2012 IEEE 4th International Conference on Cloud Computing Technology and Science

LiveCloud: A Lucid Orchestrator for Cloud Datacenters

Xiang Wang, Zhi Liu
Department of Automation
Research Institute of Information Technology
Tsinghua University, China
{xiang-wang11, zhiliu08}@mails.tsinghua.edu.cn

Yaxuan Qi, Jun Li
Research Institute of Information Technology
Tsinghua University, China
Tsinghua National Lab for Information Science and Technology, China
{yaxuan, junl}@tsinghua.edu.cn

***Abstract*—Cloud datacenters, providing Infrastructure as a Service (IaaS), need to lively orchestrate numerous resource elements with diverse requirements of service provision, which most existing approaches are difficult to meet elastically. This paper presents LiveCloud, a management framework for resources in cloud datacenters. It addresses multiple management issues by employing different views of datacenter resources to satisfy various requirements for both tenants and operators. Leveraging software defined networking (SDN) techniques, LiveCloud further integrates network resources into datacenter orchestration and service provision with improved service-level agreements and faster service delivery. LiveCloud is architected as a data-centric and event-driven system with open management interfaces and service-oriented APIs to simplify system integration and service provision. The LiveCloud system has been deployed in both private and public datacenters to provide elastic cloud infrastructure for production applications.**

Keywords – software defined networking; orchestration; service provision

I. INTRODUCTION

It becomes a notable trend that many companies migrating applications, businesses and IT jobs, into cloud datacenters. The flourish of cloud computing has imposed tremendous management overhead upon datacenter operations. In recent years, virtualization of *technology components* [1], such as rack/blade server and storage-area network (SAN), has brought significant flexibility to cloud service providers. However, those virtualization techniques are difficult to meet the scalability and elasticity management goals for large-scale cloud datacenters.

The management of cloud firstly involves the maintenance of technology components status and topology of their interconnection to guarantee availability, scalability and ease-of-use of services. It also needs to support agile, cost-effective and on-demand IaaS provision to meet requirements of both resource and application from multiple tenants. In detail, cloud management systems should address the following issues:

- **Visibility**: The system should monitor the status of managed technology components, ensuring their availability, where heartbeat [3, 10] and other private control protocols [11] are typical leveraged. Besides, technology components in cloud datacenter provide diverse types of services, which are usually catalogued into computing, storage and network. Furthermore, several technology components using virtualization techniques are even enforced at different levels [20, 21] to meet different requirements for service provision. Therefore, the management system should be aware of various service types, so that technology components can be provided and catalogued for efficient orchestration.
- **Orchestration**: It involves two aspects: one is resource allocation, the other is resource coordination. With resource allocation, the management system needs to guarantee service-level agreements (SLAs) of datacenter resources, such as network bandwidth and latency for tenants. Besides, datacenter operator leverages management system to make cost-efficient decisions for workload placement. As a consequence, the resource allocation algorithms deployed in management system should take requirements from both operators and tenants into account. On the other side, allocated resources may access the datacenter network (DCN) at different layers. For example, virtual machines (VMs) are attached to software switches [12] in physical servers, and other non-virtualized resources, e.g. network security appliances, may be connected to the top-of-rack (ToR) switches directly. In multi-tenancy environment, the management system needs to perform coordination. For example, flow tables in switches are manipulated according to the resource's accessing place for elastic cooperation.
- **Provision**: The requirements of resource type, scale and topology vary among different tenants. The management system is responsible for providing various resources templates to efficiently deliver service and facilitate management. For example, many trial users apply single VM for light weight tasks. High performance computing tasks may require multiple VMs interconnected in a layer-2 network with high bandwidth and low latency. More complicated, organizations may deploy part of their IT infrastructure into the cloud along with restrict network security policy. Thus, both tenants and datacenter operators have the requirements of

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

resource utilization statistics for purposes of resource optimization and charging.

However, most existing solutions are difficult to meet those requirements simultaneously. They may suffer from one or more drawbacks in the following aspects:

