CLOUDS-Pi: A Low-Cost Raspberry-Pi based Testbed for Software-Defined-Networking in Clouds

Adel Nadjaran Toosi, Jungmin Son, Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) Laboratory,
School of Computing and Information Systems,
University of Melbourne, VIC, 3010, Australia
anadjaran@unimelb.edu.au
October 31, 2017

Abstract
Software Defined Networking (SDN) is rapidly transforming the networking ecosystem of cloud computing data centers. However, replicating SDN-enabled cloud infrastructures to conduct practical research in this domain requires a great deal of effort and capital expenditure. In this paper, we present the architecture of the CLOUDS-Pi, a testbed for conducting research on SDN-enabled cloud computing. As part of it, Open vSwitch (OVS) is integrated with Raspberry-Pis, low-cost Linux-based computers, to build up a network of OpenFlow switches. We provide two use cases and perform validation and performance evaluation for our system. We also discuss benefits and limitations of our platform in particular and SDN in general.

1 Introduction
Software-defined networking (SDN) is an emerging approach to computer networking that separates the tightly coupled control and data (forwarding) planes in traditional networking devices. Thanks to this separation, SDN can provide a logically centralized view of the network in a single point of management. This is achieved via open interfaces and abstraction of lower-level functionalities and transforms the network to a programmable platform that can dynamically adapt its behavior. Early supporters of SDN were among those who believed that network device vendors were not meeting service providers’ needs, especially in terms of innovation and the development of required features. Other supporters were those who aimed at harnessing their inexpensive processing power of commodity hardware to run their network. The need for such agile, flexible, and cost-efficient computer networks has consequently formed the nucleus for the global efforts towards SDN [1].

Cloud computing [2] is a successful computing paradigm, which delivers computing resources residing in providers’ data centers as a service over the Internet
in subscription-based. With the growing adoption of cloud, data centers hosting cloud services are rapidly expanding their sizes and increasing in number. Therefore, resource management in clouds’ large-scale infrastructure becomes a challenging issue. In the meantime, SDN is increasingly being accepted as the new generation of networks in cloud data centers where there is a need for efficient management of large multi-tenant networks of such dynamic and ever-changing environments. In fact, SDN not only reduces the complexity seen in today’s cloud data center networks but also helps cloud providers to manage network services from a central management point.

The peaceful meeting of cloud computing and SDN embarks on significant innovation and research activities to fuse these together. However, evaluation and experimentation of SDN-based applications for cloud environments presents many challenges in terms of complexity, scaling, accuracy, and efficiency [3]. Moreover, rapid and affordable prototyping of SDN-enabled clouds is difficult and significant capital expenditure is required to replicate practical implementations obstructing the evaluation of research in this domain. Even though software emulators such as Mininet [4] expedite rapid prototyping of SDN on a single machine, there is not enough support for network dynamicity and the performance of the virtualized hosts and Virtual Machines (VMs) in such tools [5].

With these issues in mind, this paper proposes the system architecture and design of CLOUDS-Pi, our testbed platform in the CLOUDS laboratory for conducting research on SDN-enabled cloud computing. We focus on the cost-effectiveness of our setup by reusing existing equipment and coping with the budget and space limitations of an academic research laboratory. One of the most important benefits of SDN is that commodity hardware can be used to build networking devices. Therefore, we use Raspberry Pis, low-cost and small single-board computers, to build a small-scale data center network. To make a switch out of Raspberry Pi, we integrate each Pi with an Open vSwitch (OVS) [6], which is one of the most widely used virtual switches in SDN.

The rest of the paper is organized as follows. Section 2 discusses the system architecture of the CLOUDS-Pi platform along with design choices and lessons learned. Employing two use cases, we validate and evaluate our testbed in Section 3. Following that in Section 4, we discuss challenges and limitations of SDN along with opportunities and future research directions for SDN in clouds. Finally we conclude our paper in Section 5.

2 System Architecture

We present our CLOUDS-Pi research platform along with the physical infrastructure setup and utilized software.

