Executing Distributed Applications on SDN-based Data Center: a Study with NAS Parallel *Benchmark*

Anderson H. S. Marcondes, Gustavo Diel, Felipe R. de Souza,
Paulo R. Vieira Jr, Adriano Fiorese, Guilherme P. Koslovski
Graduate Program in Applied Computing – Santa Catarina State University – Joinville, SC – Brazil
anderson.marcondes@sfs.ifc.edu.br, {gustavo.diel, feliperodrigodesouza, paulorvj}@gmail.com,
{adriano.fiorese, guilherme.koslovski}@udesc.br

***Abstract*—Software-Defined Networking (SDN) paradigm has decoupled data and control planes on traditional networks. In this context, a logically-centralized controller, with full knowledge on network resources and traffic loads, executes routing algorithms to identify forwarding paths, while networking resources (switches and routers) just work on packet forwarding. This work contributes by executing and characterizing a well-know distributed application, NAS parallel benchmark, atop an SDN-based data center. A comparison with different flow routing algorithms is presented, highlighting that an application can reduce its runtime when applying a multi-path routing.**

***Index Terms*—SDN, HPC, Data Traffic Characterization**

I. Introduction

Maturity achieved by networking technologies, specially in data centers, has consolidated the wide dissemination of data experienced nowadays. Data Center-communication infrastructures are highly required and usually support business operation: data is stored and processed in data centers, and afterwards delivered to users through the Internet. An example is noted in Facebook data centers as up to 33% of an application runtime is consumed by data transfer among servers [1].

Recently, the Software-Defined Networking (SDN) paradigm has decoupled control and data planes on network architectures increasing the range of management options [2]. Such reorganization removes the exclusive control traditionally built-in network equipment, enabling an external control performed by a logically-centralized controller. Indeed, SDN allows the application of routing algorithms with full knowledge on the network topology, as the controller has a combined view of networking resources configuration (switches and routers) and traffic load, executing specialized routing algorithms (the control plan) to identify appropriated forwarding paths. Each decision is propagated to corresponding network equipment, fulfilling flow tables (data plan). In this scenario, the OpenFlow protocol [3] was adopted by the academic community to standardize the communication between devices and controllers.

SDN has been motivating the development of traffic engineering approaches applied to data centers, and triggered revisitation of application-oriented routing. Through traffic characterization of hosted applications, administrators can develop particular routing policies (e.g., load balancing, firewall, multi-path) for optimizing the identified communication profile. Therefore, each application can be individually analyzed by the controller generating flow table entries that meet Quality of Service (QoS) and Quality of Experience (QoE) requirements. It is precisely in this context that the present work contributes: the advantages of using SDN for hosting a distributed application are evidenced by a traffic characterization and by a comparison of routing approaches. A distributed application commonly used to benchmark high performance computing environments was selected for analysis: Numerical Aerodynamic Simulation – NAS Parallel Benchmarks [4]. Each application composing NAS has different networking and processing profiles. In this work, a data traffic characterization of NAS is performed by collecting data from SDN switches. Additionally, three forwarding policies are compared identifying the NAS applications' runtime.

This paper is organized as follows: Sec. II details the characterization of the target application. An experimental analysis of NAS executed atop an SDN-based topology with distinct routing algorithms is detailed in Sec. III. Related work are discussed on Sec. IV, while Sec. V concludes this paper.

II. Caracterization of NAS *Benchmark* in SDN

The NAS benchmark [4] selected as target is composed of several applications with different communication patterns. Moreover, NAS performance is directly related with data-center network configuration and performance.Among the available options for configuring NAS MPI (Message Passing Interface) distribution, the A class applications, indicated for scenarios with limited computing resources, were selected: *block tri-diagonal solver* (BT), *conjugate gradient* (CG), *embarrassingly parallel* (EP), *lower-upper Gauss-Seidel solver* (LU), *multi-grid* solver (MG) and *scalar penta-diagonal solver* (SP).

The experiments were run with Mininet [5], and the experimental scenario was composed of 4 hosts interconnected by an Open vSwitch [6]. The control plane was managed by Floodlight controller [7] with some changes on its default routing algorithm in order to keep the flow status in flow tables even after the application execution. Mininet and Floodlight were hosted on distinct servers.Characterization was accomplished by collecting data directly from the switch. Periodically (each 10 seconds) the data was acquired and consolidated by an external application. Tab. I sums up results, showing the

	# Pkt	Exec. (s)	Pkt/s	Vol. (MB)	Mbps	Pkt (B)
BT	20,569	298.85	68.83	48.04	0.16	2449
CG	62,555	26.98	2,318.57	137.23	5.09	2300
EP	189	123.85	1.53	0.01	0.00	78
LU	385,530	699.48	551.17	496.80	0.71	1351
MG	45,989	24.83	1,852.15	101.24	4.08	2308
SP	722,566	579.67	1,246.51	1,707.55	2.95	2478

TABLE I
NETWORK TRAFFIC OF THE NAS SUITE ON AN SDN TOPOLOGY.

