Vega LingCloud: A Resource Single Leasing Point System to Support Heterogeneous Application Modes on Shared Infrastructure

Xiaoyi Lu[1, 2], Jian Lin[1,2], Li Zha[1], and Zhiwei Xu [1]
[1] (Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
[2] (Graduate University of Chinese Academy of Sciences, Beijing 100049)
e-mail: {luxiaoyi, linjian}@software.ict.ac.cn, {char, zxu}@ict.ac.cn

***Abstract*—In large organizations or IDCs, different departments always occupy and maintain dedicated resources to satisfy their or their customers' heterogeneous application loads. This situation easily makes the infrastructure management a repeated and inefficient work. Even worse, it is difficult to share the resources owned by different departments even when they are idle, because the application modes on these resources are quite different. This paper introduces a live system, Vega LingCloud, which provides a Resource Single Leasing Point System for consolidated renting physical and virtual machines to support heterogeneous application modes on shared infrastructure. Furthermore, we present the asset-leasing model and the architecture of Vega LingCloud. According to the evaluation, Vega LingCloud is better than other systems like OpenNebula and Enomaly ECP in the aspects of uniformity, flexibility, security, usability, and efficiency. The experimental result of management overhead shows that the deployment speed of virtual machine in Vega LingCloud is 4.1 times of that in the OpenNebula and VIDA hybrid system for deploying 64 virtual machines concurrently. From a representative micro-cloud example in a research group, we show that the consolidated way of leasing physical and virtual machine in Vega LingCloud is approbatory. Up to now, Vega LingCloud has been deployed in real world environments, which include a private cloud of large organization in Beijing and a public cloud in the Dongguan IDC of China. The resource scale of the Dongguan cloud reaches about 504 cores, 625 GB memory, and 156 TB storage. The number of supported real applications in Beijing and Dongguan clouds has exceeded 35, and their modes involve high performance computing, large scale data processing, virtual machine leasing, data storage, and so on.**

Keywords-Cloud Computing; Resource Single Leasing Point System;Vega LingCloud; Molva; Lota; Asset-Leasing Model; Partition; Virtual Cluster; Virtual Network; Virtual Appliance

I. INTRODUCTION

When building a private cloud for a large organization or a public cloud for an IDC, we observe that different departments always occupy and maintain dedicated resources to satisfy their or their customers' heterogeneous application loads. Taking Dongguan IDC in China as an example, it has three representative applications managed by three departments: one is a public virtual machine leasing system, another one is video processing system (a batch job processing system, belonging to high performance computing mode), and the last one is user behavior analysis system (large scale data processing mode). All of them are "silo" systems, whose characteristics are as following: the applications are built directly on the physical infrastructure; independently occupy a whole set of hardware and software environment. Therefore, each department should manage and maintain the resources by themselves. It will cause some common management activities overlapped, and make maintaining work inefficient. To make matters worse, when the resources of different silo systems are idle, they also cannot be shared easily by other systems when meeting peak requirements, because the application and usage modes on these resources are quite different. However, if we simply throw these systems' maintenance work to a central management department, then the global administrators should master too much knowledge. In this example, the knowledge includes installation, configuration, and operation of virtualization hypervisors (e.g. Xen [1] /KVM [2] /VMWare [3]), batch job management systems (e.g. OpenPBS [4] / Torque [5]), and large-scale data processing system (e.g. Hadoop [6] / Sector & Sphere [7]). The Dongguan IDC scenario is not a special case. When we build a private cloud for a large organization in Beijing, we find they also have the demand of adopting a single management system to consolidate resources separated in different departments on shared infrastructure for supporting heterogeneous application modes.

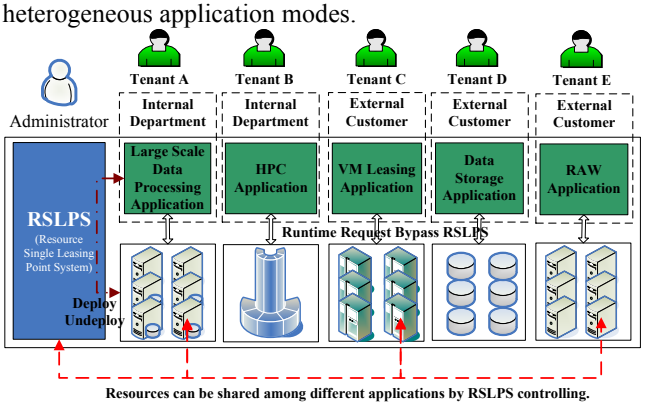

Figure 1. Features of Resource Single Leasing Point System.

Scenarios above can be abstracted to Figure 1, which demonstrates observed features of resource management requirements in cloud computing as follows:

- **Heterogeneous Application Mode Supporting**. This paper mainly focuses on five kinds of application modes, including Large Scale Data Processing Mode, High

978-0-7695-4428-1/11 $26.00 © 2011 IEEE
DOI 10.1109/ISPA.2011.58

Performance Computing Mode, VM Leasing Mode, Data Storage Mode, and RAW Application Mode. The RAW application mode means the provisioned resources are maintained by applications directly.