**Physical infrastructure:** The main aim of our small cloud data center is to provide an economical testbed for conducting research in Software-Defined Clouds (SDCs) [7]. Therefore, we focus on reusing existing infrastructure, equipment, and machines (hosts) connected through a network of OpenFlow switches made out of Raspberry Pis, used by others as compute resources [8]). Our platform is comprised of a set of 9 heterogeneous machines with specifications shown in Table 1. To keep up with the common practice, we use three separate networks namely, 1) management, 2) data and 3) control. The management network is used by OpenStack, our deployed cloud operating system, for in-
Management Network (OpenStack)

SDN Controller
OpenStack Controller
Gateway

Internet

Data Network (Tenants)

NetGear FS516
TP-Link TL-SG1016D

Control Network (OpenFlow)

Figure 1: System Architecture of CLOUDS-Pi

Table 1: Specifications of machines in CLOUDS-Pi.

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU</th>
<th>Cores</th>
<th>Memory</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 x IBM X3500 M4</td>
<td>Intel(R) Xeon(R) E5-2620 @ 2.00GHz</td>
<td>12</td>
<td>64GB (4 x 16GB DDR3 1333MHz)</td>
<td>2.9TB</td>
</tr>
<tr>
<td>4 x IBM X3200 M3</td>
<td>Intel(R) Xeon(R) X3460 @ 2.80GHz</td>
<td>4</td>
<td>16GB (4 x 4GB DDR3 1333MHz)</td>
<td>199GB</td>
</tr>
<tr>
<td>2 x Dell OptiPlex 990</td>
<td>Intel(R) Core(TM) i7-2600 @ 3.40GHz</td>
<td>4</td>
<td>8GB (2 x 4GB DDR3 1333MHz)</td>
<td>399GB</td>
</tr>
</tbody>
</table>

ternal communication between its components and resource management. It constitutes a 16-port 10/100Mbps Ethernet Switch (NetGear Model FS516) connecting hosts and OpenStack controller. The data network is used for data communication among VMs deployed within the cloud environment and providing Internet access to them through the gateway host. This network is 100Mbps fat-tree [9] network built on top 10 Raspberry Pis (Pi 3 MODEL B) with OVS integrated each playing a role of 4-port switch with an external port for control. Raspberry Pi 3 Model B has four USB 2.0 port and only one Ethernet interface. Therefore, we used TP-Link UE200 USB 2.0 to 100Mbps Ethernet adapters to add four extra Ethernet interfaces to the Raspberry Pi switch. The control network connects control ports on Raspberry Pi Switches to the SDN controller and transfers OpenFlow packets between the controller and switches. Figure 1 depicts our data center setup including the network topology for management, data, and control networks and Figure 2 depicts the CLOUDS-Pi platform.

Software: We installed CentOS 7.0 Linux distribution as the host operating
system on all nodes. Then using RDO Packstack,\(^1\) we installed OpenStack to build our cloud platform. Given that our OpenStack controller resides beside the SDN controller, we used one of our more powerful machines (IBM X3500) as the controller node. All other nodes play the same role of compute hosts in our design. In addition, we created NAT forwarding rules on the controller using Linux iptables to enable all other nodes to connect to the Internet. In this way, our controller node configured as the default Gateway for the external network access (Internet access) for all other nodes and OpenStack VMs. We also setup L2TP/IPsec Virtual Private Network (VPN) server using OpenSwan and xl2tpd Linux packages on the controller node to provide direct access to VMs for our cloud users outside the data center network (see Figure 3).

CLOUDS-Pi uses OpenDaylight (ODL), one of the popular open-source SDN controllers, to provide the brains of the network and handle OpenFlow capable Raspberry Pi switches. ODL is installed on the same host as the OpenStack controller and manages OpenFlow switches via control network (See Figure 3). Every Raspberry Pi in our setup uses a Debian-based Linux operating system of Raspbian Version 8 (jessie) and have OVS Version 2.3.0 installed as an SDN-capable virtual switch. We configured OVS as a software switch having all USB-based physical interfaces connected as forwarding ports and used Raspberry Pi built-in interface as a OpenFlow local port (Control).

\(^1\)Packstack - RDO, https://www.rdoproject.org/install/packstack/
3 Use Cases

We introduce two use cases to illustrate how CLOUDS-Pi platform and its SDN-enabled features can be utilized to offer solutions and drive research innovations. The first use case shows dynamic flow scheduling for efficient use of network resources in a multi-rooted tree topology. The second use case demonstrates that the communication cost of pairwise VM traffic can be reduced by exploiting collocation and network locality through live VM migration.

