Software-Defined Extreme Scale Networks for Bigdata Applications

Haitham Ghalwash, Chun-Hsi Huang
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT06269, USA
{haitham.ghalwash, chunhsi.huang}@uconn.edu

***Abstract*— *Software-Defined Networking* (SDN) is an emerging technology that supports recent network applications. An SDN redefines networks by introducing the concept of decoupling the control plane from the data plane, thus providing centralized management, programmability, and dynamic reconfiguration. In this research, we specifically investigate the significance of using SDNs in support of Big-Data applications. SDN proved to support Big-Data applications through a more efficient use of distributed nodes. With Hadoop as an example of Big-Data application, we investigate the performance in terms of throughput and execution time for the read/write and sorting operations. The experiments take into consideration different network sizes of a Fat-tree topology. A Hadoop multi-node cluster is installed in Docker containers connected through a Fat-tree of OpenFlow switches. The packet forwarding is either by way of an SDN controller or the normal packet switching rules. Experimental results show that using an SDN controller outperforms normal forwarding by the switches. As a result, our research suggests that using SDN controllers has a strong potential to greatly enhance the performance of Big-Data applications on extreme-scale networks.**

Keywords— Big-Data, Docker, Extreme-scale Networks, Fat-tree, Hadoop, OpenDaylight, Software-defined Networking.

I. Introduction

Networks today are rapidly growing in terms of scale and traffic. According to Cisco White paper, by the year of 2020, the number of devices connected to IP networks will be three times as high as the global population. The annual global IP traffic will reach 2.3 zettabytes, *i.e.* 194 Exabyte per month. Broadband speed will nearly double that in 2015, and the average Internet traffic will increase by twofold [1]. According to what was reported, around 70% of the traffic flows are internal to the datacenter, while 30% external.

The Legacy Three-Tier Data Center Network architectures are no longer able to support such rapid growth of bandwidth requirement and application demands. Many network architectures were proposed to solve such a problem. New network topologies, such as the Fat-tree [2], VL2 [3], DCell [4] and BCube [5] are currently being used. Moreover, traffic engineering mechanisms are also considered as Hedera [6], MicroTE [7] and D3 [8] to accommodate the increasing demand of high traffic. Recently, the integration between the applications and network configurations is a promising trend towards the future networks. Software-Defined Network (SDN) is the new trend of reinventing the network by simply decoupling the data plane from the control plane, thus providing agility, programmability and better abstraction of network applications and entities. SDNs provide a global view of the network that turned out to be promising in improving the performance of network applications including Big-Data applications. On the other side, Big-Data applications can also be applied to optimize the operation of SDNs [9].

This research investigates how SDN can support Big-Data applications traffic running in large scale networks. A Hadoop [10] multi-node cluster is considered as an example of Big-Data application running either in SDN or in normal switching mode. Both operating modes are compared under different network scales. Experimental results record the performance of both operating modes, for read, write, and sort operations, as the network extremely scales up. The paper is organized as follows. In Section II, the SDN architectures for the current and the next generation data centers is presented. The Literature review is presented in Section III. Section IV explores Hadoop and the configured topology under investigation, and Section V discusses the experimental setup and details. The results are presented in section VI. Finally, Section VII has the concluding remarks and future work.

II. Software-defined Networks

Nowadays, Social media, mobile devices, and cloud computing are pushing the traditional network traffic to a crazy limit. As a result, software-defined networking is the promising technology that reinvents the traditional data centers by providing centralized management, programmability, scalability and dynamic reconfiguration. SDN simply decouples the control and data plane of traditional networks' entities. It mainly depends on a centralized controller connected to all network devices, which are, in most cases, communicating using the OpenFlow [11] protocols. The forwarding decisions are flow-based instead of the traditional destination-based forwarding used in today's networks. Moreover, SDN is programmable through applications running on top of the core controller, which interacts with the underlying data plane devices.

According to the Open Networking Foundation (ONF) [12], SDN architecture can be presented in three main layers, namely the SDN Application, SDN Controller, and the SDN Data plane, as shown in figure 1.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

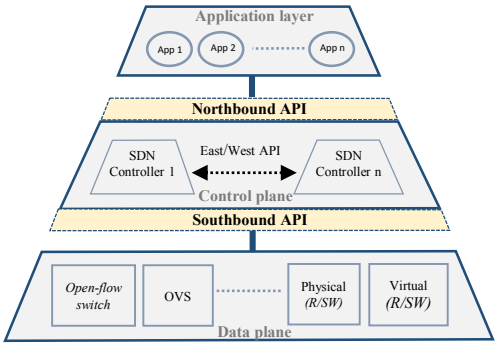


Fig. 1. SDN architecture

On the top is the SDN Application Layer, which contains all programs implementing network requirements and desired behavior. The middle layer is the Control plane, which is responsible for translating the requirements from the SDN Application down to the SDN Data plane. It also provides an abstract view of the entire network for network applications. The bottom layer is the SDN Data plane, *i.e.* the layer of data forwarding entities.