- **Limited support for network resources in orchestration:** Most cloud management systems place and schedule workloads according to CPU load and memory usage, which lacks integration of network resources, such as connectivity, bandwidth and latency. Different cloud applications have different requirements of resources. Compute-intensive tasks require more CPU and memory resources, while I/O-intensive workloads may focus on network bandwidth and latency. Without careful consideration of these aspects, the orchestration may deploy workloads with network resources oversubscribed or not fully utilized [4]. Besides, the multi-tenancy cloud is a dynamic and mutable environment, and the distribution of datacenter capacity is frequently varying. Tenants come and go, which needs agile provision of isolated virtual networks. However, network resources in most cloud datacenters are usually configured manually and statically, which is difficult for swift resource coordination.
- **Requirement of flexible and multifarious service provision**: Many solutions in clouds are usually short of user-friendly and multifarious service provision. Allocated resources are presented and managed independently in simple fashion, e.g. a simple table listing all VM instances [22], which is absent of the service-oriented topology display. It not only limits the fast service provision for tenants' large-scale or distributed systems, but also hampers tenants to manipulate and manage resource efficiently. Besides, it is the lack of service-oriented management interfaces and rich resource utilization statistics that restricts on-demand provision and *pay-as-you-go* accounting.

This paper presents LiveCloud, a novel framework to integrate and orchestrate technology components in cloud datacenters. Leveraging the *software defined networking* (SDN) techniques [9], LiveCloud is able to lively integrate and orchestrate different types of technology components. It programmatically configures switches which have open interfaces [3], coordinating network resources to provide multiple isolated virtual networks. Furthermore, LiveCloud delivers tenants with multiple types of topology for diverse service deployments. Main contributions of this paper are:

- **System design and implementation**: LiveCloud addresses three management issues: visibility, orchestration and provision, by employing three corresponding views with different network topologies. Each view is the context under which the related issue is independently solved. LiveCloud implements and integrates a SDN controller which helps LiveCloud to tackle network and topology related problems, including network-involved resource orchestration and flexible service-oriented topology provision for different cloud applications. LiveCloud is architected on ports which bind to networks to establish the relation among those three views with event-driven processing model. It also exposes wieldy APIs for end-users programmatically controlling and managing services.
- **System deployment**: LiveCloud has been deployed in different scenarios. One is for private cloud in a university campus, providing typical services of enterprise network. The other is for public cloud in a Tier-IV datacenter, supporting quality-of-service (QoS) guaranteed IaaS delivery. In our deployment and evaluation, it is proven that LiveCloud works well in both public and private clouds. By July 2012, two production cloud applications are running in the virtual infrastructure provided by LiveCloud system.

The rest of this paper is organized as follows. Section 2 introduces LiveCloud design principle and system architecture. Section 3 presents the reference model and implementation of LiveCloud controller. Section 4 shows LiveCloud deployment in two datacenters. Section 5 introduces related work of LiveCloud. Section 6 draws our conclusion and discusses the future work.

II. LIVECLOUD ARCHITECTURE

The basic idea underlying LiveCloud is decouple. Instead of handling various aspects of resource management directly on technology components, we argue that it is necessary to separate those issues and limit their solution in certain views. As a consequence, LiveCloud introduces *physical view*, *logical view* and *tenant view* to solve visibility, orchestration and provision problems, respectively. In each view, LiveCloud refers to components which deliver services as *service elements* which are interconnected by networks. Both network topology and service elements are distinctively defined for different management issues processing.

A. Physical view

This view describes topology (*physical topology*) and technology components (*physical elements*) actually deployed in cloud datacenters. LiveCloud maintains parameters of physical elements it discovers, such as performance, capacity of memory or storage and supporting functions. Service visibility is solved in this view, and technology components are view as containers which are able to provide services of multiple types. For example, a physical Ethernet switch can provide high-bandwidth and low-latency forwarding capacity. And a physical server may provide both computing resource for running VM instances and network resources, i.e. software switches, for communication between VMs. The physical view employs a unified and open format of events and commands for every specific service type of delivering elements to wrap management interfaces. Thus, LiveCloud can amalgamate supervision and manipulation of distributed technology components in centralized manner. LiveCloud employs heartbeat protocol and vendor-providing interfaces to

