Compass: A Data Center Bootloader for Software Defined Infrastructure

Shuo Yang, Weidong Shao, Haoyu Song

Huawei Technologies
Santa Clara, CA, USA, 95124
email: {shuo.yang, weidong.shao, haoyu.song}@huawei.com

Wei Xu

Tsinghua University
Beijing, China, 100084
email: weixu@mail.tsinghua.edu.cn

***Abstract*—In this paper, we present a design of data center deployment automation system, *Compass*, for bootstrapping a software defined infrastructure, including network and compute nodes. *Compass* automates the process of bootstrapping a Software-Defined Networking (SDN) -based network from bare-metal networking devices and provisioning the bare-metal servers through the SDN network in a unified management approach. The unified and streamlined deployment management of networking and compute resources not only reduces the initial deployment cost, but also provides a way to automatically scale out data center infrastructure's capacity horizontally after the initial infrastructure setup (e.g., adding networking and computing resources, etc.). Using *Compass*, we have deployed a private cloud in a medium scale data center with around 200 commodity servers and over 20 SDN switches in Tsinghua University. We present the case study in this paper to illustrate the benefits of Compass as a unified deployment and management tool in a data center's software define infrastructure.**

Keywords–Compass; data center; bootloader; automation; SDN.

I. INTRODUCTION

Servers and network devices (e.g., switches and routers) are key components of the data center infrastructure. In the past, the two parts are usually provisioned and managed by different teams using different tool sets. This situation poses several challenges in efficient system planning, optimization, and debugging, and in turn incurs high operational cost and low return on investment.

Ideally, the two parts should be treated as an integral entity. But in reality, they are handled separately from administrative perspective. There are also some technical reasons for this separation. One reason is that networking devices are vertically integrated, and the deployment procedure is handled very differently from server management. However, we observed several appealing trends in industry which can change the current status quo.

First, there is a trend of "software defined everything" movement. Server software has a long history of being 'software defined'. Software-Defined Networking (SDN) [1] has prevailed not only in academic research but also in industry adoption. Moreover, storage industry starts the 'software defined' roadmap and practice [2], [3]. IBM coined the term "software defined environment" to address the vision for automatic and dynamic computing infrastructure provision [4]. The trend becomes increasingly clear as the concepts and practices such as Software-Defined Data Center (SDDC) [5] and warehouse-scale computing [6] prevail. The entire data center can be modeled as one big computer comprised of distributed computing/storage nodes, which in turn are interconnected through a network fabric comprised of switches and routers.

Another trend we see is the "open everything" movement. Not only the entire software stack for applications [7], [8], operating systems [9], and cloud managements [10] have been opened up, but also the hardware itself [11]. On the one hand, the data center building blocks are all standard-based. The choices of both software and hardware to construct the data center are abundant and cost efficient. On the other hand, these choices can become overwhelming for data center operators. The sheer scale of the open ecosystem can be daunting even to experienced Information Technology (IT) staff. There is apparently a lack of a capable orchestrator which can glue every piece of the system together organically and make them run in concert.

The interesting question we would like to answer is: Can we consolidate the best tools and automate the deployment of entire data center infrastructure in a seamless way? To be specific, we assume a greenfield deployment of new data center with pure bare-mental servers and switches. The only need from IT staff is to physically wire the devices together according some topology plan and then power up the data center. What if the data center administrators manage the modern software defined infrastructure deployment in the same way as a Linux system admin does today with a bootloader?

Though these questions sound like system administration related, we argue that they are critical if the industry wants to adopt new SDN technologies. As everything is software defined and both software and hardware are open, a 'bootloader' at data center scale is needed to deploy the software efficiently and coherently to various commodity hardware resource. A solution toward this will enable a unified portal and a coherent method to configure and manage the entire data center infrastructure, just as we do today for installing software components and services features onto a computer.