As the network scales up, a single controller may not be able to manage the entire network [13]. This motivated the innovation of several scaling techniques through partitioning, aggregating or even replicating controllers (e.g. Kandoo [14], HyperFlow [15] and ONIX [13]).

III. Literature Review

SDN is the brain of the network, with a global view of the whole network. Having a centralized vision of the network, SDN can directly benefit the distributed requirements of the Big-Data applications by programming the network in the best way to fulfill its needs. The authors of [16] discussed the integration of network control architecture, job scheduling, topology, and routing configuration for Hadoop as a Big-Data application example. Hadoop Map/Reduce jobs, in the SDN network, was presented many times in the literature, by stressing the network in the shuffle and reduce phases. In [17], an SDN-aware version of Hadoop application was set up to use prioritized queues in the underlying OpenFlow switches. As a result, critical Hadoop traffic was first routed before other less critical traffic, so jobs were completed faster. A similar study in [18], proposed a cross point queued (CICQ) switches to schedule packets for different Big-Data applications. The switches schedule packet priorities based on the bandwidth provisioning table that is set by the controller for different Big-Data applications. Another work presented in [17] used Floodlight to assign Hadoop traffic to a high drain rate queue in the OpenFlow-enabled switches. This resulted in more than 30% reduction of job completion time for sorting a large volume of data using Hadoop.

Recently, [19] proposed to run monitoring agents at the data center hosts. These monitoring agents periodically monitor the log files produced by applications like Hadoop and tries to predict the traffic behavior before it starts. A benchmark using a 10 GB Hadoop sort job showed about 35% reduction in job completion time while re-routing only 6% of the flows. Another similar monitoring was HadoopWatch [20] which was a passive agent attached to the Hadoop cluster to monitor the meta-data and logs of Hadoop jobs. The collected data was used to forecast application traffic before the data is sent. HadoopWatch proved to forecast traffic with almost 100% accuracy for small-scale testbeds.

As SDN shows benefits to Big-Data applications, Big-Data can benefit Traffic Engineering in SDN as well. Typical objectives of traffic engineering include balancing network load and maximizing network utilization. In [9], a dynamic traffic engineering system architecture of SDN and Big-Data was described. The SDN controller aggregates and summarizes the collected Big-Data applications traffic information, and sends it to a Big-Data application. The Big-Data application analyzes and gives guidance back to controller traffic manager, which creates the traffic policies and changes the switching behavior accordingly. Another co-operation appeared in [16], where the author proposed an interface between a centralized coordinator of Big-Data applications and an SDN control plane. The interface simply passed the traffic characteristics to the SDN controller before large shuffle phase's start, which helps the controller to setup flows to avoid congestion and enable the application to take better scheduling decision about the jobs. Other work, presented in [21] proposed a time-aware SDN architecture (TASDN). Depending on the requests arrival to the data center, TASDN can coordinate Big-Data applications by introducing a time-aware service scheduling strategy. TASDN scheduled the data center services with different delay requirements and optimized the Big-Data application resources.

IV. Hadoop and Topology Emulation

Big-Data technologies are now storing and processing large data sets varying from terabytes to petabytes and up to exabytes. To efficiently query such size of data, parallel processing spread among multiple cluster nodes, that typically requires high traffic volume to distribute data, is used. The Big-Data application for this paper is Hadoop, which consists of Hadoop MapReduce and Hadoop Distributed File System (HDFS). MapReduce is a programming model and an associated implementation for processing and generating distributed data from a cluster. It contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, with tuples of (key/value) pairs. Reduce receives the output from a Map, then combines these data tuples into a smaller set of tuples. Many Hadoop jobs are communication intensive, involving a large amount of data transfer during their execution. Hadoop Distributed File System (HDFS) mainly consists of a single master node called NameNode, and many slave nodes called DataNodes. HDFS divides the data into fixed-size blocks and spreads them across all DataNodes in the cluster. The NameNode node handles job requests, divide jobs into tasks, and assign each task to a DataNode. The presented scenario basically involves a multi-node Hadoop cluster, a Fat-tree topology and a single controller on a standalone machine. Although a clustered environment may be set using virtual machines, a huge number of CPUs and RAMs is required for such scenario. Therefore, the presented design relies on lightweight software Linux