- **Single Point Controlling**. Due to supporting of heterogeneous application modes, it is really insufficient for us to only use virtualization technology to manage infrastructure because of the performance limitation of virtualization. So this paper formulates the scope of single point controlling as three points: a) single logic view and it is independent with resource provision way (physical or virtual machines); b) global management and operation through uniform UI interfaces, including portal, utility tools, and APIs; c) the operation scope involving both of the infrastructure and applications. Therefore, we should not only manage machines, but also encapsulate and deploy/un-deploy applications. By single point controlling, the administrator can manage resources and applications easily.
- **Multi-tenant Oriented Resource Leasing**. In cloud computing, resources are often used in the "pay-as-you-go" manner. One of the biggest advantages of this usage mode is resource leasing but not exclusive occupation. It can dramatically increase the possibility of resource sharing, because when the leasing relationship is expired, the system administrator can share the idle resources to other busy applications as soon as possible. This feature requires the cloud management system over shared physical infrastructure to lease desired running environments to multi-tenants, who may be from internal departments or external customers. So the isolations of account, performance, and failure are important, and we can see that the cloud management system is required to support both of sharing and isolation simultaneously.
- **Noninterference Application Runtime Management**. When we built the public cloud for Dongguan IDC, the manager of IDC repeatedly emphasized that the cloud management system should not or as little as possible interfere with the running applications on account of the application performance and reliability. The cloud management system should only guard and monitor applications independently.

This paper defines **Resource Single Leasing Point System** (**RSLPS**) to describe a cloud management system, which contains all the features above, and introduces a live system, Vega LingCloud, which provides a RSLPS for consolidated renting physical and virtual machines to support heterogeneous application modes on shared infrastructure.

Vega LingCloud's research scope covers end-user utilities, SaaS, PaaS, and IaaS. A utility tool called Lota, supports a variety of access protocols, including SSH/SFTP, VNC, and RDP. Lota provides a global personal view of the resources with single sign-on (SSO) feature. In the SaaS layer, there are two demonstration applications. One is the WebHPC application, which supports the high performance computing activities on the web, and the other is the UBA, which aims to analyze the user behavior characteristics of Internet access. Both two applications are used in SaaS manner and are supported by the platforms in the PaaS layer. Vega LingCoud provides two platforms, HPCP and DCP. The HPCP platform aims at services for the high performance computing in the cloud environment, while the DCP platform provides services for the large scale data processing. The two platforms obtain features provided by IaaS, such as the elastic expansion of computing power, resource reservation, etc. This paper focuses on the work in IaaS: Molva, which is an example of RSLPS and an infrastructure management system providing a huge virtualization resource pool with the abilities of resource adjustment elastically, provisioning on demand for the upper platforms, applications or tenants. Moreover, the resource pool offers not only virtual clusters composed of virtual machines, but also the ones constructed by physical machines, even the mixed clusters. The clusters provided to the upper platforms or applications can be automatically deployed with required software environments. These common environments include clusters composed of any number of nodes with Hadoop, Torque, Xen, Wiki, Blog, ERP, and so on. Users can use these platforms or applications directly on Molva. The goals of Vega LingCloud can be summarized as: building a bottom-up cloud computing system and researching on the basic model, architecture and key technologies of cloud computing system.

The rest of this paper is organized as follows. In section II, we present the asset-leasing model in Vega LingCloud. Section III describes the architecture of Vega LingCloud. Section IV offers the evaluation about Vega LingCloud. We list some related work in section V and section VI concludes the paper.

II. THE ASSET-LEASING MODEL OF VEGA LINGCLOUD

For building a uniform style of the resource single leasing point system, we firstly analyze the asset-leasing model in the cloud computing for the purpose of organizing system design ideas, and implement a fundamental engine.

A. Asset-Leasing Model

The asset-leasing model is illustrated in Figure 2. The external heterogeneous resources are abstracted as uniform assets in the system. Tenants accomplish the asset leasing through the lease management engine.

Definition 1. Tenant is the representative of the subject identity in the leasing relationship. Each tenant has a global unique identifier, and a tenant is denoted as Tenant = <uid, credential, attributes>.

a) ***Uid,*** the unique identifier of a tenant in global.

b) ***Credential,*** the credential of a tenant' identity.

c) ***Attributes,*** properties of a tenant.

A user registers in the system, obtains a globally unique identifier, becomes a certificated and identified tenant, and can lease the system's assets.

Definition 2. Asset represents the object identity in the leasing relationship and provides specific functions. Asset can be denoted as Asset = <aid, alid, attributes, operations>.

a) ***Aid,*** the unique identifier of an asset in global.

b) ***Alid,*** the unique identifier of an asset leaser, which is also a kind of asset and is used to identify the leaser of the

asset. A typical example is that physical machines are built as internal assets in the system, on which virtual machines as another kind of asset are deployed. If the Aid equals to the Alid, it means the asset is the boundary of the system and the root of the asset nested chain.

c) ***Attributes****,* properties of an asset.

d) ***Operations***, interfaces to operate assets are called operations. Resources outside the system are heterogeneous, and typically include CPU, memory, disk, network, physical machine, virtual machine, software, etc. Based on the unified abstraction, AssetController, which is like driver mechanism in the operating system, a set of uniform access and management operations are provided, including creation, destruction, initialization, provision, revoking, polling, controlling, and access attributes.