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

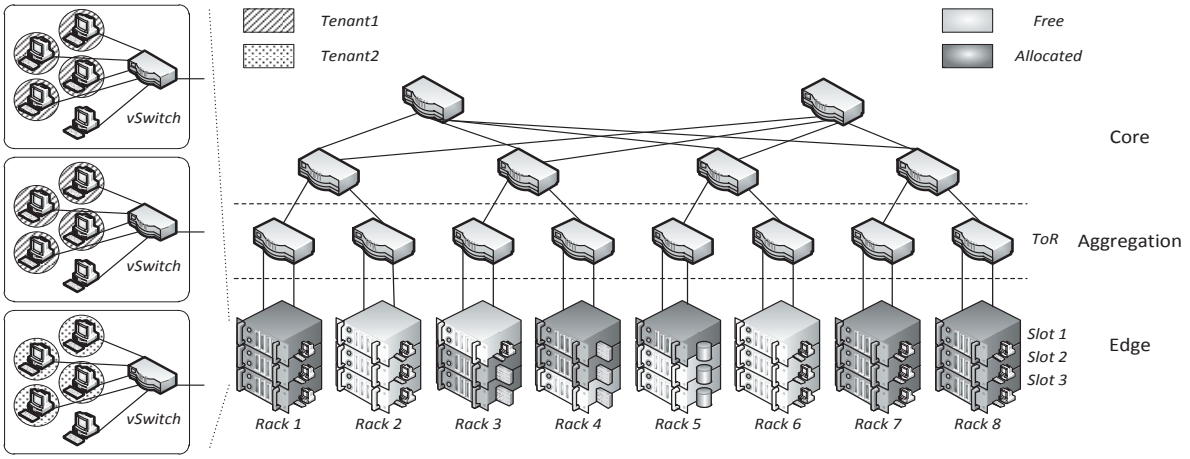

Figure 1. Physical view layout in multi-tenancy cloud datacenter

monitor the status of physical elements. Figure 1 depicts one typical physical view layout.

The physical topology is split into three layers: edge, aggregation and core. A fat tree is built with commodity Ethernet switches for network core, providing non-blocking bandwidth among directly connected appliances. The network core can also be implemented by employing vendor-specific appliances [1, 8]. Physical elements are located in edge layer, and access network core via either ToR switches or virtual server switches [12]. Network security and database elements may employ non-virtualized techniques [2], and computing service elements are in virtualized provision [17]. Physical elements can be identified using physical location, such as rack ID and slot ID. Two tenants' computing resources have been allocated among three host servers as the left of Figure 1 shows.

B. *Logical view*

As physical views in different datacenters may vary from one another, impervious orchestration in that view is too complex and inflexible to adapt to the dynamic and scaling cloud environments. LiveCloud needs a consistent topology to develop orchestration algorithms. And logical view is introduced for deploying resource allocation algorithms and performing coordination to help allocated resources gracefully cooperate. It defines multi-layer topologies according to different network infrastructure, and describes service elements accessing in cloud datacenter. In logical view, LiveCloud abstracts and quantifies various types of service elements, and forms unified resource pools. For example, CPU frequency and core number are transformed into numerical value comprehensively, representing computing capacity. Besides, bandwidth and latency are extra two network resources considered during resource allocation, as some cloud applications have specific requirements. Resource quantification takes the orchestration a major step of abstraction beyond direct manipulation of technology components. LiveCloud designs multiple evaluation models involving lease period and other SLA items to accommodate placement of workloads. Figure 2 shows the logical view in LiveCloud, which is corresponding to the physical view described former.

The logical view has a consistent three-layer access topology (*logical topology*). The outermost layer is composed of quantified technology components (*logical elements*), except for network resources. Ready-to-allocate and already-allocated resources in cloud are maintained in this layer, which is accordant to fashionable flat networking in cloud datacenters. In this view, the physical attributes of computing, database and network security are concealed using quantified values which are regarded as input of resource allocation algorithms. The intermediate layer is designed as access layer, in which the network resources are located. LiveCloud defines two sub-layers in the access layer of logical view, and each sub-layer is composed of multiple access switches providing access ports for logical elements accessing. We argue that I/O-intensive system deployment using multiple VMs should be placed in the same logical elements, which can leverage high bandwidth and low latency for east-west traffic provided by software switches in physical server. This model is suitable for cost-efficient cloud environments, which avoids the convergence of bandwidth at physical server network interface card. The innermost layer is network fabric which supports end-to-end non-blocking switching and has high bisection bandwidth in datacenter. Similarly, LiveCloud also supports the definition of forwarding model for network fabric, which allows operator to dynamically deploy traffic when using cheap plastic solution, avoiding congestion occurred in fabric.

C. *Tenant view*

For elastic provision and self-service arrangement, management system in multi-tenancy cloud datacenter should support on-demand resource allocation and isolate allocated resources among different tenants. Besides, it should expose flexible configuration interfaces of network topology (*tenant topology*) and service elements (*tenant elements*) to enable service-oriented management. Different tenants need diverse topology, according to their cloud application. Cloud management system ought to provide several templates of deployment for typical application use. LiveCloud delivers resources to tenant over virtual network with allocated elements attached. It introduces tenant view to

