On the Data Path Performance of Leaf-Spine Datacenter Fabrics

Mohammad Alizadeh
Insieme Networks
San Jose, CA
malizadeh@insiemenetworks.com

Tom Edsall
Insieme Networks
San Jose, CA
edsall@insiemenetworks.com

***Abstract*—Modern datacenter networks must support a multitude of diverse and demanding workloads at low cost and even the most simple architectural choices can impact mission-critical application performance. This forces network architects to continually evaluate tradeoffs between ideal designs and pragmatic, cost effective solutions. In real commercial environments the number of parameters that the architect can control is fairly limited and typically includes only the choice of topology, link speeds, oversubscription, and switch buffer sizes. In this paper we provide some guidance to the network architect about the impact these choices have on data path performance. We analyze Leaf-Spine topologies under realistic traffic workloads via high-fidelity simulations and identify what is important for performance and what is not important.**

I. Introduction

Datacenter networks have been rapidly evolving in recent years as the nature of their workloads have evolved. Unlike traditional enterprise client-server workloads, the modern datacenter's workload is dominated by server-to-server traffic [1]. This new workload requires intensive communication across the datacenter network between tens to hundreds and even thousands of servers, often on a per-job basis. Hence network latency and throughput are extremely important for application performance while packet loss can be unacceptably detrimental [2], [3]. Accordingly, datacenter networks no longer employ the traditional access, aggregation, core architecture of the large enterprise. Instead, commercially available switches are used to build the network in multi-rooted Clos or Fat-Tree topologies [4], [5]. Generally, architects are choosing smaller, dense, top-of-rack switches at the leaf for server connectivity connected to larger, modular switches in the spine, creating very flat, densely connected networks. Liberal use of multipathing is used to scale bandwidth. This *Leaf-Spine* architecture (shown in Fig. 1) is designed to provide very scalable throughput in a uniform and predictable manner [6] across thousands to hundreds of thousands of ports. In effect, it approximates the ideal network — a very large, non-blocking switch where all servers are directly connected.

The network architect is trying to get as pragmatically close to the ideal network as possible while at the same time controlling costs. These cost considerations force most networks to use some amount of oversubscription and for the architect to continuously evaluate the link speed and switch options available in the market. In this paper we investigate the data path performance of the Leaf-Spine architecture built using 10, 40, and 100 gigabit Ethernet links. Our goal is to give insights to the network architect regarding the implications of the different design decisions they are making for the data path performance of their networks. We ask how close they can get to the ideal network while satisfying their commercial constraints of cost and equipment availability.

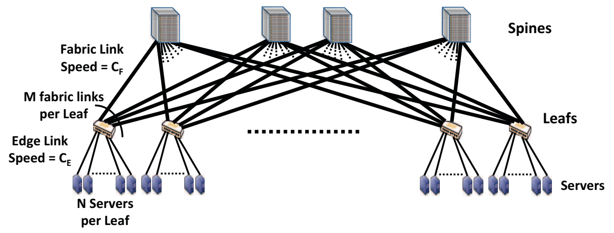


Figure 1. Leaf-Spine datacenter network architecture. Multiple leaf and spine switches are connected in a full bipartite graph. The edge link speed (C_E) may be different from the fabric link speed (C_F) and there may be oversubscription at the leaf fabric links ($MC_F < NC_E$).

Our approach is to take a pragmatic look at the Leaf-Spine datacenter network and to study the tradeoffs *considering only those parameters that the network architect can easily control* such as link speed, oversubscription, and buffering when trying to achieve a solution as close as possible to the ideal. We do not assume the use of sophisticated datacenter congestion control and transport mechanisms [3], [7]–[9], active queue management [10], [11], Priority Flow Control (PFC), or advanced load balancing algorithms [12]–[14]. While the literature has shown that such mechanisms improve performance in various scenarios, they are not widely deployed today. Instead we focus only on standard, widely deployed mechanisms: TCP/UDP transport, hash-based ECMP for path selection, and drop-tail queuing as typically found in most commercial datacenter switches.