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

containers, namely, Dockers. Hadoop nodes are implemented on separated Docker containers that are connected in a Fat-tree topology. Fat-tree proved to be promising for supporting large-scale networks, as presented in a previous study [22]. It is widely used in HPC data centers and is typically adopted as two to three levels [23]. A Fat-tree topology normally consists of *n-pods*, each with two layers of n/2 of (*n-port*) switches and $(n/2)^2$ servers. The lower layer of *edge switch* is directly connected to (n/2) server and (n/2) *aggregation switches*. Each *aggregation switch* is further connected with (n/2) *n-port core switches*. There is a total of $(n/2)^2$ *of n-port core switches*, where each core switch has one port connected to each *pod*. The i^{th} *port* of any core switch is connected to pod *i* such that, the consecutive port of aggregation layer of each pod is connected to core switch on (n/2) strides. Table I summarizes the total number of hosts, switches, and links in the Fat-tree over different scales.

The used Fat-tree topology accommodates OpenFlow switches, running OpenFlow1.3 protocol. Each member of the used Hadoop cluster is connected to one of the OpenFlow edge switches as shown figure 2. Mininet 2.2 is used for emulating the network elements such as; host, switches, and links through predefined python scripts. However, the created host, switches and links are real-world elements although they are created by means of software [24]. Mininet allows the use of Docker containers as Mininet hosts by sub-classing the original host class in Mininet. Containernet [25] is a project that integrated a Docker class to be used as hosts in Mininet. All switches, hosts and Docker containers run on a server with 64 GB RAM and 8 core processor. This configuration is later upgraded to be 128 GB RAM and 16 core processor. The final piece for this scenario is the single SDN controller, which is OpenDaylight (ODL) "Boron" release installed on a standalone server with 8 GB RAM and 4 cores.

V. Experiment Configuration

The main objective of this experiment is to study how SDN supports traffic of Big-Data application over different network scales. We are testing the network in two modes of operation, namely, NORMAL mode and ODL mode. In the NORMAL mode, the OpenFlow switches operate on MAC-Port mapping like any other L2 layer switch. OpenFlow switches are configured for NORMAL mode by adding a flow entry for NORMAL mode operation in each of the created OpenFlow switches. For flows hitting this entry, the MAC table maintained by OpenFlow switch would be referred for forwarding decision. We also enabled the *Rapid Spanning Tree Protocol* (RSTP) to avoid loops in our topology. For the other mode of operation, ODL mode, the centralized SDN controller is responsible for setting & managing all flows through all OpenFlow switches. Our SDN controller is OpenDaylight "Boron" release, running on a standalone server, with *"odl-l2switch"* and *"odl-dlux"* features enabled.

We scale our experiment by increasing the switch port capacity starting by 4-port up to 6-, 8-, and 10-port per switch. These sizes allow hosting up to 16, 54, 128 and 250 hosts respectively. The number of switches increases from 20 up to 125 switches from the low scale to the largest 10-port capacity switches, respectively. A three level Fat-tree topology is used for all switch ports capacities, figure 3[a-d] shows the network topological views for different scales, as presented through the ODL web interface. The Hadoop cluster is also scaled up accordingly, starting with a cluster of 8 nodes, up to 18, 32, and finally 50 nodes for the largest scale topology. For each scale, we are using the same file size for different tests of read/write and sort. The Hadoop cluster nodes are organized in such a way, that each node is attached to exactly one edge switch. The packet forwarding in each topology scale is either a NORMAL mode of layer two switching or using the L2 switching in ODL. Table I summarizes different sizes of topology, Hadoop clusters, and the capacity of switches and servers. Figure 2 illustrates how Hadoop nodes participate in the lower scale network.

TABLE I. EXPERIMENT CONFIGURATION

Topology	Total ports	levels	Total SW	edge SW	# max server	Hadoop nodes	File size (R/W/SORT)
	n	K	$(2k-1)(\frac{n}{2})^{k-1}$	$\frac{n^2}{2}$	$\frac{n^3}{4}$		
1	4	3	20	8	16	8	8 x 1GB
2	6	3	45	18	54	18	18 x 1GB
3	8	3	80	32	128	32	32 x 1GB
4	10	3	125	50	250	50	50 x 1GB