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

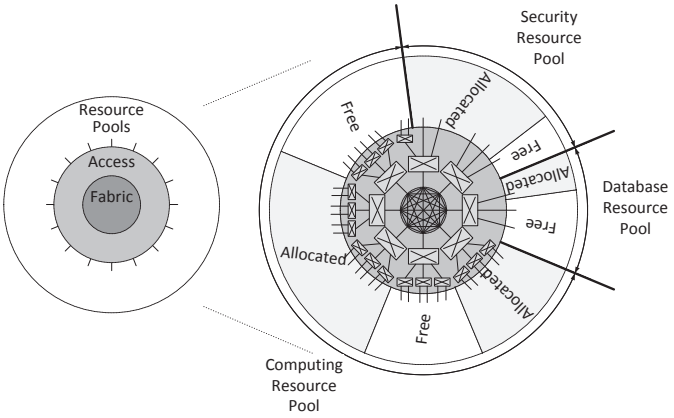

Figure 2. Logical view for orchestration in LiveCloud

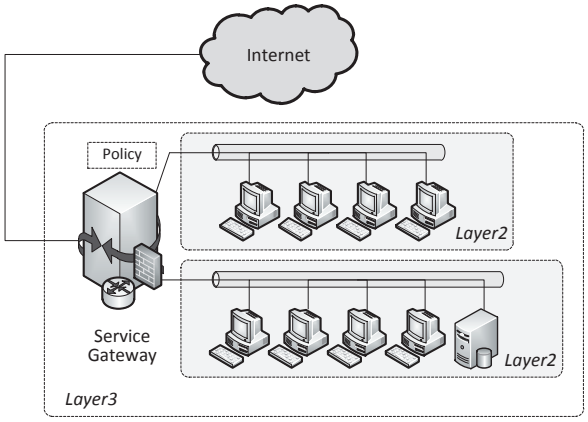

Figure 3. Tenant view for provision in LiveCloud

present layout and configuration details for certain tenant. Figure 3 shows the tenant view supported in LiveCloud.

For the tenant views, each tenant should be able to build and modify its own layer-2 networks, over which it can also design its own layer-3 networks. All layer-3 subnets can interconnect with each other using a service gateway which is also the gateway from tenant virtual network to Internet. Tenant can assign network security and QoS policies according to its view. All allocated resources are organized in that topology, and concealed from the others. The rational of tenant topology design is the requirements of most cloud applications, including both private and public cloud, can be satisfied by tailoring and reforming on this topology. Facing with different service requirements from diverse cloud application, tenant views vary from each other and can be tailored or formed into different topology described in Section I. LiveCloud also provides a text-based descriptive language based on XML for agile tenant view description and configuration. Table 1 shows the essential frame, omitting the closing tags. LiveCloud converts tenant's resource request into workflow. It performs inbuilt validation, submits requests and visually displays generated view.

III. LIVECLOUD IMPLEMENTATION

As service elements in all three views are interconnected and coordinated via networks, LiveCloud uses *generic network ports* to describe and manage attached service elements. This approach is also employed to re-correlate those decoupled views with less overhead. It maps and transforms the service elements, policies and topology described in tenant view to the configuration and manipulation of technology components actually deployed in cloud datacenter.

TABLE 1. TENANT VIEW DESCRIPTION LANGUAGE

```
<configuration>
  <vnet> <!-- a tenant view is named as a vnet -->
    <vgate> <!-- gateway is always enable in different topologies -->
      <public-interface> <!-- inf info, including MAC, IP bandwidth -->
      <interfaces> <!-- private interface, multiple -->
        <interface> <!-- inf info, similar to public-interface, vl2net binded -->

    <vl2nets> <!-- a tenant can apply for multiple vl2nets -->
      <vl2net> <!-- avl2net can be attached with multiple vm instances -->
        <vms>
          <vm> <!-- CPU, memory, disk and interface information -->
```

Furthermore, cloud datacenter is a scalable, dynamic and distributed environment. In order to master complexity of centralized management systems for massive scattered service elements, LiveCloud is architected as data-centric and uses *event-driven architecture* (EDA) to build service provision workflow triggered by incident. It monitors data changes and registers for notification which drives LiveCloud to perform adjustment. As a consequence, the data manipulation commands are asynchronous. Based on EDA, LiveCloud is able to improve performance and becomes responsive, enabling larger scale.

The system states tracked by LiveCloud controller are centrally stored. Differentiating the requirements of data update rates and availability, LiveCloud employs two different mechanisms to implement the storage of system states. For high durability and low update rate requirements, such as tenant elements location, tenant topology and policies, LiveCloud utilizes transactional persistent database. For high update rate requirements, such as resource utilization and historical statistics, LiveCloud employs NoSQL [23, 24, 25] techniques to handle large-scale, frequent and high-concurrency data update.