AssetController is a uniform mechanism to manipulate resources. Asset is the uniform data structure for all kinds of resources. AssetController and Asset help form the uniform mechanism of abstraction, discovery, operation, and flow among different asset leasers for all resources. The asset abstraction is to make external heterogeneous resources as manageable uniform internal assets in operation level. The asset discovery contains unified naming, information storage, and searching based on attributes in global scale. The asset flow refers to an asset that can be transferred from one leaser to another. The typical example is that a virtual machine can be migrated from one physical node to another. The asset flow can make resource shared among different applications and improve resource utilization.

Definition 3. Lease represents the leasing relationship of a set of assets among the tenant and asset leasers. It can be denoted as Lease = <lid, tid, {aid}, effectiveTS, expiredTS, duration, preemptible, additionalTerms>.

a) ***Lid****,* the unique identifier of a lease in global.

b) ***Tid****,* the unique identifier of a tenant.

c) ***{Aid}****,* a specific set of asset identifiers.

d) ***EffectiveTS***, the effective timestamp of a lease.

e) ***ExpiredTS****,* the expired timestamp of a lease.

f) ***Duration****,* the duration of a lease.

g) ***Preemptible****,* whether the assets in a lease could be preempted or not.

h) ***AdditionalTerms****,* other information terms of a lease.

A Lease is the uniform data structure to express a leasing relationship, and it represents a match making result of users' asset requirements, which is held by a specific set of asset identifiers. According to the timeliness, lease can be divided into three types: a> ***Best Effort Lease***: it represents there is no constraint on the effective time and the expired time of the lease. The system can select any period to make the leasing relationship effective according to the duration of the lease. If there is no duration limit, we do not take the timeliness into account. b> ***Advanced Reservation Lease***: if there are requirements on both of the effective time and the expired time, it must ensure to start and end the lease on time; c> ***Deadline Driven Lease***: it represents the lease must take effective no later than the effective time or must end no later than the expired time. In addition, a lease can be divided into a preemptible lease or a non-preemptible lease according to whether the assets in the lease could be preempted or not. To make a lease effective, we should go through three stages: asset match making, reservation and leasing. After a lease ends up, the assets should be given back to asset leasers.

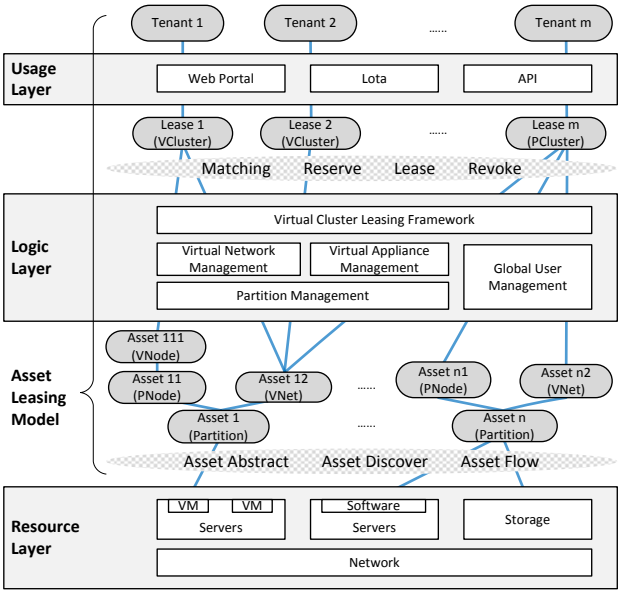

Figure 2. The Asset-Leasing Model Based Architecture of Vega LingCloud

B. Lease Life Cycle Management

The life cycle that the lease would experience from the birth to the end in the system is described as follows:

When a lease requirement is requested, it will directly enter into the ***Pending*** period. An appropriate timestamp will be calculated to the lease according to the lease's timeliness requirement. Then the lease will be submitted to the engine to be scheduled.

Negotiating: Asset Match Making is the primary task in this stage. According to the required attribute descriptions of assets in the lease, the lease management engine can find out a group of assets by the asset discovery mechanism. The engine designs a pluggable asset match maker mechanism in this stage, on which the application can freely implement its own asset match makers according to their logic. If the match making process fails, the lease will be ***Rejected*** and rolled back. Before the Preprocess stage, the lease could be ***Cancelled*** at any time. After a successful asset matching process, the lease management engine would automatically reserve all the matched assets, which would be locked simultaneously and refuse any other requests. Then, according to the timeliness requirment, the engine would arrange the next wake-up time of the lease.