In this paper, we present the scheme and open source tool we developed, *Compass*, to enable a unified software defined infrastructure. We also share our successful experience on actual deployment of a data center in Tsinghua University, which has around 200 servers and over 20 SDN switches in the first phase. Our experience is more from 'out-of-box' system administration's perspective in greenfield SDN adoption scenario. We demonstrate that even a small step toward data center bootloader can significantly reduce the roadblock of SDN adoption in the context of entirely software defined infrastructure in data center, and we demonstrate that IT administration needs a 'think-out-of-box' methodology in this 'software defined everything' and 'open everything' era.

What we add is a unified bootloading system that works

 ISBN: 978-1-61208-404-6

on both switches and servers. After proper operating systems are installed and booted up, the switches and servers are further configured with the state-of-art open-source software, which can monitor and control the computing, storage, and networking components of the data center infrastructure.

The benefit of our approach is tremendous. It can significantly reduce the data center operation and maintenance cost. It allows the data center operators to focus on their core business, that is, to provide better data services to customers. It enables zero-touch scaling of the existing data center. New software and new hardware can be incrementally deployed without disturbing the normal data center operation.

This paper makes the following contributions:

First, it presents a unified deployment management system design, which will consolidate the infrastructure bootstrapping process. Traditionally, network infrastructure deployment and server/storage infrastructure deployment are considered separate efforts, which normally results in separate teams and prolonged engineering schedule. We present the first deployment system that unifies the above procedures and fills the gap in between in the era of SDN.

Second, it describes in details the first open system, *Compass*, which reflects the above vision. The novelty of *Compass* can be summarized as follows. (a) *Compass* is an open system not only in the sense that it is open sourced [12], but also in the sense that it is open to existing building blocks such as configuration management tools and OS provisioning tools through pluggable interfaces. (b) We present our engineering experience of quickly bootstrapping an SDN infrastructure and private cloud with a unified viewpoint.

The remaining of the paper is organized as follows. Section II describes the high level architecture of the data center and the procedure to deploy it. Section III explains the building blocks of our tool and the benefits of our design in greater details. Section IV presents the real word deployment of a data center and share our preliminary performance evaluation results. Section V compares our work with some existing related work, and finally, Section VI concludes the paper and suggests the future work.

II. High Level Architecture

In this section, we describe the high level architecture of an SDN-enabled data center and the role *Compass* plays in this architecture. We partition the data center into three tiers: controller tier, network tier, and server tier, as shown in Figure 1. The controller tier hosts all the tools that are required to configure and control the data center as well as all the software that will be installed on the switches and servers. Note that the controller tier contains servers for three different roles. These roles are relatively independant and can be realized on the same or different physical machines. Before actually deploy the data center, a data center blueprint should be prepared to specify the network address scheme, target service locations, virtualization scheme, network topology, and bandwidth allocations.

The network tier includes all the switches, which can be roughly mapped to the Fabric Elements described by Casado et al. [13]. The switches can be arranged in any topologies such as Clos, Fat Tree, and 3D Torus. In our design, we use the fat tree topology and a software-defined networking architecture for which the switch behavior is programmed and controlled by an OpenFlow [14] controller.