A. Generic Network Port

Network ports are essential objects in LiveCloud. A unique global identifier is assigned to each port in all three views. Port identifiers in different views have different descriptive structure. Service element information is treated as the parameters of the attached network port. And network behaviors are also designed using port-based description. LiveCloud stores mapping among port identifiers in all views. Thus, events generated by physical elements can be transformed into registered notification for tenant view processing, and adjustments in tenant view are mapped to executions in physical view.

- **Physical network port**: LiveCloud employs open standard switch devices and software switches to serve as ToR switches and virtual server switches, respectively. They monitor attached physical

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

elements, and receive decisions from their controller on whether physical elements can access datacenter network. Port identifier in this view encompasses switch global unique identifier and switch-associated port number. LiveCloud records all links for element accessing currently up in network to formulate physical network topology. One link describes connection between two ports. All attributes of physical elements are viewed as network port parameters, and stored in LiveCloud system. Thus, any physical element can be located using attached network port.

- **Logical network port**: LiveCloud regulates physical topology to a consistent and multi-layer logical topology, where it can imperviously orchestrates datacenter's resource candidates and tenants' resource requests. Workload placement calculated by orchestration algorithm is expressed in access port. And port identifier in this view encompasses information of hierarchy, layer-associated switch number and switch-associated port number. Accordingly, LiveCloud assigns a unique address for each logical element plugged in, which is structured accommodating to forwarding mechanism in network fabric. The unique address shadows attached port identifier and tenants' information, which can efficiently support QoS mechanism. LiveCloud implements a PortLand [6] -like forwarding protocol in network fabric and also designs a novel pseudo MAC addressing involving information mentioned above.
- **Tenant network port**: LiveCloud delivers resource and service to tenants via virtual network discussed in Section 2. Port identifier in this view encompasses tenant global unique identifier, layer-2 network number and network-associated port number. LiveCloud employs tenant topology to steer traffic in datacenter network, and network behavior is expressed as flow direction and connectivity between two network ports. Furthermore, for security and measurement purposes, tenant view involves those specific service elements, which redirects original one-hop traffic to a transparent middle box, like intrusion management system and load balancer. The behavior is defined as a specific traffic path in LiveCloud, and implemented in an indivisible end-to-end multi-hop sequence.

B. *Event-driven Model*

Cloud datacenter has a large scale physical network, and LiveCloud controller is architected in event-driven model to satisfy the requirements of scalability and programmability. A unified message format is designed for events and commands processing, and workflows are executed as message processing in LiveCloud. Figure 4 shows LiveCloud controller architecture.

LiveCloud controller has three layered modules: driver, kernel and application, which adapts to handling of three views. The two bottom layers are engineered for application development. Raw and infrastructure management related messages are tackled and concealed from service-oriented application design. LiveCloud application is directly built on manipulation of tenant view. Furthermore, in order to provide the method of fine-grained functions implementation, LiveCloud supports registration of concerned messages sensing which affects events handling.