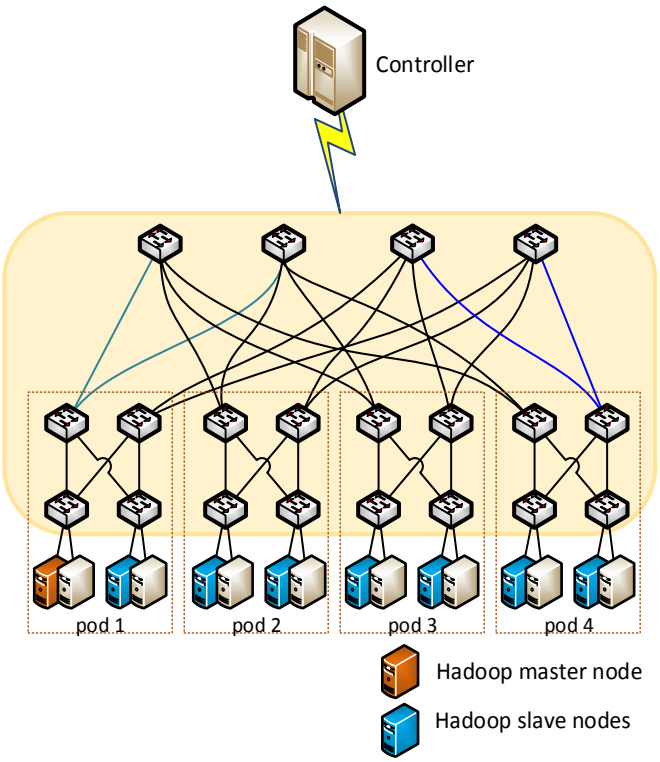
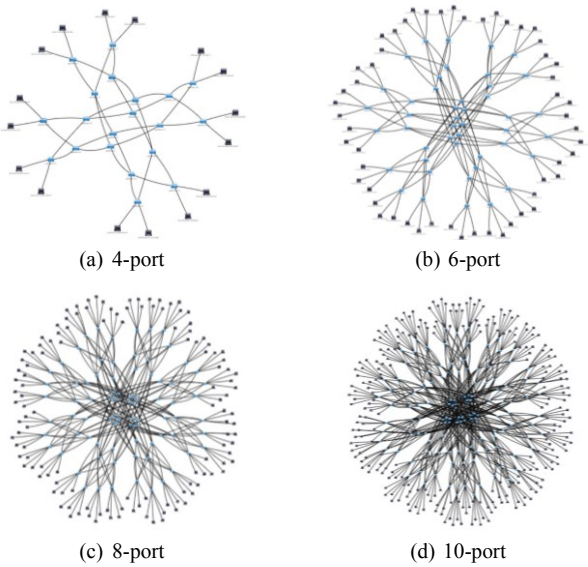


Fig. 2. Fat-tree topology of 4-port switches and the controller



(a) 4-port (b) 6-port

(c) 8-port (d) 10-port

Fig. 3. Different scales of Fat-tree topology

To push traffic into the network, two well-known tools namely, TestDFSIO and TeraSort, for benchmarking single/multi-node Hadoop cluster, are used. The TestDFSIO benchmark is a read/write test for Hadoop file systems (HDFS). It is helpful for the discovery of performance bottlenecks in the network. Throughput in (*Mbps*) and the execution time in (*s*) are the metrics used for performance testing. The TeraSort benchmark sorts any amount of data as fast as possible. It is a benchmark that combines testing the HDFS and MapReduce layers of Hadoop cluster. A full TeraSort benchmark involves, three partial steps as follows: firstly, generating the data, then sorting the data and finally validating the sorted data. The execution time in (*s*) for completing the job is the performance metric for the sorting phase. All experiments for both the TestDFSIO and the TeraSort are repeated 10 times where the average, the minimum and the maximum execution times and throughputs are recorded and presented in figures 4-9.

For the experiments, a server with 64GB RAM and 8 core processor, from an online cloud, is used for the 4-, 6-, and 8-port scales. The server is then upgraded to 128 GB RAM and 16 core processor to handle the creation of the 10-port scale topology, which accommodates a total of 125 switches and 50 Docker containers. The controller is hosted on standalone 8 GB RAM server with 4 cores processor which is upgraded to 16GB RAM and 8 cores for the enhanced largest scale version (10-port scale). It is worth to be noted that our single controller will not be able to maintain the same efficiency level with large-scale networks. A study is conducted to analyze how upgrading or clustering the SDN controller enhances the efficiency of the controller when managing very large-scale networks. An upgraded controller with an added feature, "*odl-mdsal-clustering*", over the single node, is tested for the largest 10-port scale.