Preprocess: the engine would automatically invoke the initialization operations of the matched assets to do some

preprocessing. Each asset's AssetController would determine how to initialize itself. If the initialization fails, the lease would enter into the ***Fail*** state, waiting to be deleted. On the contrary, it will enter into the ***Ready*** stage, when the engine will check the lease for the last time, ensuring everything ready before formally being leased.

Effective: all of the reserved assets would be leased after checking. In this stage, the engine will automatically poll latest states of assets in the lease.

The stages of Pending, Negotiating, Preprocessing, Ready, and Effective would check whether the lease is or not ***Expired***. There are two methods to deal with the expired lease. One is directly revoking assets, ending, and deleting the lease. The other is terminating the lease, but reserving the lease information and its assets for a period of time, and sending a message to the tenant. The tenant can choose whether to continue the lease or not.

III. The Architecture and Key Technologies of Vega LingCloud

Based on the asset-leasing model and the lease management engine, the infrastructure management system of Vega LingCloud, called Molva, is presented. Molva is a real example of RSLPS and adopts computer systems approach [8] to design the architecture. The main concerns of this approach are using uniform abstractions, and orthogonal mechanisms to build the system. Abstractions are designed to identify and solve the most important issues. Through orthogonal mechanisms, minimal functionalities could be implemented, and more functionality can be composed.

A. Basic Concepts

Molva contains six basic concepts: user, node, partition, virtual network, virtual appliance, and virtual cluster. All of these abstractions are extended from tenant, asset, and lease.

1) ***User*** represents the subject entity of the system. It is the **tenant** when the leasing relationship is built. There are two types of users in Molva: the host user and the cloud user. The host user is managed by the local OS, while the cloud user is managed by Molva. The cloud user can be authenticated both in the leased machines and the services provided by Molva. When a cloud user accesses the leased machines, he or she would be mapped to a host user. While a host user adopts a cloud shell to access services in Molva, it will be attached with cloud user identity transparently. This feature is provided by global user management in Molva.

2) ***Node*** is a kind of **asset** and it is a base abstraction to hold both virtual and physical machine information. By different types of AssetControllers, a node can sheild the different operations between physical and virtual nodes.

3) ***Partition*** is the boundary between internal assets and external resources, and the leaser of all assets in Molva. According to definition 2, a partition is also a kind of **asset** located at the root of the asset chain and a logical collection of nodes (including virtual and physical nodes), networks, softwares, and other forms of assets. Different assets with different type of AssetControllers are registered into partitions, and partitions supply uniform management operations of all assets and support resource leasing enforcement to different users and applications. In Molva, we can build partitions according to hetergeneous application modes: the DC Partition for the large scale data processing, the HPC Partition for the high performance computing, the VM Partition for the virtual machine leasing, the Storage Partition for providing storage, and the RAW Partition for applications which directly use resources. The key idea of partition is to organize a series of assets from the complex heterogeneity resource space according to some common characteristic, and adopting unified management mechanism for resources in the same partition. In addition, the partition support the flow of resources among partitions.

4) ***Virtual Network*** is regarded as a special kind of **asset**. The concept has two connotations: 1) A communication link which could be connected or isolated on demand; 2) An overlay among nodes. Virtual network can help users design and construct the required topology and network elements, including network interface card, bridge, gateway, router, MAC/IP address, etc.

5) ***Virtual Appliance*** is a kind of **asset** and contains a single or a set of templates to hold software stack, configuration, data, and resource requirements of applications. There are six features of virtual appliances in Molva: 1) Support for fast deployment; 2) Decouple applications and network configurations, e.g. IP address; 3) Configure single or multi-nodes constructed applications automatically; 4) Support for security: multiple applications, which are deployed based on the same appliance (with the same host accounts), can distinguish different tenants; 5) Support for usability: easy to make; 6) Support deployment on both virtual and physical nodes.

6) ***Virtual Cluster*** is a kind of **lease** between tenants and assets, involving physical nodes, virtual nodes, networks, and appliances. The virtual cluster can be built by physical nodes, and it can also be constituted by virtual nodes. Even more, it can be mixed by them.

There is a tight relationship among these six concepts in Molva: firstly, virtual network and virtual appliance are used to hold tenants' requirements for networks, applications, and resources. Besides, partition provide unified management operations for assets. Finally, the virtual cluster can be built and leased to tenants based on them.

B. Architecture

Molva is designed with the hierarchy approach as illustrated in Figure 2. There are three layers in total. The resource layer includes the hardware, software, and network infrastructure, which are the host environments for the cloud applications. Three types of capacities including computing, storage, and communication are provided by this layer. In this layer, the interfaces of access and management of external resources are abstracted by AssetController, which solves the resource abstraction problem. For some kinds of

assets, some management functions of AssetController will be implemented at the resource side to help manage assets. The logic layer is implemented based on the asset abstraction. The heterogeneous resources are abstracted as assets and organized in different partitions to support different application modes. Virtual network and virtual appliance management modules are introduced to maintain the corresponding resources. The virtual cluster leasing framework is the core module in the logic layer, which constructs the leasing framework of virtual clusters with the running mechanism in the lease management engine. Besides, this layer is also responsible for global user management. The usage layer provides the resource management and accessing interfaces for users. The web portal is the management interface of a Vega LingCloud instance, while Lota is the terminal for the tenants who leases physical or virtual machines. This layer also provides a suite of API for resource management and accessing to make third-party development feasible.