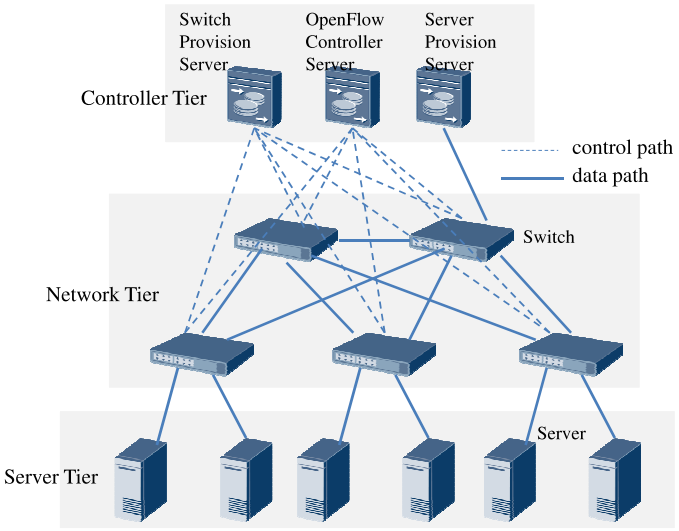

Figure 1: Architecture of Software Defined Data Center Infrastructure

The server tier includes all the servers. There are two types of configurations at this tier. One is to use virtual switches, such as Open Virtual Switch (OVS) [15], on the servers to enable edge intelligence. In this case, *Compass* will deploy a virtual switch on each server and configure the control plane connectivity to the same OpenFlow controller, which controls the overall network fabric. This step essentially extends the network tier into the server tier through virtualization. In this case, there should also be dashed lines from the OpenFlow controller server to all the servers in Figure 1. In the other type of configuration, virtual switches are not used. A server could still spawn multiple virtual machines, but all the networking packets to and from these virtual machines will be switched by the physical switches in the network tier.

The workflow is as follows. When the data center is powered up, the network switches are first installed with a Network Operating System (NOS) with OpenFlow agent enabled. This is done through the switch's control interface. After the NOS boots up, the OpenFlow agent hands over the switch control to OpenFlow controller. The controller first learns the network topology and then starts to configure the network. The controller conducts the network sanity check, partitions the virtual networks, configures the gateways, and provisions the flows and flow bandwidths. Once the network is configured, all the servers are reachable. The server Operating System (OS) and application software are then installed over the network. If virtual switches are installed in this step, then the OpenFlow controller must first configure these virtual switches to make the virtual machine reachable. Each virtual machine can then be provisioned individually.

III. System Design of Compass

To fulfill the vision and showcase a workable system described in previous sections, we designed *Compass*, a data center bootloader. It provides a unified view and workflow for the networking and server infrastructure deployment process of a software defined data center. Programmability and extensibility are the primary goals our design aims to achieve.

As shown in Figure 2, *Compass* provides six core com-

 ISBN: 978-1-61208-404-6

ponents: RESTful Application Programming Interface (API) engine, Resource Discovery engine, OS Provisioning engine, Package Deployment engine, Messaging engine, and Data Persistence engine. Here is how they work together as a system. Through the Restful API engine, *Compass* user can specify how they would like to design the software defined data centers, i.e., what the result system looks like. Resource Discovery engine provides the functionality to automatically discover hardware resources with corresponding network topology information in the data center once they are physically rack-and-stacked. For example, *Compass* uses SNMP protocol to query MIB table on switches to figure out the server MAC addresses connected to specific networking devices. Since each machine will send ARP requests to a switch during bootstrapping process, this approach gives *Compass* a view of all computing devices. Moreover, as long as the computing resources are rack-and-stacked following well-defined rules, *Compass* can translate its port position information into the location information; therefore a physical to logical mapping is created. OS Provisioning engine can install the specified operating system or hypervisor accordingly, if needed, onto those physical resources. Package Deployment engine will specify the service package for different building block and properly configure them into the functioning states. The above engines communicate with each other through the Messaging engine, so that everything is push-based event driven. This design makes *Compass* orchestrate the complex deployment process of whole data center like a symphony. Last but not least, the Data Persistence engine is used to store the states for *Compass* to act properly at each step.

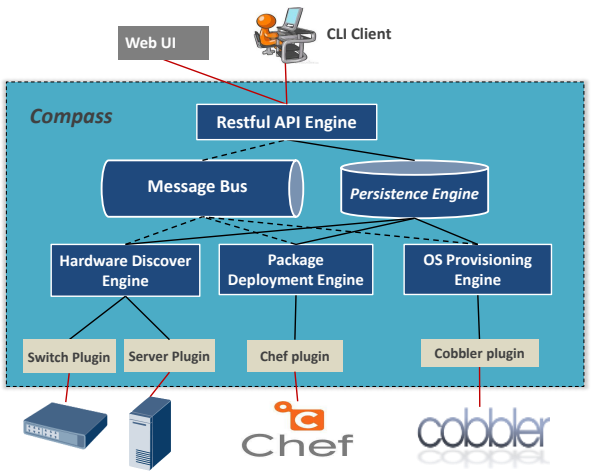

Figure 2: Compass Software Function Compoments

A. Programmability

Compass's programmability is enabled through a RESTful API engine. A set of APIs provide a programming contract to external applications. Each Uniform Resource Identifier (URI) [16] in our RESTful API corresponds to a resource object in data center infrastructure. A resource object supports operations such as create, update, delete, and execute actions. In addition, *Compass* provides a client-side Python library as a RESTful API wrapper to facilitate complex data center deployment scenario. A user can configure and deploy a cluster through Command Line Interface (CLI) scripts.