VI. SIMULATION RESULTS

As mentioned in Section V, the first test is the read/write using TestDFSIO benchmark. The switching topology is connected as Fat-tee that operates in two different modes, namely, the ODL and the NORMAL modes. The network scale varies from three levels of 4-port switches, accommodating 16 servers and around 20 switches, up to three levels of 10-port switches, with up to 250 hosts capacity and 125 switches. Figures 4-9, show the performance metrics for the read/write and the sorting tests over the various scales; the low scale starting by the 4-, 6- and 8-port switches, up to the largest scale of 10-port. For each scale, we are using the same file size for reading, writing, and sorting operations. For the lowest scale of 4-port, we use an 8GB file size. As the network scales up, the file size is increased to 18GB, 32GB and 50GB for the scales of 6-, 8- and 10-port, respectively. Each metric for each scale is recorded, for both operating modes of the network (NORMAL and ODL).

Figure 4 [a-d], shows the execution times of reading and writing over different scales. The ODL shows better performance over the normal switching mode for all metrics of average, minimum, and maximum execution times. For the write operation, the ODL shows an average execution time improvements of 5.5%, 9.3% and 20% for the low scale of 4-, 6- and 8-port switches, respectively. This improvement is not much noticeable for the largest scale when using 125 switches. The significant large amount of traffic generated from the write/read testing at the 10-port scale is not efficiently handled using a single controller when compared to the lower scales. To achieve a better performance, a more powerful clustered version of the controller is required. Figure 4 (d) shows the performance metrics when upgrading the SDN controller to a higher configuration with clustering feature added for a single node controller. A better performance is achieved for the newly configured SDN controller over the normal forwarding with 7.2% and 8.3% improvements on the average execution time for the write and read operations, respectively.

For the same scales of the network, figure 5 [a-d] illustrates the throughput. The average throughput, using ODL shows an improvement of 10.3%, 18% and 38% over the NORMAL mode for the scales of 4-, 6- and 8-port, respectively. Similarly, the improvement is not noticeable when the highest scale of 10-port switches is considered. With the upgraded version of the controller, the writing average throughput recorded nearly 11.4% improvement for the largest scale, figures 6 & 7 show how scaling up affects the overall execution time and throughput for the read and write operations.

For the sorting test, the execution time for each scale is shown in figure 8[a-d]. Using ODL reduces the average and max execution times over all scales. The improvements on the average execution time are as follows: 2.2%, and 8%, for the 4- and 6-port switch topology, respectively. As we test the larger scale of 8- and 10-port, using ODL still records some improvements over the NORMAL mode. However, the overhead of using a single controller slightly affects the improvement at such scales. A large amount of traffic generated, at these scales, will need more powerful clustered version of controllers to maintain the gradual improvement in the performance. Table II has a summary of the recorded improvement results. Figure 9, shows that average sort execution time improves as the network scales up.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