Simply speaking, the Molva architecture can be concluded as: Asset is the basic abstraction for the global assets. AssetController is the unified mechanism to operate assets. Lease is the representation for the leasing relationship between tenants and assets. With these mechanisms, Molva designs a unified approach of asset leasing: the virtual cluster management is used to make the decision for the lease of assets, such as nodes, networks, softwares, etc., and the partition management with several types of AssetControllers completes the enforcement.

C. Key Technologies

Three key technologies are adopted in Molva: application encapsulation and deployment based on virtual appliance, asset leasing and sharing framework based on partition, and global user management. For saving space, the goals, main methods, and effects of these technologies are shown in Table I.

TABLE I. KEY TECHNOLOGIES OF VEGA LINGCLOUD

Technology	Goals	Main Methods	Effects
Application encapsulation and deployment based on virtual appliance	To support encapsulation of applications; To support fast deployment of applications in the form of virtual clusters; To enable automated configuration of virtual networks and applications.	Copy-on-write images and snapshots [9] of VMs for application encapsulation and deployment; Embed and transparent DHCP based network and cluster context automated configuration.	Diverse applications are easy to be encapsulated into virtual appliances and redeployed in batches rapidly. Applications with their overlay networks and cluster configurations can be set up automatically in the target environment.
Asset leasing and sharing framework based on partition	To support heterogeneous application modes flexibly and elastically in a shared infrastructure; To provide asset leasing with policies in an organized way; To compromise among asset utilization, application performance, and user requirements.	Partition management for heterogeneous application modes; System image cloning [10] for elastic resource flowing; Five different scheduling policies based on system load and user preferences.	Heterogeneous application modes coexist in different partitions of the same infrastructure; Assets are assigned among applications elastically; Multi-tenants share the resources with isolation; Scheduling policies of the virtual cluser deployment compromise user requirements with utilization and performance.
Global user management [11]	To keep usability for both end users who access servers/VMs, and developers who engages in the further development; To ensure security isolation of the VMs derives from the same virtual appliances.	Linux PAM [12] and Windows GINA [13] for authentication integration; Grid process technology [14] for maintaining security context.	Global identities are enabled for servers/VMs logins and the applications developed based on this mechanism; Security isolation among users using the same appliance is enforced.

IV. EVALUATION

This section will evaluate Vega LingCloud by implementation cost, comparisons to other systems, and real application examples.

A. Implementation Cost

Vega LingCloud is implemented mainly in Java. Some OS-dependent modules are implemented in C/C++ or shell scripts. The amount of valid code in current implementation is counted as shown in Table II.

TABLE II. VEGA LINGCLOUD CODE STATISTICS

Language	Valid code lines	
	Total	*Inherited from Vega GOS [8]*
Java	262,741	185,405
C/C++	4,700	879
Bash scripts	2,881	486
Windows batches	1,528	0
Web pages/scripts	23,069	0
Total	294,919	186,770

B. Comparisons to Other Systems

Comparisons are conducted among Vega LingCloud (mainly about Molva), OpenNebula [15], and Enomaly ECP [16] in five non-functional goals: uniformity, flexibility, security, usability, and efficiency.

1) ***Uniformity.*** Molva is entirely designed based on the asset-leasing model with using uniform abstractions, and orthogonal mechanisms to build the whole system. However, OpenNebula with its ecosystem provides infrastructure management by combining several application level tools.

2) ***Flexibility***. Molva has achieved unified management of different types of assets by the AssetController mechanism, which is beneficial to the flexibility of the system. For instance, Molva can support the leasing of both physical and virtual machines, and can use different types of partition controllers to support heterogeneous application modes. While the OpenNebula and Enomaly ECP are only concerned about the virtual machine provisionning, so that

they have no other alternatives when confronted with difficult scenarios which cannot be solved by virtual machines.

3) ***Security.*** As mentioned in section III, Molva's global user management and partition management can protect the private information of users from being stolen while the other two systems do not guarantee this.

4) ***Usability.*** Molva and Enomaly ECP can both provide mechanisms to help users encapsulate applications while OpenNebula requires the administrator or the application deployment staff to encapsulate applications by themselves. What's more, Molva ensures that its users can have the global single sign-on feature by extending the local operating system authentication mechanism. Molva's appliance Agent is able to automatically configure the host environment, which also contributes to Molva's usability.

TABLE III. THE PERFORMANCE OF OPENNEBULA AND VIDA HYBRID SYSTEM TO DEPLOY 64 VIRTUAL MACHINES CONCURRENTLY.