Table I shows a subset of resource objects. */switches* and */machines* represent the hardware resources *Compass* discovers and deploys the target systems to (such as Haddop or OpenStack). */drivers* is a metaphor we borrowed from OS community, and it defines the final software infrastructure that *Compass* bring the data center into. */drivers* is composable in a data center, i.e., *Compass* can bring a data center into an OpenStack [10] compute cluster along with Ceph [2] storage cluster and with Pica8 [17] SDN switches as the networking fabric. */clusters* and */clusterhosts* are the resource objects describing the resulting systems. Note that we only show a subset of resource objects here to illustrate the programming capability of *Compass*. The other resource objects are omitted here for conciseness. Interested reader can refer to the *Compass* website [18] for detailed documents.

TABLE I: Example URI of Compass API

URI	Resource	Operations
/switches	Networking switches	create, edit
/machines	Physical servers	create, edit
/drivers	An installer of a target system(e.g, OpenStack)	create, list
/clusters	A cluster with a target system to be installed	create, edit, delete
/clusterhosts	A host in a cluster	create, edit, delete

B. Extensibility

The Resource Discovery engine, the OS Provisioning engine, and the Package Deployment engine are the heavy-lifting internals of *Compass*. The Resource Discovery engine can collect the physical resources that are connected to a management plane and understand the network topology through protocols, such as Link Layer Discovery Protocol (LLDP) and the resource capability through mechanisms such as Ohai [19]. The Resource Discovery engine updates the hardware cluster status in the Persistence engine and notify the other components through the Messaging engine.

After the hardware discovery is done, the OS Provisioning engine is able to deploy the corresponding operating system or hypervisor to the physical nodes following the setup instruction stored in the Persistence engine. Note that OS or hypervisor setup instruction is defined by *Compass* user through the RESTful API engine. Therefore, the behavior for this step is programmable. The current *Compass* implementation uses Cobbler [20] as the actual OS provisioning tool. Cobbler is integrated into *Compass* as an OS Provisioning engine driver plug-in. And then the Package Deployment engine follows the setup instruction or policy-based rules stored in the Persistence engine to deploy software components onto the resource nodes. Moreover, the Package Deployment engine is in charge of the proper configuration, such as setting up SDN controller IP and trust credentials on the SDN switches, so that the deployed distributed systems function as a logical cluster as they are designed. The current *Compass* implementation uses Chef [21] as the actual configuration management tool, and Chef is integrated as a package deployment driver plug-in.

As we can see from the above, *Compass* takes a 'microkernel' software architecture and uses plug-in mechanism to delegate works to the actual 'drivers'. In this way, it can

 ISBN: 978-1-61208-404-6

provide extensibility in the following dimensions with *minimal plug-in development*:

- It is extensible with regard to the server hardware that it can support, be it Dell, HP, or Open Compute Project (OCP) [11] servers.
- It is extensible with regard to the switch hardware that it can support, be it Pica8, BigSwitch, or OCP [11] switches.
- It is extensible with regard to the target systems that it can configure, be it an SDN enabled OpenStack cloud or SDN enabled distributed file system cluster such as Ceph and Hadoop cluster.
- It is extensible with regard to the network OS it can provision, be it PicaOS [17], BigSwitch Open Network Linux (ONL) OS [22], or Cumulus OS [23]; and it is extensible with regard to the server OS or hypervisor it can provision, be it CentOS, Ubuntu, or even ESXi.

