Fincher: Elephant Flow Scheduling based on Stable Matching in Data Center Networks

Yuxiang Zhang, Lin Cui and Qiao Chu
Department of Computer Science
Jinan University
Guangzhou, China
Email: samuelzyx0924@gmail.com, tcuilin@jnu.edu.cn, logan_chu@outlook.com

***Abstract*—With the development of cloud computing in recent years, data center networks have become a hot topic in both industrial and academic communities. Previous studies have shown that elephant flows, which usually carry large amount of data, are critical to the efficiency of data centers. In this paper, we study the flow scheduling problem in data centers with a focus on elephant flows. By applying stable matching theory, the scheduling problem is modeled and some useful method is complemented. Then, we propose *Fincher*, an efficient scheme leveraging Software-Defined Networking (SDN) to reduce latency and avoid congestions in data centers.**

I. Introduction

Driven by modern Internet applications and cloud computing technologies, data centers are being built dramatically around the world. To obtain high bandwidth and achieve fault tolerance, Data Center Networks (DCNs) are often designed with multiple paths between any two servers[1][2]. Multiple paths are not only useful in dealing with potential failures, but can also be exploited to enhance network performance. Especially, when the traffic load get large, the scheduling of any pairs of servers communication becomes more important.

Grouping data into flows helps researchers to get a better view of data transmissions in data centers. Flow scheduling solves the reordering problem introduced by multi-path routing. The data packets in the same flow are transferred sequentially without being reordered. And when a flow has a lot of data or lives a long period, we call it *elephant flow*. Otherwise, it is referred as *mice flow*. Elephant flows scheduling is a challenging problem in data centers, which can cause congestion and occupy large resource. In previous years, many research works and solutions have been proposed to schedule elephant flows[1][2].

Among all solutions, Equal Cost Multi-Path routing (ECMP) is the state-of-the-art for multi-path routing and load-balancing in DCNs. However, ECMP, which is based on flow hashing, would cause congestion when hash collisions occur and perform poorly in asymmetric topologies[3].

Based on fat-tree topology, Al-Fares et al. [1] propose a centralized flow scheduler, which schedules elephant flows with assigning a path without reserved path to each elephant flow. However, it assumes that at most one elephant flow originates from one host at a time, which is impractical in current data centers. Hedera, another centralized flow scheduling system for fat-tree, is proposed in [2]. Hedera detects large flows at edge switches, and selects a path according to the result of the estimated demand of large flows. However, due to only considering bandwidth limitation, Hedera may cause unbalanced traffic or congestions.

In this paper, we use stable matching theory to handle the scheduling problem for elephant flows in data centers. According to the characteristic of stability, we present *Fincher*, a novel stable matching based elephant flow scheduling algorithm, which provides high utilization and low latency in data centers. Fincher leverages the global information of data centers to generate the preference list of flows and switches. By obtaining an optimal stable matching between flows and switches, Fincher can match all flows to their appropriate paths and achieve optimal performance of the network. The stability of output matching enables Fincher to avoid congestion, which are caused by elephant flows, effectively. We have preliminary evaluation for Fincher in Mininet and Fincher's performance is better than ECMP and Hedera in flow completion time (FCT).

II. Fincher Scheme

In data center networks, there are two main objects needed to be considered carefully when performing flow scheduling : 1) Switch: a set of switches which can receive packets, store packets in memory and forward packets; 2) Flow: a set of flows which transfer data between two end hosts. Flows occupy switches memory and demand for resources. In practical data center, fat-tree is commonly implemented. A k-pod fat-tree topology is considered, which means each switch has k ports. For inter-pod flows, there are $(k/2)^2$ possible paths between any given pair of hosts in the network [1]. Each of these paths corresponds to a unique core switch. Thus, when a new flow arrives, we need to assign an appropriate core switch to this flow and this chosen core switchs corresponding path becomes flows chosen path. A similar operation can be performed for intra-pod elephant flows, i.e., assigning a unique aggregation switch to represent the path for intra-pod flow.

The description above reminds us the Stable Matching Theory which originates from SM (Stable Marriage) problem[4]. We want to apply the stable matching theory in elephant flow

Corresponding Author: Dr. Lin Cui (tcuilin@jnu.edu.cn)
This work is supported by the "Fundamental Research Funds for the Central Universities" No.21614330 and NSFC No. 61402200.