Deploy-way (am: aria2 + metalink)	Cost time (s)	Deploy-speed (GB/s)
scp	1230	0.833
wget with 1 HTTP source	1200	0.874
am with 9 HTTP sources	1860	0.564
am with 1 BT seed	4200	0.250
am with 9 BT seeds	3060	0.343

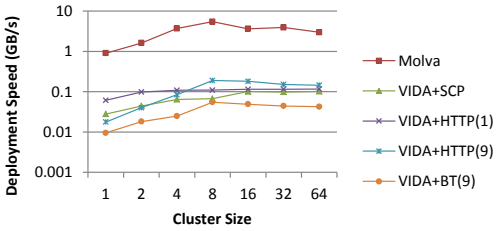

Figure 3. Average Speed of Different Ways for Concurrently Deploying Applications

5) ***Efficiency.*** The management overhead of a cloud system is concerned by users. We compare the application deployment speed between Molva and OpenNebula hybrid with VIDA [17].

The experimental environment consists of 8 Dell 1950 servers, and the configuration of each server is as follows: Intel (R) Xeon (R) CPU E5405, 2CPU * 4 core, 8 GB memory, and the operating system is CentOS x86_64. The hypervisor is Xen 3.1. NFS & NAS is used, and the IO bandwidth can reach 40 MB/s. We divide the experiments into 7 groups. The number of cluster nodes in each group increases according to the sequence 1, 2, 4 … 64. The configuration of each virtual machine is 1 vCPU, 1 GB memory, and 16 GB QCOW [9] virtual disk (actual size is 1.7 GB). Each virtual machine will deploy Hadoop platform and will configure itself automatically when it boots. Each test result is the average value of 20 repeated experiments.

We test the OpenNebula and VIDA hybrid system with 5 different deployment ways. As shown in Table III, in the concurrent deployment of 64 virtual machines, the OpenNebula and VIDA hybrid system can only reach a maximum speed of 0.874 GB/sec, while the average deployment speed in Molva is about 3.59 GB/sec. Therefore, the deployment speed of virtual machine in Molva is about 4.1 times of that in the OpenNebula and VIDA hybrid system. If we use 16 GB raw virtual disk instead, Molva has more obvious advantages as shown in Figure 3 because of the snapshot based appliance usage reducing the completed image copying. These experiments show that the management mechanism of virtual appliance in Molva has a better deployment performance than that in OpenNebula and VIDA hybrid system.

We also test the physical machine deployment speed. After optimizing the ways of deployment in SystemImager, we can deploy a physical machine with 300 GB disk in about 3 minutes.

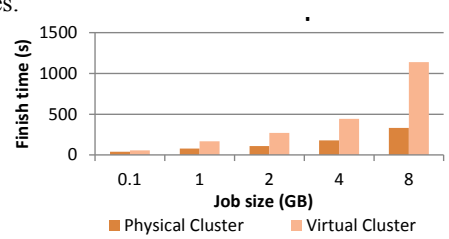

Figure 4. Finish Time for Hadoop WordCount

TABLE IV. FINISH TIME FOR SOME OTHER HADOOP JOBS

Jobs	Finish time (s)	
	*Physical cluster with 2 * 8 cores*	*Virtual cluster with 1 * 16 cores*
Pi (10^8 digits)	38	39
Pi (10^9 digits)	124	178
WordCount for 5500+ small files	1484	**1186**

C. *A Real Micro-Cloud Example in Research Group*

There is a small group in our affiliation with only 10 nodes (same resource configuration as above test, and every member has a PC for programming) to research projects in data computing. They have the requiments as follows: a) all the team members need machines to test their programs validity; b) after the preparation finished, they need to run their programs on physical nodes to get the real performance data. However, the total amount of resource is difficult to satisfy all of their requirements simultaneously, and there are always conflictions among the members. This case causes that the efficiency of the team is very low. These requirements are also very common in other research groups.

We adopt Vega LingCloud system to build a micro-cloud for their research activities. First of all, we build a DC partition (6 nodes at beginning) and a VM partition (4 nodes at beginning) to support data computing and virtual machine leasing modes. Then, we give the administrator account to the group leader, and register every member a valid cloud account. The group leader can lease one or a group of virtual machines to each member as their program validity checking. 4 physical nodes in VM parttion are enough to supply about 32 VMs to the team members. When someone completes the validity checking and needs to run the program in a real

environment, the team leader can lease the physical nodes in DC partition to satify the requirement. If the experiment environment scale is not enough, the team leader can migrate or shutdown other virtual machines, and adopt the asset flow mechanism among different partitions to migrate physical nodes from the VM partition to the DC partition to enlarge the scale. In addtion, the leasing management engine can manage the timeliness of all leases effectly, so it can drive all members to cherish the resource available time. This example demonstrates the advantages of Vega LingCloud in aspects of the uniform leasing mechanism of virtual and physical machines, and the supporting of heterogeneous application modes.