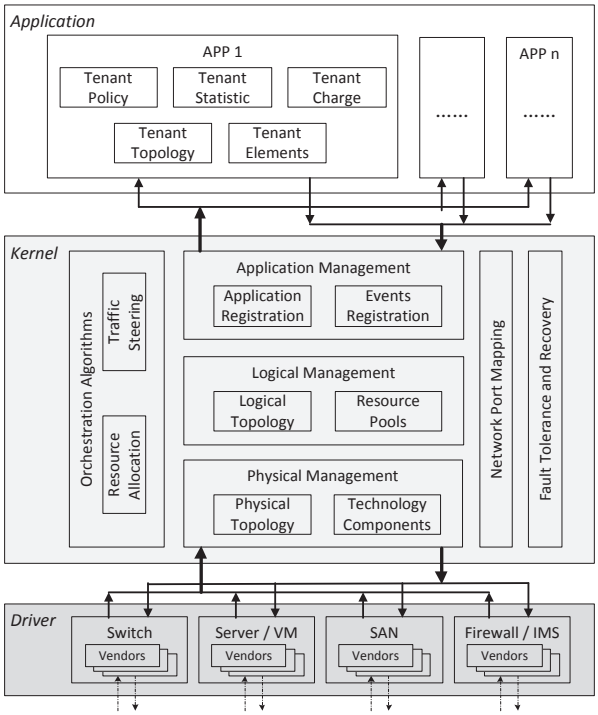

Figure 4. LiveCloud controller architecture

- **Driver**: Preparing for physical view generation, driver module abstracts a set of core functions and configuration actions for catalogued service supplier, such as OpenFlow compatible switches [26], hypervisor and network security appliances. It encapsulates them with LiveCloud unified message format, and wraps technology component management system. Driver module maintains communication between controller and technology components. Thus, it offers kernel module a reduced but sufficient set of interfaces to manipulate datacenter infrastructure.
- **Kernel**: In order to provide a flexible and convenient programming platform, kernel should not only support low level programming approach, but also expose service-oriented APIs that agile application design needs. This module includes physical and logical view generation essentially. It further supports registration for application, and stores network port mapping among physical, logical and tenant views. Kernel module listens for application joining, and supports multiple applications to register the update notifications of allocated resources states. After processing the messages of infrastructure management, kernel forwards messages transformed by mappings to

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

application who has subscribed before. In addition, kernel module is also responsible for stability with distributed backup databases.

The real intelligence of LiveCloud resides in the orchestration algorithms. The resource allocation algorithms ought to decide the placement of tenants' workload, and the traffic steering algorithms is responsible for updating the forwarding tables of network switches, according to tenants' topology. LiveCloud quantifies both technology components in datacenter and tenants' resource requests by certain measure. Benefiting from quantification, it is easy for LiveCloud to design multi-objective allocation optimization model. It comprehensively considers resource type, computing power, memory size, storage capacity, network bandwidth, lease period and other factors to generate optimal workload placement which achieves low power consuming, less resource fragment, guaranteed performance and other objectives. The placement decision generated by allocation algorithms are presented as candidates with ranks of different orientations. Tenants or Operators will decide the final placement. Then the kernel module dynamically steers traffic, according to those workloads placement.

- **Application**: On basis of abstraction and mapping of the two bottom modules, LiveCloud applications can simply employ service-oriented APIs to implement their designs. LiveCloud also provides frequently used libraries, coordinating with those APIs. Additionally, it has a GUI (graphical user interface) for tenants to drag service elements and draw topology to meet their needs. They can dynamically attach a service element to certain network port in tenant view where infrastructure specification and charge are associated.

IV. LiveCloud Deployment

LiveCloud has been deployed in two different scenarios: private cloud and public cloud. An experimental analysis was also conducted to evaluate the performance and scalability of LiveCloud.

A. Private Cloud

LiveCloud has been deployed in the FIT building at Tsinghua University. It supports private cloud for labs, cooperating with their enterprise network environment. The private cloud consists of 5 racks, each with 10 slots. Each rack contains 9 host servers, with about 100 ~ 120 VMs provision. 60 VMs have been reserved for security elements supported by [15, 16]. LiveCloud allocates private cloud resources for multiple labs. Each team is served as a tenant in LiveCloud, and a virtual L3 network is assigned. Websites and other network services have been migrated into the cloud. Besides, all labs' networks provide both wired and wireless connections to access private cloud. LiveCloud enables each team's local area network extending to virtual network in cloud, and VMs in cloud are peer nodes of members' desktops. Further, LiveCloud supports short-term lease for

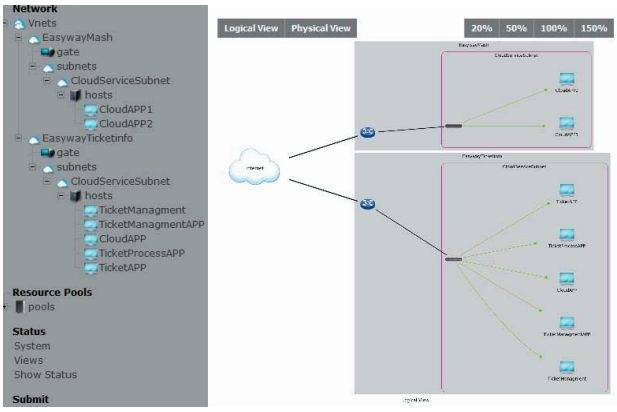

Figure 5. Views of Production Applications in Public Cloud

algorithm evaluation, network protocol testing and other research requirements.

B. Public Cloud

LiveCloud has also been deployed for public cloud in a Tier-IV datacenter. It manages 50 physical servers and provides over 1,000 VMs. All these resources are interconnected by commercial-off-the-shelf (COTS) core switches [8]. Fibre Channel (FC)-SAN is employed for storage service provision. LiveCloud provides public cloud tenants with an entire virtual datacenter (VDC) environment. Each VDC is a tenant-defined virtual L3 network, virtually isolated from other tenants' in the same datacenter. The on-demand virtual compute and storage resources are attached to the network, and all virtual L2 networks, as well as the Internet, are interconnected by a virtual gateway, on which tenants can specify their own management and security policies. Figure 5 shows the essential part of user interface for cloud datacenter operators. The left sidebar shows the abstract tree-form tenant view. It simply lists the tenant elements according to the tenant topology. The right part shows the status of service elements. The operator can click on them to browse the detail. The figure shows that there are two production cloud applications deployed on the IaaS provided by LiveCloud. Each of them is classic Web application. As a consequence, single virtual layer-2 network can meet their requirements.