3.1 Dynamic Flow Scheduling

Data center network topologies such as fat-tree typically consist of many equal-cost paths between any given pair of hosts. Traditional network forwarding protocols often select a single deterministic path for each pair of source and destination and sometimes protocols such as Equal-Cost Multi-Path (ECMP) routing [10] is used to evenly distribute load on multiple paths based on a certain hashing function. These static mapping of flows to paths do not take into account network utilization and duration of flows. We propose and demonstrate the feasibility of building a dynamic flow scheduling for a given pair of hosts in our multi-rooted tree testbed using ODL APIs.

The proposed dynamic flow scheduling algorithm aims to maximize the bandwidth between these given hosts while ensuring that the latency is minimized. The algorithm receives IP addresses of a given pair of hosts as input. It finds multiple paths of equal length between the source and the destination and iteratively (e.g., every 15 seconds) checks the statistics on those paths. Next, it finds the path with least utilization and pushes essential forwarding rules to OpenFlow switches to redirect traffic to that path.

In order to evaluate the impact of the proposed dynamic flow scheduling on bandwidth, using ipref3 in TCP mode, we generated 10 synthetic and random flows between different hosts in the network. We measured the transmission time and available bandwidth for the flow of interest (given pair of hosts) with or without enabling dynamic multi-pathing. Figure 4 shows a graphical repre-
sentation of network topology detected by ODL User Interface (DLUX) along with the labeled given pair of hosts. Table 2 shows the available bandwidth and transmission time for 700MB of data between the given hosts. By enabling dynamic flow scheduling, transmission time is reduced by 13.6% compared to the static routing method. The average bandwidth was also improved from 35.24Mb/s with no flow management to 40.61Mb/s with dynamic flow scheduling.

Table 2: Transmission time and average bandwidth with and without dynamic flow scheduling

<table>
<thead>
<tr>
<th>Method</th>
<th>Transmission Time (s)</th>
<th>Bandwidth (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Dynamic Flow Scheduling</td>
<td>159</td>
<td>35.24</td>
</tr>
<tr>
<td>With Dynamic Flow Scheduling</td>
<td>137</td>
<td>40.61</td>
</tr>
</tbody>
</table>

The performance gain of the proposed dynamic scheduling is heavily dependent on the rates and durations of the flows in the network. Results also demonstrate the feasibility of building a working prototype of dynamic flow scheduling in CLOUDS-Pi platform.

### 3.2 Virtual Machine Management

Live migration is one of the core concepts in modern data centers which allows moving a running VM between physical hosts with no impact on the VM availability. While VMs in data centers are often migrated between hosts to reduce energy consumption or for the maintenance purposes, live VM migration also provides an opportunity to enhance link utilization and reduce network-wide communication cost. This can be done by relocating communicating VMs to...
hosts with near vicinity and less number of connecting links in the higher layers of the data center network topology.

We measure the bandwidth between two communicating VMs running on physical hosts connected through core switches (Labeled hosts in Figure 4). We run experiments by moving these two VMs closer to each other in the network topology. First, we perform a VM migration to remove any core switches on the connecting shortest path and then another migration to remove both core and aggregation switches. Meanwhile, we investigate the impact of migrations on the available bandwidth.

During the experiment, random and synthetic background traffic is generated between hosts both periodically and constantly. The graph in Figure 5 shows available bandwidth from the source to the destination VM measured with iperf3 tool. Before the first migration (time 0 to 70), the source VM is placed in the host marked as 1 in Figure 4, on a different pod from the destination VM. As shown in the graph, during this period the average bandwidth is roughly 71 Mb/s and fluctuates considerably due to the background traffic generated by other hosts. After 70 seconds, a live VM migration is performed to move the source VM to the host marked as 2 in Figure 4 and connected to a different access switch in the same pod of the destination VM. During the live migration process, the VMs’ networking becomes unavailable for a short period. After this migration, bandwidth becomes less fluctuating and increases to 79Mb/s on average because less number of background flows are conflicting on the traffic of source/destination VM pair. The last migration is done after 170 seconds of the experiment. The source VM is migrated to the host marked as 3 which is connected to the same access switch as the destination VM. With this migration, bandwidth becomes more stable and rises to the average of 86Mb/s where least background traffic affects the networking performance of the VMs.

