve the performance, we use the Virtio driver [19] in the virtual machines. Virtio driver provides paravirtualized functions for network virtualization and disk I/O virtualization. In Figure 15, we found the Network speed is tackled about 166 Mb/s without Virtio, because the I/O bottleneck is between virtual machine and hypervisor. Therefore, the Virtio is activated in the On Demand Virtual Cluster system. The performance of Network I/O is nearly the same with native machine. In the performance of Disk I/O scenario, we compared with Virtio and without Virtio. With Virtio, it can be improved write performance about 120% and read performance about 20%, as the following Figure 16 is shown.

The performance of DLPS algorithm is compared with several algorithms, such as Round Robin, Short-Job-First (SJF), Big-Job-First (BJF), and First-Come-First-Serve (FCFS). We submitted testing jobs, which were generated randomly with the synthetic models as the Figure 17 is shown. The vertical axis is the value of min makespan (seconds), and the horizontal axis is the number of jobs. The makespan of DLPS algorithm is notably less than other algorithms, especially when a huge number of jobs are submitted. Therefore, the objective function of DLPS approaches the minimized makespan. The dynamic loading prediction characteristic of presented system is proved to be better in this experiment.

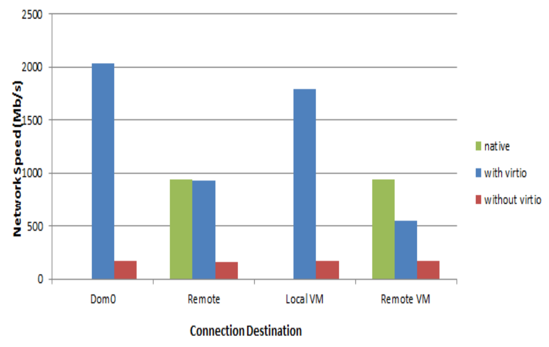


Figure 15. The Performance of Network I/O

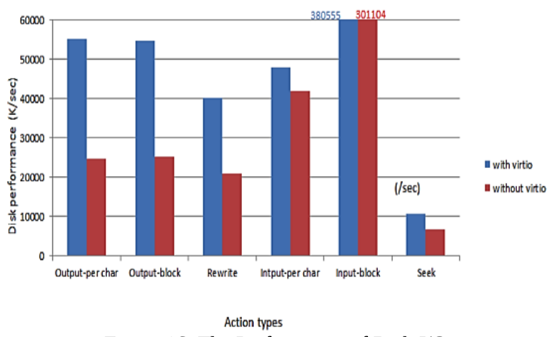


Figure 16. The Performance of Disk I/O

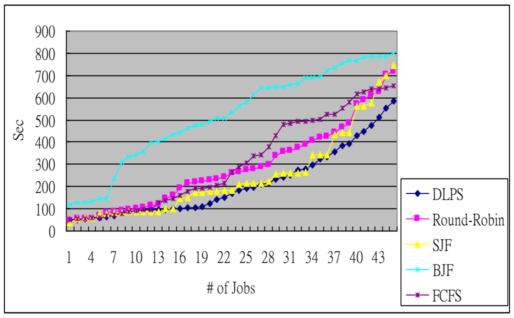


Figure 17. Compare Makespan of DLPS with Other Algorithms

When a small number of jobs are submitted, the efficiency of DLPS may be worse than other algorithms, especially for SJF and FCFS. This situation is reasonable, because small jobs are easy consumed by SJF and FCFS. When the number of jobs is increasing, the developed DLPS is absolutely better than SJF and FCFS, because the notable drawback of SJF and FCFS happens, which the large numbers of jobs are queued inefficiently in the local scheduler of cluster. Comprehensively the above efficiency figure, the best efficiency of DLPS occurs at full usage of each cluster.

V. CONCLUSION AND FUTURE WORK

The research – On-Demand Virtual Cluster system in Cloud WebOS, provides Cloud users with an interface that is user-friendly, more straightforward, and more efficiency. The On Demand Virtual Cluster System in Cloud WebOS not only helps user to build virtual cluster easily and automatically, but also provides different varieties of computing environment such as Linux, Win7, and so on. The Virtio driver is activated in the On Demand Virtual Cluster system. Thus, the performance of Network I/O is nearly the same with native machine. The performance of Disk I/O can be improved write performance about 120% and read performance about 20%.

Furthermore, the research leverages virtualization techniques combined with cluster queuing system and job scheduling mechanism. According to the pervious experiment, the DLPS has better efficiency than other scheduling algorithms; especially the huge numbers of job are submitted into the computing cluster. Finally, we obtain an important property that the algorithm is appropriate to deal with large amount of jobs in real Clouds or distributed environment.

VI. REFERENCES

[1] W. Kim, “Cloud computing: Today and Tomorrow,” Journal of Object Technology, 8, 2009.

[2] G. Gu and X. Lu, “Simple Web OS System Based on Ext Framework and Cloud Computing,“ International Forum on Information Technology and Applications, pp. 448-450, IEEE, 2010.

[3] http://www.eyeos.org/, [accessed: July, 2011].

[4] http://www.chromium.org/chromium-os/, [accessed: July, 2011].

[5] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A, Patterson, “Searching for the Sorting Record: Experiences in Tuning NOW-Sort,” *The 1998 Symposium on Parallel and Distributed Tools (SPDT '98)*, Welches, Oregon, August 3-4, 1998.

[6] http://osx.portraitofakite.com/logon.htm, [accessed: July, 2011].

[7] http://www.glidedigital.com/, [accessed: July, 2011].

[8] http://www.xindesk.com/, [accessed: July, 2011].

[9] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors, " Proceedings of ACM SIGOPS/Eurosys European Conf. on Computer Systems, pp. 275-287, Mar. 2007.

[10] D. Price and A. Tucker, "Solaris Zones: Operating Systems Support for Consolidating Commercial Workloads, " Proceedings of 18th Large Installation System Administration Conf., pp. 241-254, Nov. 2004.

[11] B. Zhang, X. Wang, R. Lai, Y. Liang, Z. Wang, Y. Luo, and X. Li, "Evaluating and Optimizing I/O Virtualization in Kernel-based Virtual Machine (KVM)," International Conference on Network and Parallel Computing, pp. 220-231, Zhengzhou, China, September 13-15, 2010.

[12] John Paul, "VMWare ESX Server Workload Analysis: How to Determine Good Candidates for Virtualization," 33rd International Computer Measurement Group Conference, pp. 483-484, San Diego, CA, USA, December 2-7, 2007.

[13] X. Ge, H. Jin; S. Wu, X. Shi, W. Gao, "A method of multi-VM automatic network configuration," Intelligent Computing and Intelligent Systems, pp. 309-313, 2009.

[14] http://aws.amazon.com/ec2, [accessed: July, 2011].

[15] Ignacio M. Llorente and Rubén S. Montero, “OpenNebula: A Cloud Management Tool,” Internet Computing, IEEE, pp. 11-14, March-April 2011.

[16] http://www.clusterresources.com/products/torque-resource-manager .php, [accessed: July, 2011].

[17] http://vmstudy.blogspot.com/2010_04_01_archive.html, [accessed: July, 2011].

[18] http://www.coker.com.au/bonnie++/, [accessed: July, 2011].

[19] http://www.linux-kvm.org/page/Virtio, [accessed: July, 2011].