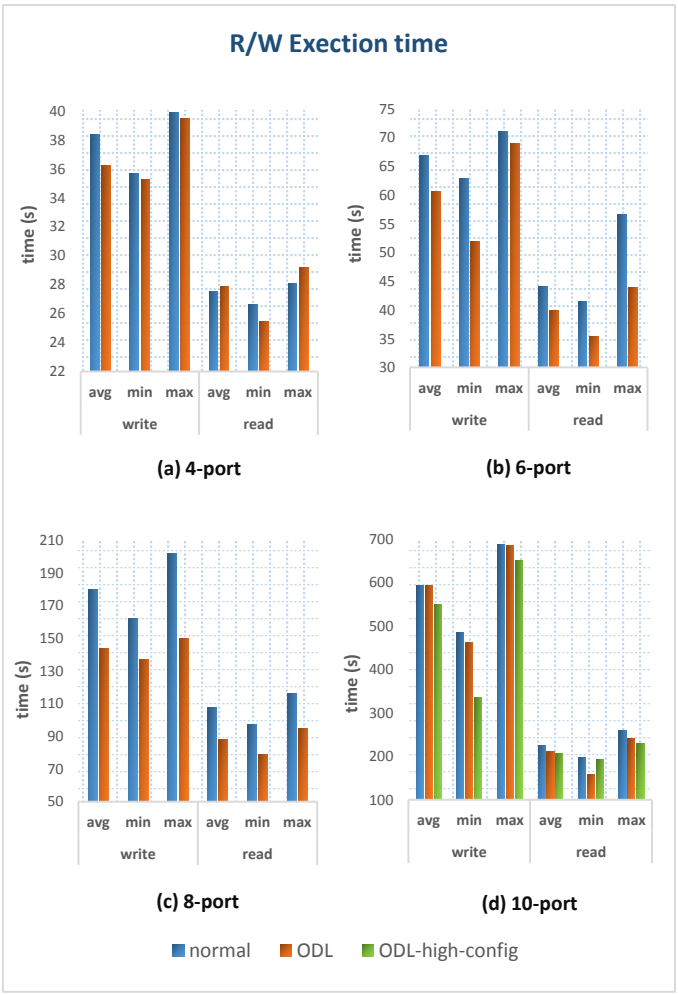


Fig. 4. Execution time read/write

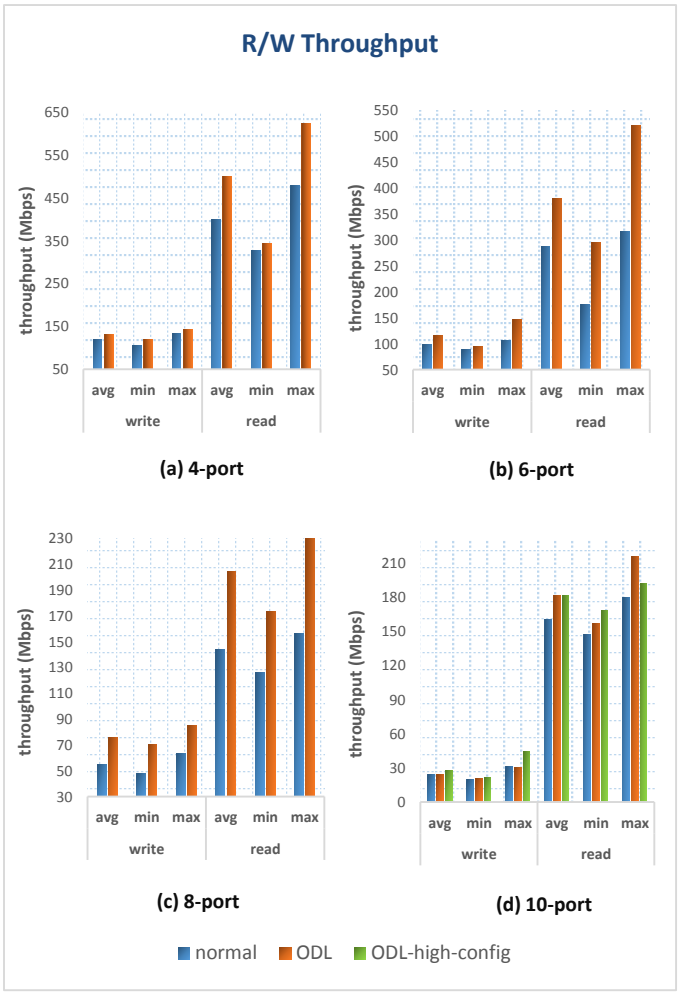


Fig. 5. Throughput read/write

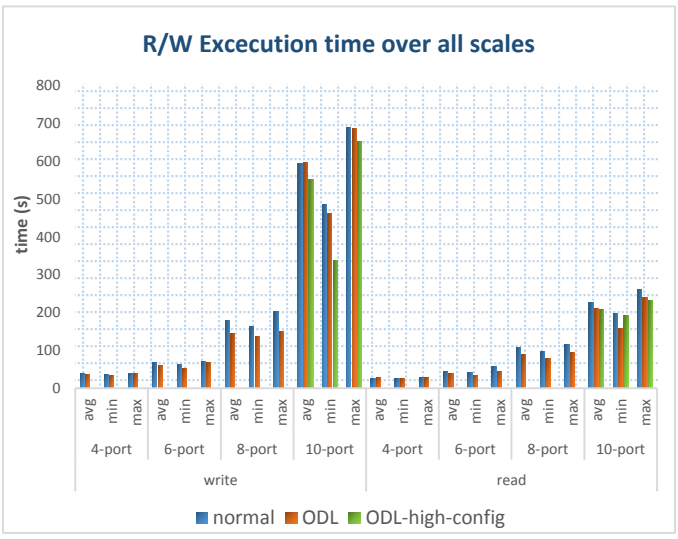


Fig. 6. Execution time read/write for all scales

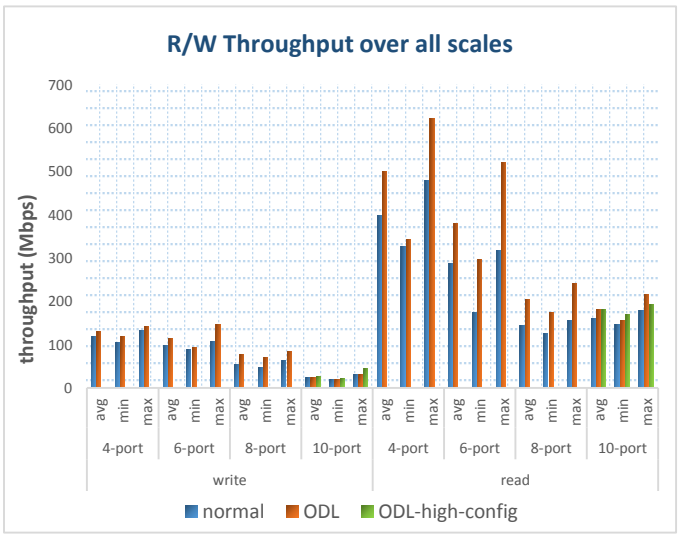


Fig. 7. Throughput read/write for all scales

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

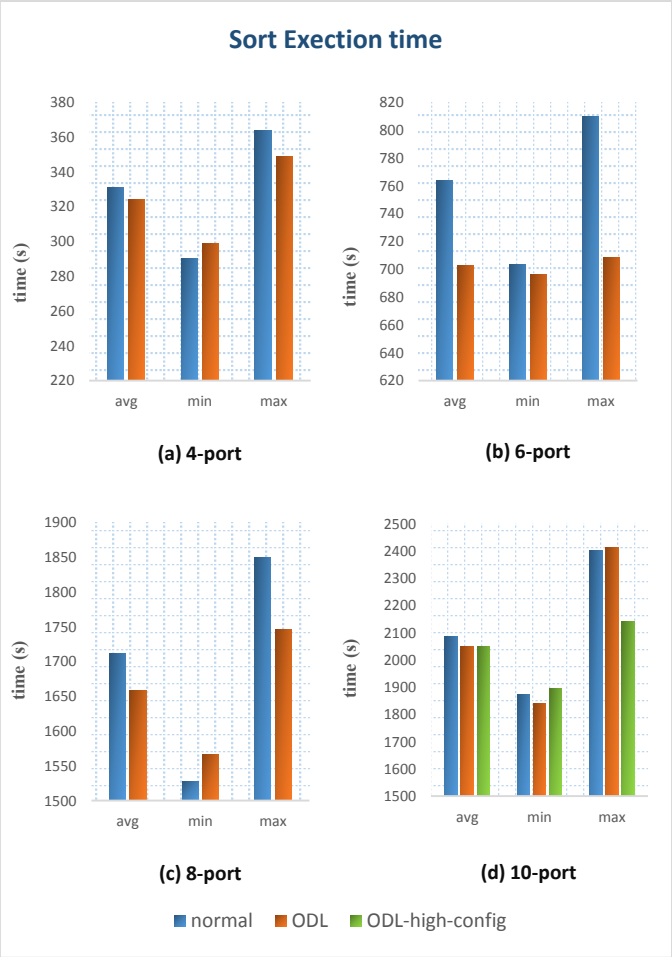