The results demonstrate the feasibility of building a prototype system allowing network-wide communication minimization in a cloud data centre and open up the research possibility on joint VM and traffic consolidation.

4 Challenges and Future Directions

Our results demonstrate the potential of the CLOUDS-Pi platform in enabling investigation on different aspects of SDN-enabled cloud computing. One of the main benefits of CLOUDS-Pi for conducting research on SDN in cloud computing compared to other prototyping methods or emulators such as Mininet is the option of VM management. VM management and the VM migration possibility, in particular, is an important aspect of cloud computing which allows for VM consolidation to reduce power consumption and overbooking resources to increase cost efficiency. Thanks to OpenStack setup in our platform, we can jointly leverage virtualization capabilities and SDN for performing research on VM and traffic consolidation [11].

Another benefit of CLOUDS-Pi is the possibility of conducting innovative research in traffic control and network management with high accuracy and performance fidelity. As shown in the first use case in Section 3, using ODL northbound APIs and its flow scheduling feature, we can investigate dynamic traffic engineering and load balancing for reducing network congestion with a level of confidence that would otherwise be beyond our reach.
Figure 5: Impact of source VM live migration on the bandwidth of communicating VM pair

Apart from network management, other research directions that can be followed on our testbed platform includes but not limited to Security (e.g., Network intrusion detection [12]), Quality of Service support [13], Service Function Chaining [14], and application-specific networking [15].

The current setup of CLOUDS-Pi has limited number of resources. The number of switches and hosts used in building CLOUDS-Pi platform is far from enough to test the scalability of approaches that need to be deployed in cloud data centers hosting tens of thousands servers and thousands of network switches. In addition, since we are using USB 2.0 ports on Pis with a nominal bandwidth of 480 Mb/s and 100 Mb/s USB to Ethernet adapters as switch ports, our network bandwidth, even though suitable for the scale of our testbed, is much lower than gigabit or terabit networks used in real-world cloud data centers. Although CLOUDS-Pi offers a suitable environment to carry out empirical research into SDN and cloud computing without expenditure of full-size testbeds, we recommend the use of simulators such as CloudSimSDN [16] to evaluate scalability of policies.

Besides our testbed limitations, currently SDN itself faces some challenges such as scalability, reliability, and security that hinder its performance and application. The logically centralized control in SDN causes scalability concerns and especially the controller scalability is one of the problems that needs special attention. The scalability issue of SDN becomes more evident in large-scale networks such as cloud data centers compared to small networks [17]. Controller distribution and the use of east/west APIs is one way to overcome computational load on the controller but it brings consistency and synchronization problems [17]. It also requires standard protocols for an interoperable state exchange between controllers of different types.
The centralized control plane of SDN, for example the single host deployment of ODL controller in our testbed, has a critical reliability risk, such as a single point failure. To tackle this problem, running backup controllers is a commonly used mechanism. However, defining the optimal number of controllers and the best locations for the primary control and the backup controllers is challenging. Synchronization and concurrency issues also need to be addressed in this regard [18].

SDN has two basic security issues which are [19]: 1) the potential for single point of attack and failure and 2) the southbound API connecting the controller and data-forwarding devices is vulnerable to interception and attacks on communications. Even though TLS/SSL encryption techniques can be used to secure communications between controller(s) and OpenFlow switches, the configuration is very complex, and many vendors do not provide support of TLS in their OpenFlow switches by default. SDN security is critical since threats can degrade the availability, performance and integrity of the network. While many efforts are currently being made to address SDN security issues, this topic is expected to draw increasing amounts of attention.

5 Conclusions

We presented CLOUDS-Pi, a low-cost data center environment for SDN-enabled cloud computing. All aspects of our platform from its overall architecture to detailed software choices are explained. We also demonstrated how economical computers such as Raspberry Pis can be used to mimic a network of OpenFlow switches in SDN-enabled cloud data center in scale. In order to evaluate our testbed, two use cases for dynamic flow scheduling and virtual machine management are identified. We discussed benefits and limitations of CLOUDS-Pi as a research platform for different aspects of SDN in clouds and proposed future research directions. Our work demonstrated the potential of CLOUDS-Pi environment for conducting practical research and prototyping SDN-enabled cloud computing environments.

References