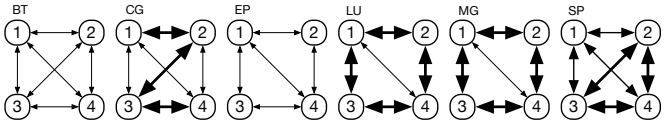

Fig. 1. Communication pattern of NAS benchmark on an SDN infrastructure.

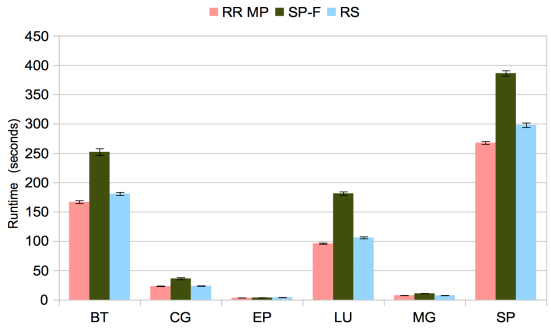

Fig. 2. Runtime of NAS benchmark in an SDN managed fat-tree topology.

average of ten executions with a confidence interval of 95%. For each application, it is informed the number of transferred packets (# Pkt), the total run-time (Exec. (s)), the packets per second rate (Pkt/s), the total data volume (Vol. (MB)), the average bandwidth (Mbps), and the average packet size (Pkt. (B)).

SP was the application with the highest bandwidth usage, with 1.67 GB. EP has a non significant communication dependency, sending a small amount of data, meanwhile CG has the highest number of packets per second rate and the highest average traffic rate, with 2,318.57 pkt/s and 5.09 Mpbs, respectively. Fig. 1 shows flows proportionality for each application. Dense edges represent intensive communication while thinner ones denote low traffic. Data transfer proportionality is analyzed per application, that is, there is no relationship between different scenarios. Virtual host 1 acts as the MPI master node in charge of starting the execution. Consequently, host 1 has connectivity with all remaining hosts (MPI workers). Regarding communication pattern, LU and MG have a ring-based topology, while CG is based on pipelined execution. As described by Tab. I, EP has low data traffic requirements, basically composed of MPI environment initialization messages. Finally, SP pattern comprises an initial data distribution step followed by data-intensive communication among MPI workers. The characterization of NAS applications indicates distinct communication patterns in terms of topology, volume and bandwidth. This analysis highlights the complexity of designing an efficient routing algorithm for distributed applications, as QoS requirements differ even for applications implemented over a single communication tool.

III. ROUTING ALGORITHMS STUDY

Three algorithms were selected for implementation on Floodlight. These algorithms basically differ on how to explore the available paths in a fat-tree network topology [8].

A. Experimental Scenario

The testbed consists of three 4 GB RAM, AMD Phenom II X4 PCs running Ubuntu 14.04, interconnected by 1 Gbps links. One PC executed Floodlight, while the others are running, each one, 8 servers and 10 virtual switches. Switches were created using Open vSwitch and Mininet was adapted to allow its execution comprising multiple physical hosts.

B. Routing Algorithms

1) Shortest Path (SP-F): The default Floodlight routing module forwards packets through legacy or SDN supporting equipment. Route is selected by looking for the shortest path between sender and receiver in terms of hops. Considering fat-tree topology, although there are several paths between a sender A and a receiver B, the deterministic pattern of the Forwarding module always selects the same forwarding path for data packets between A and B.

2) Random Selection (RS): In order to set a baseline to results comparison, a random selection routing module was implemented. Among the multiple existent paths between a sender A and a receiver B, this module randomly selects, at each query, a path to assign to the given flow.

3) Round-robin over Multiple Paths (RR MP): The fat-tree topology allows communication between a sender A and a receiver B using multiple paths between access, aggregation and core switches. To exploit this feature, a routing algorithm that performs a round-robin path selection among the possibilities of existing paths between A and B has been implemented. Paths are sorted in ascending order by the number of hops. In short, for each forwarding request between A and B, the SDN controller identifies the existing paths, responding to the query with a circular algorithm among the candidate paths.

C. Result Analysis

Fig. 2 presents the selected NAS applications' runtime. Results represent an average of ten executions considering a 95% confidence interval. Characterization discussed in Sec. II indicates that EP presented the lowest traffic volume. Thus, due its communication structure (Fig. 1), EP's runtime is only impacted by processing start up and finish messages (composition of the MPI environment). In this case, EP showed equivalent results for the shortest path and round-robin routing approaches. On the other hand, a small overhead was identified for the random paths selection routing approach (about 20%).

