On the Efficacy of Fine-Grained Traffic Splitting Protocols in Data Center Networks

Advait Dixit
dixit0@cs.purdue.edu

Pawan Prakash
pprakash@cs.purdue.edu

Ramana Rao Kompella
kompella@cs.purdue.edu

Department of Computer Science
Purdue University

ABSTRACT

Multi-rooted tree topologies are commonly used to construct high-bandwidth data center network fabrics. In these networks, switches typically rely on equal-cost multipath (ECMP) routing techniques to split traffic across multiple paths, such that packets within a flow traverse the same end-to-end path. Unfortunately, since ECMP splits traffic based on flow-granularity, it can cause load imbalance across paths resulting in poor utilization of network resources. More fine-grained traffic splitting techniques are typically not preferred because they can cause packet reordering that can, according to conventional wisdom, lead to severe TCP throughput degradation. In this work, we revisit this fact in the context of regular data center topologies such as fat-tree architectures. We argue that packet-level traffic splitting, where packets of a flow are sprayed through all available paths, would lead to a better load-balanced network, which in turn leads to significantly more balanced queues and much higher throughput compared to ECMP.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Performance

Keywords

Data centers, traffic splitting

1. INTRODUCTION

To scale the data center network to provide the level of connectivity required, most data center network fabrics are organized in the form of a multi-rooted tree topology. Further, they use multipathing between servers so that load is balanced among several alternate paths and the chances of congestion are reduced. One popular multipathing mechanism used in data centers today is equal-cost multipath (ECMP). Many recent works have identified the load imbalance that can arise due to ECMP and suggested different approaches for making the load more balanced across the different available paths.

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

We revisit the conventional wisdom that packet-level traffic splitting (PLTS) is inherently harmful to TCP. Specifically, our observation is grounded on the fact that many popular data center network designs such as the fat tree, or more generally, multi-rooted tree topologies are symmetric in their architecture, and by spraying packets across different paths leads to a more balanced and predictable network architecture, that interacts well with TCP.

We propose different variants of the per-packet load balancing algorithms in this paper that exploit the basic idea, but vary depending on the amount of state each of the solution maintains in the routers. Using simulations, we compare these various strategies in terms of TCP throughput achieved. We also compare them against ECMP on various network parameters like latency, fairness etc. Specifically, we show that per-packet load balancing outperforms ECMP in all the dimensions—1.5× better throughput and 6× better latency at the 99^{th} percentile.

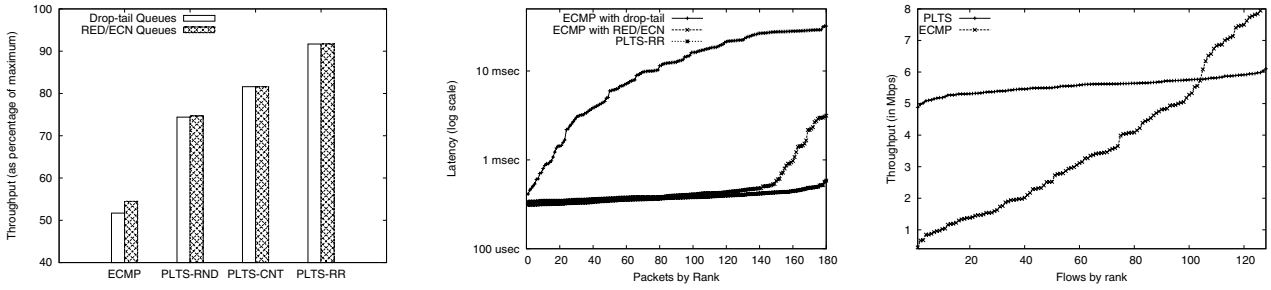
2. PACKET-LEVEL TRAFFIC SPLITTING

We use three simple techniques—random, counter-based and round-robin—to demonstrate the power and limitations of PLTS. All these techniques essentially split a flow across all available paths, but differ based on the amount of state maintained at the switches. Random picks an arbitrary port at random (among the multiple next hops) to forward a packet, and hence requires no state to implement. At the other end of the spectrum, we discuss a per-flow round robin technique where packets within a flow are transmitted in a round robin fashion. However, it requires remembering the last port used on a per-flow basis making it more complicated than the random.