Application runtime performance is another concerned issue for users. In this example, we test 2 groups of Hadoop WordCount jobs separately on a physical machine cluster with 2 * 8 cores and a virtual machine cluster with 1 * 16 cores, and both of which are deployed by Molva. The virtual disks in this test are placed in local disk but not NFS. The total slots of mappers and reducers are 16. The results of finish time for different job sizes are shown in Figure 4. The finish time on the virtual machine cluster is 1.4 times of that on the physical machine cluster when the size of analyzed data set is 0.1 GB, while the multiple increases to 3.4 when the data set size goes up to 8 GB. But for the jobs of computing Pi or WordCount with a large number of small files in DFS, there is no explicit difference between a physical machine cluster and a virtual machine cluster, and sometimes a virtual machine cluster performs better, which is shown in Table IV.

In this test, we guarantee the total amount of resource is the same but the provision way (one uses virtual machine, and the other uses physical machine) is different, and we compare their performance. We find that in the case of big data set analysis, the physical cluster are better than the virtual cluster; in the case of CPU intensive workloads (computing Pi), the virtual cluster is close to the physical cluster; in the case of IO intensive workloads (WordCount) on number of small files, the virtual cluster is better than the physical cluster. From this experiment, we can see that the same resource with different provision ways can be fit for different workloads. So it shows that the consolidated leasing way of physical and virtual machines in Molva is valuable in real situations.

D. Real Application Examples

There are two application examples of Vega LingCloud in the real world environment: a private cloud of a large organization in Beijing and a public cloud in the Dongguan IDC of China. The typical deployment architecture of Vega LingCloud is shown in Figure 5. The resource scale of the Dongguan IDC Cloud has reached about 70 physical nodes with 504 cores, 625 GB memory, and 156 TB storage. And these resources are divided into four partitions: the DC Partition supported by the Hadoop platform, the HPC Partition supported by the Torque platform, the VM Partition supported by the Xen hypervisor, and the Storage Partition supported by NFS / Lustre. With the aid of different partitions, many application modes can be supported, involving large scale data processing, high performance computing, virtual machine leasing, and data storage. Up to now, the number of supported applications in the Beijing and Dongguan clouds has exceeded 35, including the web applications, management of information systems, simulation computing systems, office automation systems, and so on. Vega LingCloud has been widely recognized because of its flexible cluster leasing framework, rapid deployment, global user with single sign-on (SSO), and effective support for a variety of heterogeneous workloads.

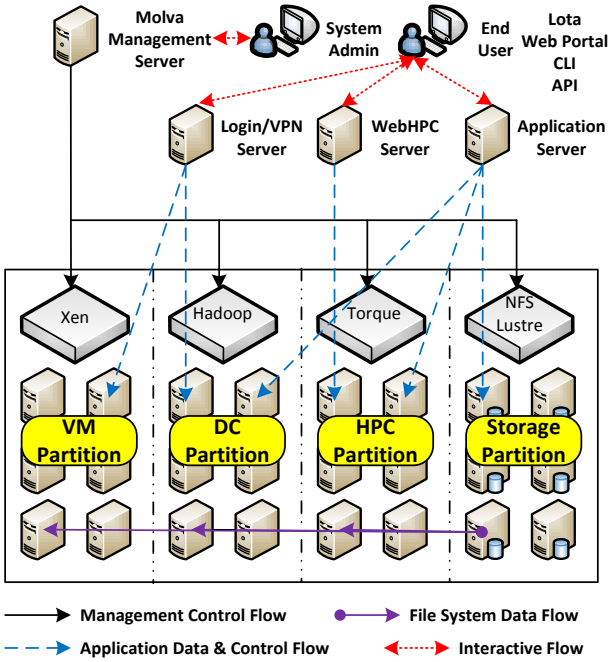

Figure 5. The Real Application Example of Vega LingCloud

V. RELATED WORK

Eucalyptus [18], OpenNebula [15], and Nimbus [19] are typical open source software frameworks for resource virtualization management. Eucalyptus is an IaaS framework. It implements a virtual machine service and a storage service, which are compatible with Amazon EC2/S3's interfaces. It is designed with a modular and extensible hierarchical architecture, which covers virtual machine, virtual network, and storage management. OpenNebula is a cloud computing toolkit for managing heterogeneous distributed data center infrastructures. It can orchestrate virtualization, network, and storage technologies to build multi-node and multi-tier services as virtual clusters on distributed infrastructures according to specific policies. The OpenNebula Ecosystem has been completed, including Haizea [20] – a leasing management architecture, VIDA [17] – a virtual cluster deployment architecture, and so on. Nimbus is a toolkit for building IaaS cloud. It leverages virtualization hypervisors and virtual machine schedulers to allow deployment of self-configured virtual clusters via contextualization. The Science Clouds [17] are the successful applications of Nimbus, which integrate a group of voluntary computing resources to

provide computing cycles for scientific communities. All of these systems are valuable to be studied to help build IaaS level Cloud. The common feature of these software frameworks is to provide virtual machine resources for the tenants based on the physical resources, expecting to form a virtualization resource pool decoupled with the physical infrastructure. However, for the complexity and performance sensibility of the real world applications, virtualization technology cannot support all the application modes.