Because of the above design principle, *Compass* code base is 'small'. It is around 6000 line of Python code at its core and the complexity of extending plug-in is low. Here is an example of extensibility through *minimal plug-in development* effort at the dimension of resource discovery. We supported Huawei switch for server hardware auto discovery in our initial development. During our real deployment scenarios, we encountered Pica8 switches and Arista switches. Because of our 'microkernel' architecture and plug-in interface, we were able to add just about 200 line of Python plug-in code to support these switches and achieve the same functionality.

IV. Experiment and Evaluation

In this section, we share our experience using *Compass* to deploy a private OpenStack [10] cloud with Pica8 [17] SDN switches as the network tier in Tsinghua University. We used the OpenStack's Grizzly release with Neutron as the cloud virtual networking provisioning engine.

OpenStack is an exemplification of the level of complexity of the modern software defined infrastructure. It requires not only configuring the networking infrastructure but also configuring the server infrastructure. Configuring a production OpenStack cloud is notoriously deemed as a maze for most system administrators. Moreover, the majority of OpenStack issues originated from the networking misconfiguration. The number of Neutron (networking) related configuration is around 100, and the number of Nova (compute) related configuration is around 150. It is extremely hard if not impossible for administrators to properly configure the system in a productive way.

Figure 3 shows how *Compass* deploys the entire cloud system not only the SDN networking infrastructure, i.e., the network tier described in Figure 1, but also the distributed system on the server tier. In the deployment process, the operator uses the web User Interface (UI) we provide to follow the step-by-step configuration – these steps essentially reflect the operator's thought process toward the design of the whole software defined data center. The web UI talks to our RESTful API engine to persist the design decisions for the heavy lifting components to make decision for the real deployment command and control process. It first deploy the Pica8 switches, which is the network tier as described in Figure 1. Currently, PicaOS does not allow OS provisioning. Therefore our OS provisioning step is skipped (as a no-op) at

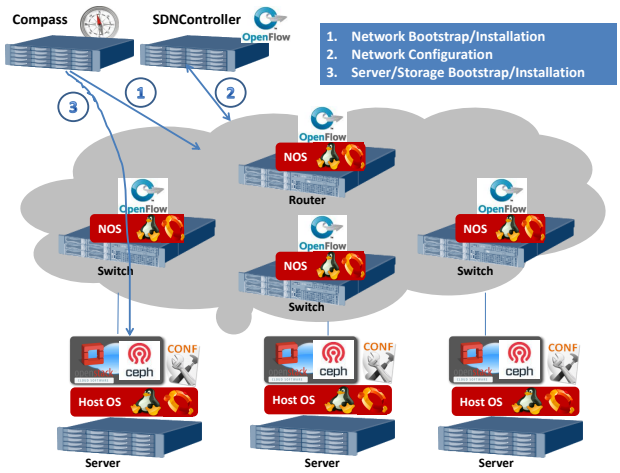

Figure 3: A Unified Process of the Entire Software Defined Infrastructure Deployment

this moment. The *Compass* Package Management engine kicks in right after the switches are discovered. During the package management process, *Compass* orchestrates Pica8 switches to configure themselves into the proper service state and establish connectivity to their controllers. We are working with Pica8 for OS provisioning using the Open Network Install Environment (ONIE) [24] approach, so that other OS, such as Debian, can be provisioned to the bare-metal switches. In the long run, we envision ONIE will be adopted by 'open switches', and therefore, the Compass logic can bootstrap switches from bare-metal. After this step, *Compass* proceeds to work on the servers including OS provisioning and package deployment of server software such as OVS and OpenStack management agents.

In our real deployment procedure, *Compass* helped the operator find the way out of the above maze. Instead of configuring totally 250 parameters, the operator only needs to program the RESTful API server through the web UI with 6 wizard-based steps. Our deployment process took a little over one hour to deploy the entire SDN enabled OpenStack cluster, with over 20 Pica8 switches and around 200 physical rack servers, from bare-metal to a fully functional OpenStack cloud.