Fig. 8. Sorting execution time

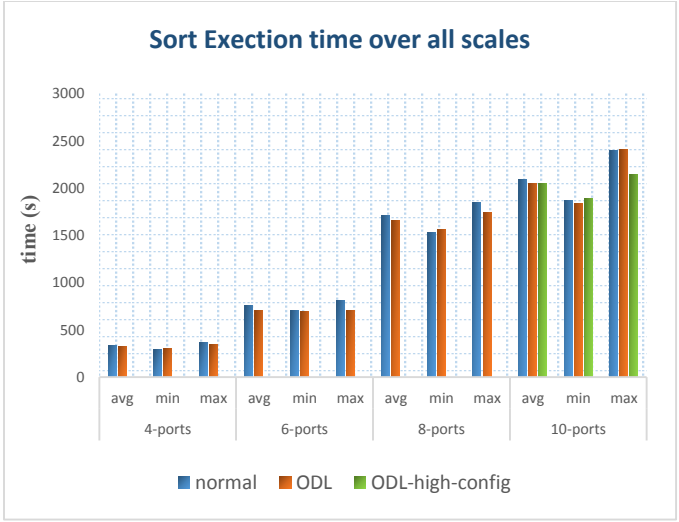


Fig. 9. Sorting execution time for all scales

TABLE II. AVERAGE IMPROVEMENTS

Size (Ports)	Exp.	Execution Time		Throughput		Sort
		write	*read*	*write*	*read*	
4	ODL	5.5%	-	10.3%	25.1%	2.22%
6	ODL	9.3%	9.4%	18.0%	31.9%	7.94%
8	ODL	20.1%	18.0%	38.3%	41.8%	3.11%
10	ODL	-	6.3%	-	13.2%	1.78%
	ODL upgraded	7.2%	8.3%	11.4%	13.0%	1.81%