Vega LingCloud refers to the pervious work, supporting both virtual and physical resource leasing from a single point. To the best of our knowledge, Vega LingCloud is the first to adopt single system based on asset-leasing model to support multiple application modes on shared infrastructure.

VI. Conclusion And future work

This paper defines the Resource Single Leasing Point System (RSLPS) to hold the requirements of cloud management system according to the real practical experiences, and presents the asset-leasing model based architecture and key technologies of a live system, Vega LingCloud. The implementation of the infrastructure management system in Vega LingCloud, called Molva, is the production example of RSLPS. By adopting resource partition mechanism in Molva, we have successfully built the Dongguan IDC cloud and a private cloud in a large organization in Beijing. All of these clouds can consolidate physical and virtual machines to support heterogeneous application modes on shared infrastructure. The resource scale of the Dongguan cloud has reached about 504 cores, 625 GB memory, and 156 TB storage. The number of supported applications in Beijing and Dongguan clouds has exceeded 35, and their modes include high performance computing, large scale data processing, virtual machine leasing, data storage, and so on. The experimental result of management overhead shows that the deployment speed of virtual machine in Vega LingCloud is 4.1 times of that in the OpenNebula and VIDA hybrid system in the case of 64 virtual machines deployment concurrently. From a real micro-cloud example, which is common in research groups, this paper shows that the consolidated way of leasing physical and virtual machine in Vega LingCloud is valuable in heterogeneous application mode supporing.

In the future, Vega LingCloud will be deployed in more IDCs or other organizations in China. We prepare to open the sources of the whole project.

Acknowledgments

We thank Vega LingCloud team members including Jie Liu, Shicai Wang, Ziyong Liu, Jiangning Cui, Guanyuan Zhang, Ruijian Wang, and Liang Li. We also thank Hongchen Guo, Litao Deng, Hao Li, and Peng Han from Beijing Institute of Technology for their tremendous hard work and contributions to the Vega LingCloud project. This research is supported in part by China Hi-Tech Research and Development Program (the 863 Program) (Grant No. 2009AA01A131 and 2009AA01A130) and China Basic Research Program (the 973 Program) (Grant No. 2011CB302502 and 2011CB302803).

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization,” Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, 2003, pp.164-177.

[2] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, “KVM: the Linux Virtual Machine Monitor,” in OLS '07: The 2007 Ottawa Linux Symposium, 2007, pp. 225-230.

[3] VMware, http://www.vmware.com/.

[4] OpenPBS, http://www.openpbs.org/.

[5] Torque, http://www.clusterresources.com/products/torque/.

[6] Apache Hadoop, http://hadoop.apache.org/.

[7] Y. Gu and R. L. Grossman, “Sector and Sphere: the Design and Implementation of a High-Performance Data Cloud,” Philosophical Transactions, Series A, Mathematical, Physical, and Engineering Sciences, vol. 367, 2009, pp.2429-2445.

[8] Z. Xu, W. Li, L. Zha, H. Yu, and D. Liu, “Vega: A Computer Systems Approach to Grid Computing,” Journal of Grid Computing, 2(2), 2004, pp.109-120.

[9] M. McLoughlin, “The QCOW Image Format,” http://people.gnome.org/~markmc/qcow-image-format-version-1.html.

[10] A. Righi, A. Jort, A. Gonyou, et al., “SystemImager v4.0.0 Manual,” Brian Finley, 2007.

[11] J. Lin, X. Lu, L. Yu, et al., “VegaWarden: A Uniform User Management System for Cloud Applications,” Proceedings of 5th IEEE International Conference on Networking, Architecture, and Storage, 2010, pp.457-464.

[12] V. Samar, “Unified Login with Pluggable Authentication Modules (PAM),” Proceedings of the 3rd ACM Conference on Computer and Communications Security, 1996, pp.1-10.

[13] GINA, http://msdn.microsoft.com/en-us/library/aa380543(VS.85).aspx.

[14] T. Liu, X. Li, W. Li, N. Sun, and Z. Xu, “Notes on a Run-Time Construct for Grid,” Journal of Computer Research and Development, Vol. 40, No. 12, 2003, pp.1811-1815.

[15] OpenNebula, http://www.opennebula.org/.

[16] Enomaly, http://www.enomaly.com/.

[17] VIDA, http://dc.crs4.it/projects/vida.

[18] Eucalyptus, http://www.eucalyptus.com/.

[19] K. Keahey, I. Foster, T. Freeman, et al., “Virtual Workspaces: Achieving Quality of Service and Quality of Life on the Grid,” Journal of Scientific Programming, 2005, 13(4): pp.265-276.

[20] B. Sotomayor, K. Keahey, and I. Foster, “Combining Batch Execution and Leasing Using Virtual Machines,” Proceedings of the 17th International Symposium on High Performance Distributed Computing, 2008, pp.87-96.

[21] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa, “Science Clouds: Early Experiences in Cloud Computing for Scientific Applications,” Cloud Computing and Its Applications 2008, 2008.