V. Related Work

ONIE [24] is an open source project which solves the networking device OS installation problem. Here are some issues of ONIE. First, ONIE only works for network switch deployment and it does not provide a unified mechanism for both switch and server deployment. Secondly, ONIE provides the automation at individual device level, i.e., to benefit from ONIE, the bare-metal switch is required to pre-install a special boot-loader image while our tool eliminates this requirement (our current practice is taking a non-ONIE in the loop approach). ONIE naturally fits into *Compass* extensible plug-in architecture, in which *Compass* can leverage its capability to bring a global view of entire networking infrastructure deployment. We are working on the ONIE plug-in extension for ONIE enabled networking devices.

Some long standing software deployment solutions, such

 ISBN: 978-1-61208-404-6

as Rembo Preboot eXecution Environment (PXE) [25], IBM Tivoli [26], and Symantec Altiris [27], include the bootstrap of operating systems on a diversity of hardware architectures. However, these solutions all assume the networking infrastructure is ready for server software deployment.

Crowbar [28], an deployment automation project lead by Dell, assumes that network infrastructure has been deployed before it takes over the server software deployment process. This is a reasonable assumption in traditional data centers where vertically integrated networking boxes were the only option for networking infrastructure – an old paradigm that data center builders did not have an option to deploy software defined networking infrastructure. But as we have described, these assumptions do not hold any more. *Compass* is different from Crowbar as it designed for not only server infrastructure deployment but also for networking infrastructure deployment through a unified viewpoint.

Fuel [29] and TripleO [30] are tools tightly designed for deploying OpenStack cloud management platform, while *Compass* is designed for extensible capability toward software defined infrastructure deployment. We extended *Compass*'s capability to support Ceph deployment through a Ceph driver while keeping the rest of code unchanged (see III-A for the driver concept). Configuration management tools such as Chef [21], Puppet [31], and Ansible [32] provide configuration capability for software. However, they do not provide resource provisioning capability, which is a key step in data center deployment. As we described, *Compass* is open to utilized these tools as components of its automation process and it delegates the configuration management functionality to these existing tools to avoid re-inventing wheels. Specifically, our current implementation use Chef as our configuration management plug-in and we are working toward an Ansible plug-in.

VI. Conclusions

The hardware and software decoupling is becoming the new norm of the network camp thanks to the advent of the software defined networking and the open network movement. The ubiquitously available open-source software and bare-mental devices gives data centers unforeseen opportunity to optimize their cost structure and provide agile services at scale. Finally, we are able to program and control network devices just like we program and control a server. Then, why would we still need to use two different tool chains and skill sets to deploy and manage servers and networks?

In this paper, we revisit the data center provision problem. We treat the entire data center infrastructure as an organic entity and use a unified tool to deploy and provision the data center from a scratch. SDN unlocks network management in a great simplicity, and our system utilizes that promise and demonstrates the real benefit in a greenfield case study from system administration perspective. Our approach is fundamental in that it consolidates the two historically separate worlds together and significantly simplifies the data center infrastructure deployment. As far as we know, this is the first such tool available with this vision in industry.

Compass is just a start of the effort of building a unified management tool for software defined infrastructure. Several works are ongoing. As *Compass* is a 'microkernel' design and can be easily extended in functionality, we are working on ONIE integration for switches pre-installed ONIE when they are shipped to data center. We are working on handling the server like PXE booting sequence for networking devices as another *Compass* plug-in. All of these promise the unification mechanism for both server and networking devices.

Thanks to the SDN evolvement, the last locked infrastructure in data center is now opening up. Data center infrastructure deployment is just the start. Our tool can be extended and integrated with other tools to continue managing, monitoring, and controlling the data center. For example, servers and switches can be upgraded or replaced without interrupting the data center services. Our tool should be able to schedule and automate the tasks with minimum human interference. While we leave these as our future work, we are confident that the data center automation would become the first-class requirement in building, running, and maintaining a data center. It is our hope to evolve *Compass* to become an invaluable tool for future cloud providers.

VII. Acknowledgment

We would like to thank Sam Su, Xiaodong Wang, Xicheng Chang, Grace Yu, Jiahua Yan for participating the design and implementation of *Compass*.

References

[1] D. Kreutz et al., "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, January 2015, pp 14-76.