MG application sends the second highest packets rate per second (Tab. I) and its communication pattern is a non-blocking ring. As communication occurs between multiple pairs of source and destination asynchronously, the burden of the selection of shortest paths is approximately 29% compared to the random and round-robin path selection algorithms (which present equivalent results considering standard deviation). LU application has the same communication pattern

of MG one. However, a greater volume of data is exchanged while the average throughput is decreased. In this case, the already identified communication pattern is preserved. Thus, the shortest path routing approach is slower than the round robin one (RR MP is approximately 1.8 times faster).

According to Fig. 1, the communication pattern for BT is composed of an uniformly distributed traffic among the nodes with few travelling packets per second. Once more, the multiplicity of paths in the RR MP approach has resulted in lower BT runtime, while the shortest path selection approach showed the bigger one. CG application presents the highest data traffic per second (Tab. I). Its communication pattern can be represented as a pipeline execution (Fig. 1). For this application, the random and round-robin path selection approaches present equivalent and superior results compared to the shortest path one. Finally, SP is the application with the largest volume of data traffic, especially on the work nodes. As in other cases, the approach using round-robin over multiple paths resulted in shorter runtime. Summing up, applications running with RS achieved a better performance compared to SP-F. Indeed, the load balancing of RR MP significantly reduced the runtime of the tested NAS suite applications.

IV. Related Work

Both the characterization of applications and definition of specialized controllers have been addressed in the literature. An extended SDN architecture has been proposed by Mekky and colleagues [9] to allow the inspection of data beyond layers 2 and 4 of the TCP/IP stack. Packets classified as table miss are intercepted between OVS switches [6] and controller. Afterwards, they are analyzed and the forwarding decision is made by consulting a table representing application information. The architecture allows most of the network processing occurs in the data plane, reducing the controller overhead. Egilmez and colleagues [10] have provided an end-to-end QoS multimedia delivery controller, in which QoS requisites were translated as forwarding delay that was minimized.

The Youtube video streaming service was the target of Jarschel et al. [11] to show that the knowledge of the application state allied with the SDN control can improve the user experience when compared to traditional QoS methods. The video buffer information is sent to the controller, responsible for prioritizing traffic when the buffer does not contain enough data to sustain the video quality. On its turn, Watashiba et al. [12] proposed a system for scheduling tasks in clusters exploring the knowledge of the cluster topology to optimize the NAS suite. Each set of the NAS suite represented a user of the cluster. However, the traffic optimization between processes have not used information extracted from applications. An open QoS policy management framework is presented by Bari and colleagues [13]. The framework includes a monitoring module guided by metrics defined by the application administrator. When a dissatisfaction threshold is reached, the controller is triggered to reorganize the data traffic. To accomplish that, it is usually applied circular forwarding on the shortest paths (similar to RR MP discussed in Sec. III).

V. Conclusion

The SDN paradigm introduced the separation of control and data planes, allowing an uniform management of communication devices. In this scenario, a logically centralized controller has full knowledge of the devices. In order to investigate the characterization of distributed applications and the impact of routing algorithms in total runtime, this paper presented a study of NAS suite. Different processing and communication phases where identified, emphasizing the complexity in defining a unique algorithm for routing flows. An experimental analysis compared the NAS applications' runtime using different routing algorithms. Results indicated that the shortest path flow routing approach induces the formation of bottlenecks and consequently increases the applications' runtime. In the target topology (fat-tree), the routing approach that explored the multiple possible paths among the nodes has reduced the runtime of applications with intensive communications.

Acknowledgements

The authors would like to thank to PROMOP/PROAP support as well as to UDESC LabP2D.

References

[1] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 98–109, Aug. 2011.

[2] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber *et al.*, "The NAS parallel benchmarks," *Int. Journal of High Performance Computing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[5] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. of the 9th SIGCOMM Hotnets*. ACM, 2010.

[6] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of Open vSwitch," in *Proc. of the Conf. on Networked Systems Design and Implementation*. USENIX, 2015.

[7] Floodlight, "Floodlight is an open SDN controller," http://www.projectfloodlight.org/floodlight, 2012.

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

[9] H. Mekky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. Lakshman, "Application-aware data plane processing in SDN," in *Proc. of the Third HotSDN*. ACM, 2014, pp. 13–18.

[10] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks," in *Signal Inf. Processing Assoc. Annual Summit and Conf. (APSIPA ASC)*, Dec 2012.

[11] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of YouTube video streaming," in *EU Workshop on Software Defined Networks*, Oct 2013.

[12] Y. Watashiba, S. Date, H. Abe, Y. Kido, K. Ichikawa, H. Yamanaka, E. Kawai, S. Shimojo, and H. Takemura, "Efficacy analysis of a SDN-enhanced resource management system through NAS parallel benchmarks," *The Review of Socionetwork Strategies*, vol. 8, no. 2, 2014.

[13] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "Policycop: an autonomic QoS policy enforcement framework for software defined networks," in *Future Networks and Services (SDN4FNS)*. IEEE, 2013.