978-1-4673-8590-9/15/$31.00 ©2015 IEEE

scheduling to solve the confliction between switches and flows to avoid congestion. First, we should build a preference list for both flows and switches. In this paper, we focus on data rate and unoccupied memory as the main factor of both flows' and switches' preference lists. Each flow has a preference list of switches $P(f_i) = \{s_1, s_2, \ldots, s_k\}$, where each switch can forward the f_i and its capacity is sufficient $(c_j > r_i)$, the k is the number of these switches. The order of the preference list can be determined by the descendant order of the length of switches unoccupied memory. Each switch has a preference list of flows $P(s_i) = \{f_1, f_2, \ldots, f_k\}$, where each flow can be transferred to the switch and its data rate does not exceed the switch's capacity$(c_j > r_i)$, where k is the number of these flows. The sequence order of this list can be determined by the descendant order of the data rates of flows. Moreover, to avoid congestion, switches' memory can not be overflowed. It means that switch should follow the constraint: a switch can accept multiple flows as long as the total flow data rate does not exceed its capacity.

Under the preference lists for both flows and switches with the switchs memory occupancy constraint, our objective is to find an optimal matching between flows and switches, in which for each flow, no better switch can receive it, and for each switch, no larger flow is rejected when it still can accept flows.

To solve the conflict between flows and switches, and find stable matching between them, we present our modified Gale-Shapley algorithm[5], which we called Fincher algorithm: elephant flow scheduling based on stable matching in data center networks. Fincher is guaranteed to find a stable matching for a given problem. The key idea is to ensure that, whenever a flow is rejected, any less preferable flows will not be accepted by a switch, even if it has enough capacity to do so.

When we implemented our scheme, we found that directly apply DA (Deferred Accept) algorithm in flows scheduling may cause congestion and it would output a greed matching. In order to overcome this challenge, we set a threshold for each switch which can balance load among switches as evenly as possible. And our threshold's definition is as follow.

$$T = \frac{\sum c_j - \sum r_i}{|S|} - \text{ranked } (|F| + 1 - |S|)\text{flow's data rate} \tag{1}$$

We believe that the definition of threshold like above can assure no free flows which no switch can accept it and avoiding greedy matching.

III. Preliminary Evaluation

We have evaluated the performance of Fincher in a relatively large scale with higher path diversity on a Mininet[6] and POX platform. Fincher algorithm is running on the POX controller module, which is connected to the Mininet through a LAN. We use the workload with traffic patterns provided in [1] and Hedera [2]. We compare our Fincher scheme to ECMP and Hedera. We use flow completion time (FCT) as the performance metric throughout the evaluation.

Figure 1 shows each elephant flows average FCT of ECMP, Hedera and Fincher in our experiment. From the Figure 1 we

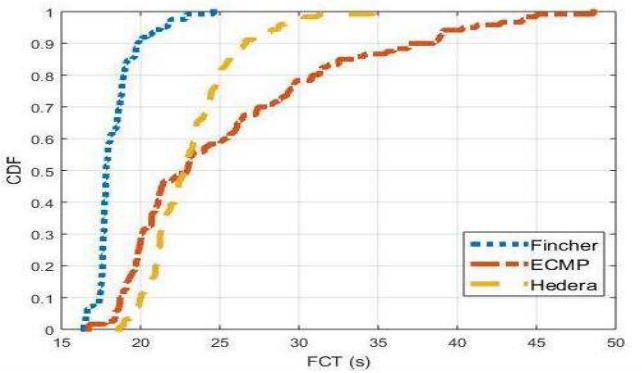

Fig. 1. CDF of FCT

can know that Fincher achieve reduction in FCT effectively. By calculating the reduction rate in FCT, we know that Fincher achieves 14% to 35% reduction in the FCT of elephant flow compared to ECMP and average 29% reduction in the FCT of elephant flow, meanwhile, Fincher achieves 10% to 28% reduction in the FCT of elephant flow compared to Hedera and average 21% reduction in the FCT of elephant flow. The effectiveness of avoiding congestion and load balancing contributes to the effectiveness of Fincher in scheduling. The experiment result also proves that stable matching is a strong method in flow scheduling.

IV. Conclusion

In this paper, we studied the challenge of elephant flow scheduling problem and proposed a solution, Fincher, by applying stable matching theory. Through finding a stable matching between elephant flows and core/aggregation switches, Fincher can effectively improve the performance of large flow and avoids congestion. Preliminary Evaluation shows that Fincher's effectiveness on utilizing the capacity of the network and reducing the completion time of the flows. Future work is needed to improve the performance of the algorithm and a further study of the data center network.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 63–74.

[2] 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*, 2010, pp. 19–19.

[3] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 465–478.

[4] K. Iwama and S. Miyazaki, "A survey of the stable marriage problem and its variants," in *Informatics Education and Research for Knowledge-Circulating Society, 2008. ICKS 2008. International Conference on*. IEEE, 2008, pp. 131–136.

[5] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *American mathematical monthly*, pp. 9–15, 1962.

[6] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 253–264.