C. Controller Deployment

The LiveCloud controller can be implemented using

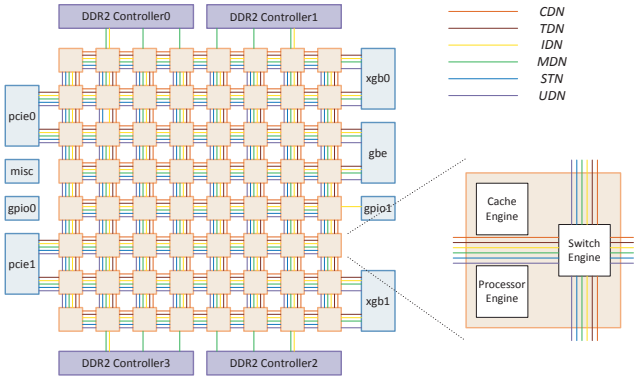

Figure 6. Tilera TILEPro64 architecture [19]

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

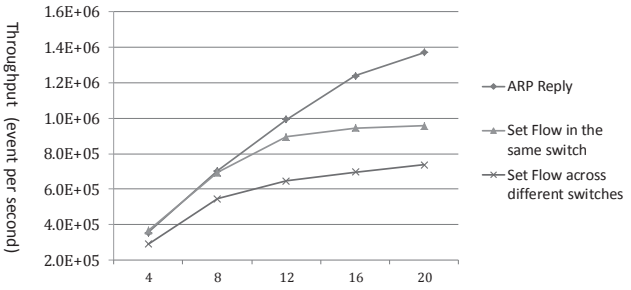

Figure 7. Throughput on Different Core Number

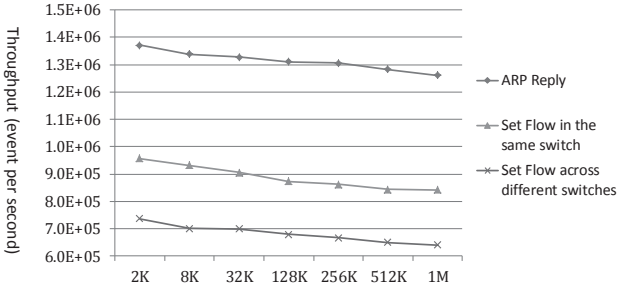

Figure 8. Throughput on Different Switch Number

commodity X86 servers. To achieve high performance, we also implement LiveCloud controller on Tilera [18], a high performance platform with low clock rate and low power consumption. Figure 6 shows Tilera TILEPro64 architecture. It provides Linux programming environment with optimized dataplane processing. It has 64 full-meshed processor cores and three programmable inter-tile networks for efficient communication. It has four on-chip memory controllers which can address up to 16GB shared memory with cache-coherent support.

The performance and scalability of LiveCloud controller are evaluated by measuring event processing speed on different core number and network scale. SmartBits600 is employed to generate packets as event triggers, and measures throughput of different events processing in event per second (eps) unit. Figure 7 shows the performance on different core number. Using 20 cores, the controller can reply 1,371K ARP requests, set 957K flows in the same switch and set 736K flows across two switches per second, respectively. Figure 8 shows the performance on different switch number. With network scaling, the performance reduction is about 8 % ~ 13%.

V. Related Work

Cloud datacenters consist of millions of technology components that provide computing, storage and networking resources. Value added services, including security, load balance and service acceleration are provided by specific appliances. As datacenter scales, it becomes difficult to put all these service elements together for efficient service provision. Network cannot effectively adapt to multi-tenancy, and multiple service elements are difficult to be lively involved. For network evolving in datacenter, both industry and academy present solutions to isolation and scalability [5, 7]. Thus, network, security and other similar elements should not be viewed as approach or add-ons any longer. They need to be treated as resources as computing and storage are, and should be well-orchestrated together.