In addition to the above, we also propose a new algorithm that splits traffic based on local port counters. This reduces the amount of state required significantly, but also results in slightly worse load balancing in the network (as we shall see) and consequently, a slight loss in throughput. While this list of algorithms is not meant to be exhaustive, we chose these three schemes as potential candidates since they can be implemented in a rather straight-forward fashion.

3. EVALUATION

Most of our experiments in this paper are based on simulations using QualNet , a packet-level discrete event simulator. For evaluating PLTS, we focus mainly on fat-tree [1] architecture. However, the results shown should hold good in many topologies with multiple equal-cost paths between end hosts.



(a) Throughput (b) Latency comparison (c) Fairness among flows

Figure 1: Throughput, latencies and fairness achieved among flows.

For this evaluation, we simulate a 6-pod fat tree topology with 54 end-hosts and create 27 FTP applications which run for the entire duration of the simulation. The end hosts for each FTP application are chosen at random while ensuring that each end host executes either one FTP server or client. Since a fat tree topology offers full bisection bandwidth, ideally each FTP application should be able to transfer data at full link capacity.

Figure 1(a) shows the average throughput observed by FTP applications, under different schemes, as a percentage of the ideal throughput.

All the three PLTS techniques achieve better throughput than ECMP as shown in Figure 1(a). [4] reports that MP-TCP achieves almost 90% utilization for the same experimental setup, which is comparable to what the PLTS-RR achieves. Among packet-level traffic splitting, PLTS-RND attains the least throughput because skews in short sequences of random numbers increases variability in latencies across different path. PLTS-CNT attains around 81% of the ideal throughput. PLTS-RR ensures that all the paths between source and destinations are utilized. Figure 1(a) confirms this behavior and PLTS-RR is able to achieve almost 92% of the ideal throughput.

The latencies for the sampled packets are ranked and plotted in figure 1(b). PLTS with drop-tail policy shows significantly better packet latency values as compared to ECMP with RED/ECN. A packet in PLTS experiences only one-third of the packet latency of ECMP with RED/ECN at the 90^{th} percentile and one-sixth of the packet latency at the 99^{th} percentile. In PLTS, flows achieve better TCP throughput on an average as compared to flows with ECMP forwarding. Figure 1(c) shows the throughput observed by all flows. Flow throughputs for ECMP show a huge variability as compared to those of PLTS. There is an order of magnitude of difference between the highest and lowest throughput seen by ECMP.

4. RELATED WORK

The most related to our work are those mechanisms that rely on flow-level traffic splitting such as ECMP, Hedera [2] and Mahout [3]. Techniques like Hedera and Mahout, which select a path for a flow based on current network conditions suffer from a common problem: When network conditions change over time, the selected path may no longer be the optimal one. VL2[5] propose using Valiant Load Balancing (VLB) at a per-flow granularity, but they too do not split an individual flow across multiple paths.

Two research efforts propose traffic splitting at a sub-flow granularity. MPTCP [4] splits a TCP flow into multiple flows at the end hosts. FLARE [6] exploits the inherent burstiness of TCP flows to break up a flow into bursts called flowlets. We did experiment with some simple variants of FLARE, such as forwarding a small number of consecutive packets of a flow along the same path. But we observed that, since bursts of packets may actually lead to disparity in queue lengths across different paths, they cause much more packet reordering and reduction in throughput.

5. CONCLUSION

We show that simple packet-level traffic splitting mechanims can yield significant benefits in keeping the network load balanced resulting in better overall utilization, despite the well-known fact that TCP interacts poorly with reordered packets. These schemes are also readily implementable and are of low complexity making them an appealing alternative to ECMP and other complicated mechanisms like Hedera or MPTCP. While more work needs to be done, we believe that the myth that TCP interacts poorly with reordering, while may be true in more general settings, does not seem to hold true in regular data center topologies.

6. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM 08*.

[2] M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *NSDI*, 2010.

[3] A. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Infocom*, 2011.

[4] Alan Ford, Costin Raiciu, and Mark Handley, "Tcp extensions for multipath operation with multiple addresses," Internet-draft, IETF, Oct. 2009.

[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *SIGCOMM '09*.

[6] Shan Sinha, Srikanth Kandula, and Dina Katabi, "Harnessing TCPs Burstiness using Flowlet Switching," in *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, November 2004.