[2] S. A. Weil, S. A. Brandt, E. L. Miller, D. E. Long, and C. Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System," in *OSDI*, 2006, pp. 307-320.

[3] E. Thereska et al., "IOFlow: A Software-defined Storage Architecture," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 182-196.

[4] Software Defined Environment. [Online]. Available: http://www.ibm.com/systems/infrastructure/us/en/software-defined-environment/ [retrieved: March, 2015]

[5] The Journey Toward the Software Defined Data Center. [Online]. Available: http://www.cognizant.com/InsightsWhitepapers/The-Journey-Toward-the-Software-Defined-Data-Center.pdf [retrieved: March, 2015]

[6] L. A. Barroso, J. Clidaras, and U. Holzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 2nd ed., ser. Synthesis Lectures on Computer Architecture. Morgan Claypool Publishers, 2013.

[7] K. Douglas and S. Douglas, *PostgreSQL*. Thousand Oaks, CA, USA: New Riders Publishing, 2003.

[8] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.

[9] A. Kivity, "KVM: the Linux Virtual Machine Monitor," in *OLS '07: The 2007 Ottawa Linux Symposium*, July 2007, pp. 225-230.

[10] OpenStack: The Open Source Cloud Operating System. [Online]. Available: http://www.openstack.org/software/ [retrieved: March, 2015]

[11] Open Compute Project Community. [Online]. Available: http://www.opencompute.org/ [retrieved: March, 2015]

[12] Compass Github Entry. [Online]. Available: https://github.com/stackforge/compass-core [retrieved: March, 2015]

[13] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A Retrospective on Evolving SDN," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN)*, 2012, pp. 85-90.

[14] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, April 2008, pp. 69-74.

[15] B. Pfaff et al., "Extending Networking into the Virtualization Layer," in *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*, 2009, pp. 1-6.

[16] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, 2000, aAI9980887.

 ISBN: 978-1-61208-404-6

[17] Pica8 Switch and Network OS. [Online]. Available: http://www.pica8.com/ [retrieved: March, 2015]

[18] Automating Distributed Systems Deployment. [Online]. Available: http://www.syscompass.org/ [retrieved: March, 2015]

[19] Document of Ohai. [Online]. Available: http://docs.opscode.com/ohai.html [retrieved: March, 2015]

[20] Cobbler Project Website. [Online]. Available: http://www.cobblerd.org/ [retrieved: March, 2015]

[21] Online Documentation for Chef. [Online]. Available: https://wiki.opscode.com/ [retrieved: March, 2015]

[22] BigSwitch Controller and Network OS. [Online]. Available: http://bigswitch.com/ [retrieved: March, 2015]

[23] Cumulus Linux. [Online]. Available: http://cumulusnetworks.com/product/overview/ [retrieved: March, 2015]

[24] Open Network Install Environment. [Online]. Available: http://onie.github.io/onie/ [retrieved: March, 2015]

[25] REMBO: A Complete Pre-OS Remote Management Solution REMBO Toolkit 2.0 Manual. [Online]. Available: http://goo.gl/DRQOdl [retrieved: March, 2015]

[26] IBM Tivoli. [Online]. Available: https://www.ibm.com/software/tivoli [retrieved: March, 2015]

[27] Endpoint Management powered by Altiris Technology. [Online]. Available: http://www.symantec.com/endpoint-management/ [retrieved: March, 2015]

[28] The Crowbar Project. [Online]. Available: http://crowbar.github.io/home.html [retrieved: March, 2015]

[29] Mirantis Fuel Project. [Online]. Available: http://software.mirantis.com/key-related-openstack-projects/project-fuel/ [retrieved: March, 2015]

[30] OpenStack on OpenStack. [Online]. Available: https://github.com/openstack/tripleo-incubator [retrieved: March, 2015]

[31] Online Documentation for Puppet. [Online]. Available: https://docs.puppetlabs.com [retrieved: March, 2015]

[32] Online Documentation for Ansible. [Online]. Available: http://docs.ansible.com/ [retrieved: March, 2015]

 ISBN: 978-1-61208-404-6