VII. CONCLUSION AND FUTURE WORK

The presented research investigated the potential of using SDN in Big-Data environment over different scales. The study relied on the widely-used Fat-tree topology that showed promising results for large scale networks. Experiments were conducted by emulating the topology using OpenFlow switches connecting a multi-node Hadoop cluster in Docker containers and the SDN controller is OpenDayLight "Boron" release. The results revealed that SDNs are a promising foundation for more scalable, faster convergence and higher performance networks. Using SDN controller showed a better performance in terms of execution time and throughput for almost all scales of the networks. The Improvement was small over the small scale and gradually showed out as the network scaled up. Using a single controller in OpenDayLight proved to be able to maintain a sustainable performance as the network scaled up. For very large-scale networks, a more powerful and clustered controller needed to be considered to maintain the gradual improvements in network performance. For the future work, we will consider clustering the SDN controller over multiple nodes and consider different groups to be managed via different controllers. Moreover, we need to make use of the global network view provided by the SDN controller in supporting Traffic engineering (TE) for measuring, analyzing and managing the network traffic. Different routing mechanisms will be considered in delivering packets efficiently and enhancing the overall quality of service (QoS).

VIII. ACKNOWLEDGEMENT

This work was supported by the U.S. Department of Education's GAANN Fellowship through the Department of Computer Science and Engineering at the University of Connecticut.

REFERENCES

[1] Cisco Systems, "Cisco Visual Networking Index:Forecast and Methodology, 2015–2020," *white Paper*, San Jose, CA, 2016.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[3] A. Greenberg *et al.*, "VL2: A Scalable and Flexible Data Center Network," *Commun. ACM*, vol. 54, no. 3, pp. 87–93, 2010.

[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008, pp. 75–86.

[5] C. Guo, G. Lu, D. Li, H. Wu, and X. Zhang, "BCube: a high performance, server-centric network architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 63–74.

[6] M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera : Dynamic Flow Scheduling for Data Center

Networks," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI'10)*, 2010.

[7] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, 2011, pp. 8–20.

[8] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better Never than Late: Meeting Deadlines in Datacenter Networks," in *Proceeding ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM'11)*, 2011, pp. 50–61.

[9] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE Netw.*, vol. 30, no. 1, pp. 58–65, 2016.

[10] K. Shvachko, "The Hadoop Distributed File System," *IEEE 26th Symp. Mass Storage Syst. Technol.*, pp. 1–10, 2010.

[11] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[12] O. N. Fundation, "Software-Defined Networking : The New Norm for Networks," *ONF White Paper*, 2012.

[13] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, 2010.

[14] S. Hassas Yeganeh, Y. Ganjali, S. H. Yeganeh, and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceeding HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 19–24.

[15] A. Tootoonchian and Y. Ganjali, "HyperFlow : A Distributed Control Plane for OpenFlow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking (INM/WREN'10)*, 2010.

[16] G. Wang, T. S. E. Ng, and A. Shaikh, "Programming your network at run-time for big data applications," in *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, 2012.

[17] S. Narayan, S. Bailey, and A. Daga, "Hadoop acceleration in an openflow-based cluster," in *Proceedings - 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, SCC 2012*, 2012.

[18] W. Hong, K. Wang, and Y. H. Hsu, "Application-aware resource allocation for SDN-based cloud datacenters," in *Proceedings - 2013 International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA 2013*, 2013, pp. 106–110.

[19] A. Das, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and C. Yu, "Transparent and Flexible Network Management for Big Data Processing in the Cloud," *Present. as part 5th USENIX Work. Hot Top. Cloud Comput.*, 2013.

[20] Y. Peng, K. Chen, G. Wang, W. Bai, Z. Ma, and L. Gu, "HadoopWatch: A first step towards comprehensive traffic forecasting in cloud computing," in *Proceedings - IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 19–27.

[21] Y. Zhao, J. Zhang, H. Yang, and X. Yu, "Data Center Optical Networks (DCON) with OpenFlow based Software Defined Networking (SDN)," in *8th International Conference on Communications and Networking in China (CHINACOM)*, 2013, pp. 771–775.

[22] H. Ghalwash and C. Huang, "On SDN-Based Extreme-Scale Networks," in *High Performance Extreme Computing Conference (HPEC)*, 2016.

[23] M. Bradonjić, I. Saniee, and I. Widjaja, "Scaling of capacity and reliability in data center networks," *Perform. Eval. Rev.*, vol. 42, no. 2, pp. 3–5, 2014.

[24] F. Keti and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," in *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS*, 2015, pp. 205–210.

[25] M. Peuster, H. Karl, and S. Van Rossem, "MeDICINE : Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks*, 2016.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE