EFM: Improving DCNs throughput using the transmission rates of elephant flows

André Luiz B. Rocha, Fábio L. Verdi
Computing Department - DComp
Federal University of São Carlos
Sorocaba, SP, Brazil
debeltrami@gmail.com, verdi@ufscar.br

***Abstract*—One of the major issues in Data Center Networks (DCNs) is related to elephant flows. These flows typically are primarily responsible for network congestion, causing impacts on applications that are intolerant to delays. Although there are several works in the literature that analyze this type of traffic, none of them consider the impact of elephant flow size on the throughput and on the Flow Completion Time (FCT). This article presents the EFM (Elephant Flow Manager) as a complete solution for monitoring and re-routing elephant flows in DCNs, by observing the size of these flows in the network. The tests performed show that the size of the elephant flows influences the throughput and the FCT of the applications and, therefore, deserves to be observed. The evaluation of the solution was performed comparing the joint use of EFM with ECMP (Equal-Cost Multi-path). In all tests, our re-routing solution proved to be better when compared to pure routing solely with ECMP. Then, this paper presents three main contributions: (1) it compares our solution with the ECMP, (2) highlights the importance of the size of the elephant flows in the re-routing of these in DCNs and (3) preliminary tests indicate that due to the decrease in FCT, it is possible to reduce the energy consumption in the data center.**

***Index Terms*—Data Center Networks, Elephant Flows, Re-routing, Monitoring, SDN, OpenFlow, Energy Consumption.**

I. Introduction

Data Center Networks (DCNs) transmit a large amount of data daily, demanding that its architecture is scalable and the routing of the flows has a low computational cost. In order to provide greater scalability, DCNs are constructed based on fat-tree or VL2 [1] topologies. Another feature of the above mentioned topologies is that they have multiple routes with the same cost between a pair of hosts, enabling the use of the Equal-Cost Multi-path (ECMP) routing strategy. Studies conducted in Data Centers state that only 10% of the flows are responsible for more than 80% of the amount of bytes of the entire network. Such flows are termed elephant flows [2].

In order to minimize the impact of elephant flows in DCNs, the most common solutions in the literature are: re-routing them to an specific region of the network as done in [6], [7] or subdividing the elephant flows into mices as done in [3]. Recently, solutions using MPTCP (MultiPath TCP), such as in [4], [5] have been studied to benefit from multiple paths in the network to route flows. However, none of the solutions currently found in the literature consider the size of the elephant flows and its impact on re-routing.

The objective of this paper is to present the EFM (Elephant Flow Manager) as a solution to detect elephant flows in the network, observe their sizes and then take decisions for re-routing. The monitoring performed is responsible for detecting a congestion point and act to prevent negative impacts in the network such as packet loss. Our approach will only work if a certain link's occupation threshold is reached. When such a threshold is reached, our solution will look for elephant flows going through the link and, if any, will identify the biggest and the smallest so that re-routing decisions can be performed. In this work, the concept of size is related to the transmission rate (in bps) of the elephant flows. In addition, the elephant flow will only be re-routed if the threshold is not exceeded after the reallocation of the flow in a new route.

The analyzes carried out in this article evaluate the impact of elephant flow size on re-routing them in the network. The proposal presented here works in conjunction with ECMP. The results show that in all scenarios, the EFM presented results superior to those obtained when only the ECMP is used.

The following sections are organized as follows. In Section 2 we present the related works. In Section 3, the proposed architecture and the interactions between its components are exposed. In Section 4, we present the evaluation of EFM. In Section 5, we discuss about the work.

II. Related Works

The work proposed by Hedera [7] implements two algorithms to perform the scheduling of flows in DCNs: Global First Fit and Simulated Annealing. It is also proposed a traffic estimator, which calculates how much each flow will occupy in the network. The idea of TinyFlow [3] is to use the OpenFlow controller to split an elephant flow into multiple mice and then apply the ECMP to route the flows more efficiently. In Mahout [8], a traffic management architecture with low overhead is demonstrated. To establish low overhead, the detection of the elephant flows is done at the end-host.

The main differences between the works described above and ours are: (1) a flow will be re-routed only if there are congestion points (based on a defined threshold) in the network and the new route is able to receive the flow; (2) identification of the biggest and smallest elephant flows based on their

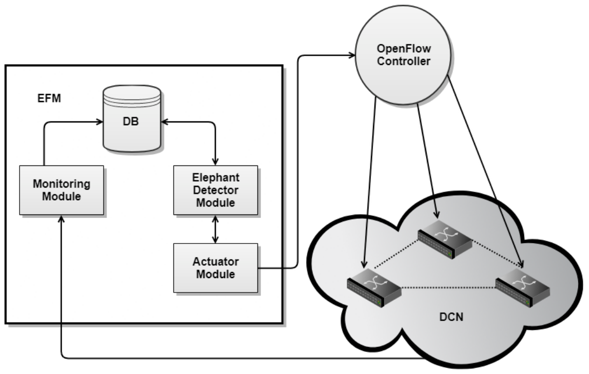

Figure 1: Proposal of the architecture.

transmission rates and subsequent analysis of which of them presented better gains after re-routing.

III. Proposal

Our proposal is to define and implement an architecture that aims to optimize FCT and throughput of DCN flows through the re-routing of elephant flows. The architecture consists of three modules: the Monitoring Module, the Elephant Detector Module and the Actuator Module. Each module is responsible for specific functionalities ranging from monitoring the network to actuation, if there is a possible congestion. In Figure 1, we illustrate the modules that make up our architecture, as well as the interactions between them.

The EFM, more specifically the Monitoring Module, constantly checks the occupancy of each link in the topology and, if it detects a congestion point, the Elephant Detector Module is notified. Such module, then checks for elephant flows at these points. Through the monitoring used in this work, it is possible to measure how many flows are present at the congestion point and their transmission rates. With this information, the module sorts the elephant flows using the transmission rate as the parameter. The ordering of elephant flows is of paramount importance for this work as we are interested in analyzing the efficiency between re-routing the biggest or the smallest elephant flow. Finally, the Elephant Detector Module selects the biggest or smallest flow and notifies the Actuator Module so that a route can be found to receive the flow.

The Actuator Module is responsible for finding a better route for each elephant flow. The "best route" concept in this paper refers to a route where the sum of individual links occupation on a given route plus the placed elephant flow does not exceed the threshold defined for congestion. If this sum exceeds one of the links in the route, a new route should be checked. At the end of the execution, if no route satisfies the described conditions, a new elephant flow is chosen to be re-routed. If a route is found, the flow will be re-routed using the OpenFlow controller.

The complexity of the Actuator Module is equal to the number of possible paths given the source and destination of the flow to be re-routed. This number in a fat-tree topology is equal to $k^2/4$ paths for each pair of hosts that are on different pods, and k is equal to the number of ports in the switches[1].

IV. Evaluation

A. Implementation and TestBed

The three EFM modules presented earlier were implemented with the Java language. The monitoring protocol used was sFlow because it uses the push method and collects statistics by sampling packets. In this work, we choose the sFlow-RT [2] tool collect DCNs traffic metrics. The Floodlight OpenFlow controller was chosen to control the network elements. Finally, we use Neo4J [3] to store the topology graph and the CYPHER language to make queries on it.

We validate our proposal in an emulated virtualized environment. For this, we use the Mininet network emulator with Open vSwitches. We instantiate a fat-tree topology with 20 Open vSwitches, 5 hosts and k equal 4. Thus, the used fat-tree topology has three levels containing 20 switches (4 core, 8 aggregation and 8 edge).

The topology links have been configured to operate at a maximum rate of 10 Mbps. Therefore, in this work a flow is considered elephant if its transmission rate exceeds 10% of the total capacity of the link, that is 1 Mbps. We define the congestion threshold at 70% of use of a given link, whereas, if it exceeds 7 Mbps of usage, the EFM should act. The pushing frequency set in the sFlow agent was 1 second and the packet sampling rate was 1 in every 10 packets.

B. Methodology

The EFM evaluation was done taking into account the FCT (Flow Completion Time), the throughput and the power consumption. In every evaluation, our tests compared two approaches: the first one used sole ECMP for flow routing (label ECMP). The second approach uses the EFM in conjunction with ECMP. In this last case, EFM was exercised in two ways: re-routing the biggest elephant flow (label Biggest) and re-routing the smallest elephant flow (label Smallest).

The collisions that can be minimized in DCNs are those known as Upstream and Dowstream collisions detected in the core switches. Therefore, the scenarios presented in this paper reproduce those collisions in all the tests. To adjust these collisions, we use the Iperf software to generate the elephant flows in the network. To evaluate the FCT and the throughput we performed four tests with four variations of the transmission rate in each test. Each test was performed five times and the average among them was obtained.

C. Results of upstream and downstream collisions

We created three flows on Tests 1 and 3, four flows in Tests 2 and 4. We vary the transmission rates four times for each flow of all tests, there flows are TCP and transmited 20 MBs. The Flow 4 (F4), apresented in Tests 2 and 4, aims to

[1]Assume that all DCN switches have the same number of ports.

[2]InMon Corporation. sFlow-RT Page, http://www.inmon.com/products/sFlow-RT.php, 2004

[3]Neo4J Company. Documentation of Neo4J, https://neo4j.com/docs/, 2016.

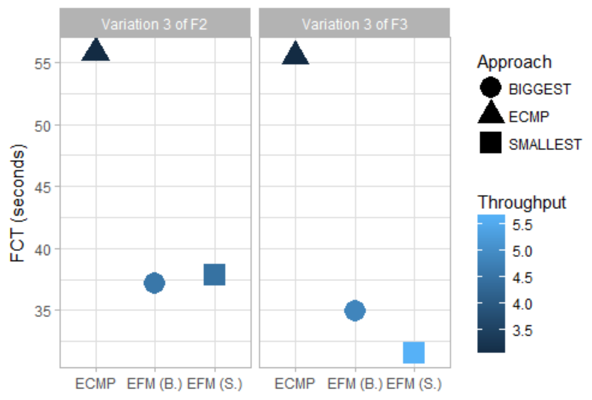

Figure 2: Test 1 Upstream Collisions.

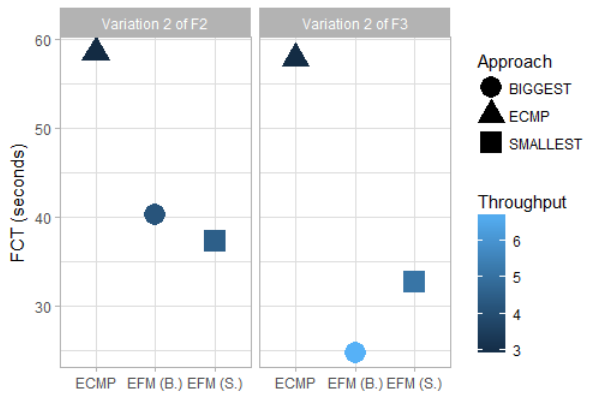

Figure 4: Test 3 Downstream Collisions.

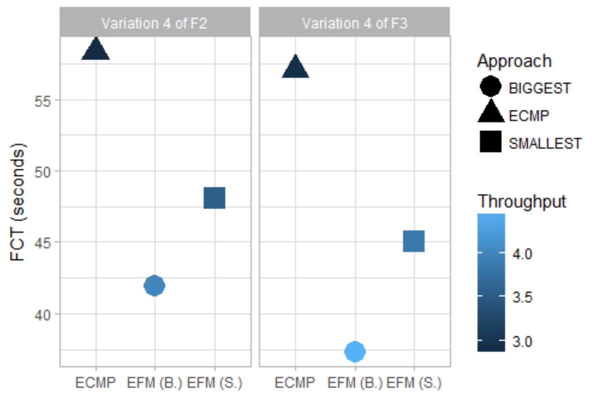

Figure 3: Test 2 Upstream Collisions.

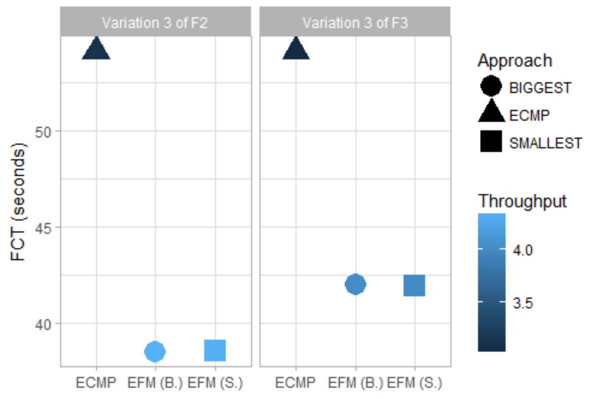

Figure 5: Test 4 Downstream Collisions.

insert background traffic in the network. The objective of the variations is to contemplate a larger number of scenarios in order to analyze the impact of the size of elephant flows after re-routing. In summary, we have four tests with three (Tests 1 and 3) or four flows (Tests 2 and 4) and four variations on transmission rates of all flows. Each variation was executed five times and we collected the average of this values.

In this paper, due to space limitations, only few graphs obtained from the four tests will be presented. The Figs. 2 and 3 present a sample of all tests performed. We can observe that in both graphs the EFM obtained better results in comparison to ECMP. It is noteworthy that in most variations of all three measured flows re-routing the biggest elephant flow was better than re-routing the smallest elephant flow. However, there were cases in which re-routing the smallest elephant flow was better as for example in Fig. 2, variation 3 of Flow 3 (F3).

The results for Tests 3 and 4 are shown in Figs. 4 e 5. Again, both EFM approaches were superior to sole ECMP. In most of the variations of Test 3 the results were slightly better when re-routing the smallest elephant flow. Meanwhile, in Test 4 the results between the EFM approaches did not present a significative difference.

Table I summarizes all the results obtained and demonstrates how EFM biggest and EFM smallest were superior to sole ECMP in all variations of the flows in each test. Analyzing the results obtained, we can say that there is a predominance when re-routing the biggest elephant flow in relation to re-routing the smallest elephant flow. However, there have been cases where the opposite could be observed. Such information can and should be explored so that the FCT and throughput of the flows are improved, optimizing the DCN as a whole.

D. *Results for power consumption*

We performed two preliminary tests to measure power consumption in the hosts, comparing sole ECMP with EFM + ECMP. The test environment was similar to the one used in the tests described above, the only difference is that one of the hosts was replaced by a physical machine instead of using a Mininet VM. In order to collect the power consumption information of the hosts, the RAPL [9] tool was used. In

Table I: Percentage of gain all results compared to ECMP.

	BIGGEST	SMALLEST
Test 1	36.53 %	32.23 %
Test 2	36.78 %	16.71 %
Test 3	52.44 %	54.43 %
Test 4	29.1 %	24.13 %

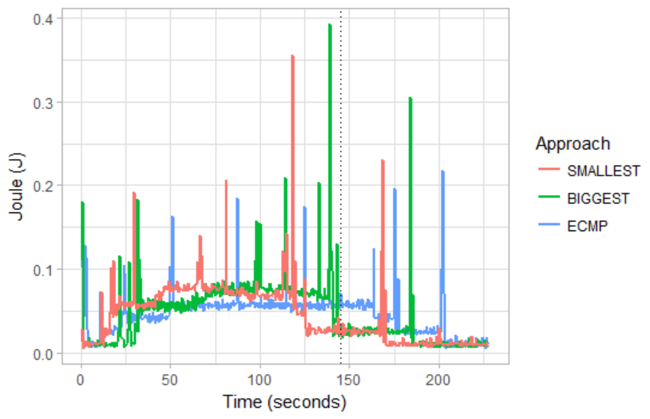

Figure 6: CPU Core Power Consumption from Test 1.

this work only the power consumption of the motherboard and CPU cores were considered. Futhermore, the labels that define such metrics are subsequently defined as Motherboard and CPU cores.

The two tests performed contemplate upstream and downstream collisions. We chose to collect the power consumption information from only one of the hosts in each test, in this case one of the source hosts of the flows. Each test run 5 times for each approach, as in the previous tests and then we calculated the power consumption of the motherboard and CPU cores on an interval of 0.5 seconds.