To take global optimal orchestration, a network integration platform which has open industry standard APIs is needed to interconnect diverse service elements. Based on the platform, control logic can be built to dynamically change network behaviors, which is referred as SDN. OpenFlow is an open standard which provides a channel to manage forwarding tables in switches. And it is the key technique in SDN. Onix [9] is one of the most advanced OpenFlow controllers, and it also offers a general SDN programming platform. It can be used in almost every large-scale production network. However, it exposes APIs manipulating network states rather than service provision, which hardly meet agility, ease-of-use and service-oriented operation requirements in clouds.

VI. Conclusion and Future Work

This paper presents LiveCloud, a flexible orchestration platform for IaaS provision in cloud datacenters. Based on the decoupling of management issues, LiveCloud handles those problems in three different views with perspectives of physical, logical and tenants. Leveraging SDN techniques, LiveCloud brings network resources into orchestration, guarantying the SLA of network bandwidth and latency. To further step, LiveCloud delivers services with isolated virtual networks, which helps tenants to deploy their applications efficiently. This paper also presents LiveCloud controller's reference model. It takes generic network port as essential object, and is architected in event-driven model. Moreover, it provides service-oriented APIs for application developing, which significantly reduce the management complexity. We implement the prototyping system and deploy it in different objective datacenters.

In addition, several areas are identified as our future work. Firstly, LiveCloud controller is deployed on single device currently, and it will be a choke point as datacenter's scale grows. We believe event-driven model is suitable for controller evolving to distributed system. Secondly, multicast and broadcast has been designed by leveraging VLAN and multicast IP group to implement broadcast domain in tenant view, but currently still under development. Besides, resource allocation algorithm we have implemented is in static methods [13]. It will be improved with feedback consideration of element status, coordinating with migrating techniques. Moreover, further abstraction and quantification for security elements need to be designed to integrate hardware-based solution, not only software-based ones [14].

References

[1] Juniper Networks Inc., "Cloud-Ready Data Center Reference Architecture", White Paper, 2011.

[2] Juniper's Virtual Gateway. http://www.juniper.net/us/en/products-services/software/security/vgw-series/

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. In SIGCOMM CCR, 2008.

[4] H. Ballani, P. Costa, T. Karagiannis and A. Row-stron. Towards Predictable Datacenter Networks. In Proc. of SIGCOMM, 2011.

[5] J. Mudigonda, P. Yalagandula, J. C. Mogul, B. Stiekes, Y. Pouffary. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Data-centers. In Proc. of SIGCOMM, 2011.

[6] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In Proc. of SIGCOMM, 2009.

[7] A. Greenberg, J. Hamilton, and N. Jain. VL2: A Scalable and Flexible Data Center Network. In Proc. of SIGCOMM, 2009.

[8] Cisco Nexus 7000 Series Switches. http://www.cisco.com/en/US/products/ps9402/index.html

[9] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A Distribut-ed Control Platform for Large-scale Production Net-works. In Proc. of OSDI'10, 2010.

[10] libvirt: The virtualization API. http://www.libvirt.org

[11] BMC Cloud Lifecycle Management. http://www.bmc.com/products/product-listing/cloud-lifecycle-planning-management-software.html

[12] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M, Casado and S. Shenker. Extending Networking into the Virtualization Layer. In Proc. of HotNets-VIII, 2009.

[13] Y. Gao, L. Li, J. Jiang, B. Yang, Y. Xue and J. Li. SEAL: Hybrid Resource Distribution for Multi-tenant Data Centers. In Proc. of CAMAN, 2012.

[14] Y. Qi, F. He, K. Wang, X. Chen, J. Fong, F. Xie,Y. Shao, Y. Gao, Y. Xue, J. Li. LiveSec: OpenFlow-based Security Management for Production Net-works. In Proc. of the IEEE INFOCOM (Demo), 2011.

[15] Snort. http://www.snort.org

[16] Clam AntiVirus. http://www.clamav.net

[17] Xen Cloud Platform. http://www.xen.org/products/cloudxen.html

[18] Tilera Corporation, "TILEmpower Appliance Us-er's Guide", User Guide, 2011.

[19] Tilera Corporation, "Tile Processor Architecture Overview for the TILEPro Series", User Guide, 2011.

[20] lxc Linux Containers. http://lxc.sourceforge.net/

[21] KVM. http://www.linux-kvm.org/page/Main_Page

[22] OpenStackDashboard.http://wiki.openstack.org/OpenStackDashboard

[23] The Apache Cassandra Project. http://cassandra.apache.org/

[24] MongoDB. http://www.mongodb.org/

[25] Redis. http://www.redis.io/

[26] OpenFlow Switch Specification. http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf

978-1-4673-4510-1/12/$31.00 ©2012 IEEE