The results obtained for Test 1 are displayed in Fig. 6. In the y-axis it is possible to observe the power consumption in Joules (J), whereas in the x-axis we visualize a time line which illustrates the beginning and the end of the traffic in an interval of 0.5 seconds. In this graph, we can analyze that generally, the power consumption running only the ECMP is slightly lower than both EFM solutions. However, from the instant 145 the ECMP retains the same power consumption while the power consumption using EFM decreases. This happens because when using EFM the flows terminate earlier due to the decrease in their FCTs. As a consequence, power consumption in the host also decreases as the flows end. Therefore, the time difference between the termination of a given flow with ECMP and with ECMP + EFM is also the time when the power consumption of our solution will be lower than the power consumption using solely ECMP.

We conclude that due to the decrease of FCT through the performance of EFM, it is also possible to reduce the power consumption in DCs. Table II illustrates the average power consumption (in Joules) of the two tests executed. We observed cases where the EFM was more economical than the ECMP and cases where the opposite occurred. The power savings will always be proportional to the gain obtained with respect to the FCT, comparing the EFM + ECMP with sole ECMP. If the gain in the FCT is not significant the power consumption will remain the same. However, if FCT shows interesting gains, the trend in decreasing power consumption may be advantageous.

Table II: Average power consumption in Joules from Tests 1 and 2.

	ECMP	BIGGEST	SMALLEST
Test 1 - CPU Core	0.04718	0.04837	0.04062
Test 1 - Motherboard	2.86575	2.87505	2.85102
Test 2 - CPU Core	0.08279	0.07823	0.07961
Test 2 - Motherboard	2.97591	2.95849	2.97840

V. Conclusion

In this work, we present the EFM whose objective is to optimize the FCT and the throughput of the flows in DCNs. The EFM works in conjunction with ECMP and uses the transmission rate of the elephant flows to decide which stream to re-route (the biggest or the smallest).

We observed that it is useful to know the current transmission rate of the elephant flows present in the network, in order to maximize the throughput and to decrease the FCT of the flows through the re-routing. Our architecture proved to be efficient when compared to sole ECMP, using both re-routing options, acting on the biggest or smallest elephant flows. In addition, preliminary tests illustrate a decrease in energy consumption proportional to the gain obtained in the FCT when using the EFM + ECMP. Since the flows re-routed with EFM will finish earlier when compared to the flow routed by ECMP, this time difference reflects the decrease in the power consumption of the data center as a whole.

Acknowledgment

This work was funded by a research grant from São Paulo's Research Foundation (FAPESP:15/19766-9) and National Council for the Improvement of Higher Education (CAPES).

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, commodity data center network architecture. In Proc. ACM SIGCOMM, 2008.

[2] Benson, Theophilus, A. Akella and David A. Maltz. Network traffic characteristics of data centers in the wild. ACM SIGOMM 10th, 2010.

[3] H. Xu and B. Li. TinyFlow: Breaking elephants down into mice in data center networks. Local Metropolitan Area Networks (LANMAN), 2014 IEEE 20th International Workshop on, 2014.

[4] J. Zhao, J. Liu, H. Wang and C. Xu. Multipath TCP for Datacenters: From Energy Efficiency Perspective. The 37th IEEE International Conference on Distributed Computing Systems, 2017.

[5] A. Silva, F. L. Verdi. Empowering Applications with RFC 6897 to Manage Elephant Flows in Datacenter Networks. IEEE International Conference on Cloud Networking (Cloudnet), 2017, Prague, Czech Republic.

[6] R. Kanagevlu and K. M. M. Aung. SDN Controlled Local Re-routing to Reduce Congestion in Cloud Data Center. 2015 International Conference on Cloud Computing Research and Innovation (ICCCRI), 2015.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. Proceedings of the 7th USENIX conference on Networked systems design and implementation, 2010.

[8] A. R. Curtis and W. Kim and P. Yalagandula. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. IEEE INFOCOM, 2011.

[9] Desrochers, Spencer and Paradis, Chad and Weaver, Vincent M. A Validation of DRAM RAPL Power Measurements. ACM, 2016.