We study the Leaf-Spine architecture's performance via high-fidelity packet-level simulations in the OMNET++ [15] simulator. We use the Network Simulation Cradle [16] package to port the actual TCP source code from Linux 2.6.26 to our simulator. Since correctly capturing TCP behavior is perhaps the most challenging aspect of simulating real networks, using unmodified code from a real operating system gives us high confidence in our results. Our study mainly focuses on flow completion time (FCT) [17] performance for realistic dynamic workloads with a mix of small and large flows and also bursty traffic patterns with the Incast [2] problem. We derive our traffic patterns from empirically observed workloads in production datacenters [3], [5]. The key findings of our study are:

1) Using higher speed links in the fabric (between the leaf and spine switches) relative to the edge (at the servers) significantly improves the efficiency of Equal Cost Multipath (ECMP) load-balancing. In particular, we observe up to 40% degradation in FCT for large flows relative to an ideal non-blocking switch when both the edge and fabric links run at 10Gbps. However, with 40Gbps or 100Gbps fabric links, the performance is very close to ideal.
2) Oversubscription does not negatively impact performance for moderate levels of load (e.g., up to 60% of the available bisection bandwidth), but it makes the network more fragile compared to non-oversubscribed (or lesser oversubscribed) networks at high load.
3) The leaf switch uplink ports and spine switch ports observe similar degrees of queue buildup. Hence, per-port buffer sizes at the leaf and spine tiers should generally be consistent to be effective. In particular, having extremely large buffers (e.g., as afforded by off-chip memory) in one tier is not useful if the buffers in the other tiers are much smaller. At the same time, we find that Incast problems (traffic bursts) are more severe at the leaf switches than at the spine switches. Therefore, increasing the buffer size in the leaf switches mitigates Incast more than increasing the spine buffers.

We do not claim that all these findings are new. Indeed, some of these issues have been previously observed in the literature. However, in speaking with network designers, we have found there is a lot of confusion regarding the impact of different design choices on the data path performance of datacenter fabrics. We hope the present paper will help shed light on these issues and be useful to practitioners.

II. Methodology

Our goal is to understand how close the data path performance of the Leaf-Spine architecture is to a large ideal (non-blocking) switch for workloads commonly encountered in practice. Specifically, the Leaf-Spine architecture differs from an ideal switch in two important ways which may reduce its performance: (i) inefficiencies in multipath load-balancing may create hot-spots and reduce effective bandwidth; and (ii) queue buildup in the spine of the network may delay or, worse, cause drops for packets traversing the spine (even traffic destined to non-congested ports). We now briefly describe the setup we use to study these issues.

Simulation platform: We start with the INET framework [18] that provides models of standard L2/L3 networking protocols and make extensions to support multipath routing via ECMP. We leverage the NSC [16] package to port the Linux 2.6.26 TCP stack to our simulator and enable Reno (with SACK) as the congestion control algorithm.

Topology: We use a Leaf-Spine topology (Fig. 1) with 100 10Gbps servers organized in 5 racks. We consider various scenarios with different link speeds (10Gbps, 40Gbps, or 100Gbps) and different over-subscription ratios (1:1, 2.5:1, and 5:1) for the fabric links between the leaf and spine switches. The leaf switches are assumed to have a base latency (without buffering) of 700ns and the spine switches have a base latency of 2μs. The host networking stack adds an additional latency of 10μs, for a total one-way socket-to-socket latency of 25μs (round-trip time is 50μs).

Switches: We model all switches as shared-memory output-queued switches. Unless otherwise specified, the total buffering at each switch is 10MB. Each switch queue has a reserved buffer of 30KB and can grab up to half (5MB) of the total buffer during congestion.

Workload: We simulate a dynamic workload with foreground query traffic and background traffic. The query traffic models traffic spawned by user queries in the backend of a web application. Queries arrive according to a Poisson arrival process with an average rate of 10 queries per second per server. When making a query, a server requests a 1MB file from n other randomly chosen servers in parallel, with each of the servers responding with $1/n^{th}$ of the file (the fanout, n, is chosen uniformly at random). The query completes when the *entire* file is received. The transfers are over persistent TCP connections; hence the query traffic is very bursty and creates the TCP Incast [2] problem since transfers start with the TCP window already open as opposed to going through the slow-start algorithm.

The background flows are a mix of small and large flows with the flow sizes drawn from distributions empirically observed in production datacenters. We consider distributions from a web search [3] and a data mining [5] cluster. In the interest of space, we only report the results for the web search distribution (the results for the data-mining distribution are qualitatively similar, though it generally creates less congestion than the web search distribution). The distribution exhibits common heavy-tailed characteristics: 70% of the flows are smaller than 1MB but contribute only 5% of all data bytes, with the rest from 1–30MB flows. These flows also arrive according to a Poisson process and we vary the rate of the arrivals to obtain a desired level of load for the background traffic. Since our goal is to compare the Leaf-Spine network with a large non-blocking switch we generate flows such that all traffic traverses the spine. That is, we ensure that the (randomly chosen) source and destination servers for each flow are in *different* racks. Of course this is not realistic since some amount of locality usually exists at the rack level, but it allows us to stress the network and tease out the impact of different aspects more clearly.

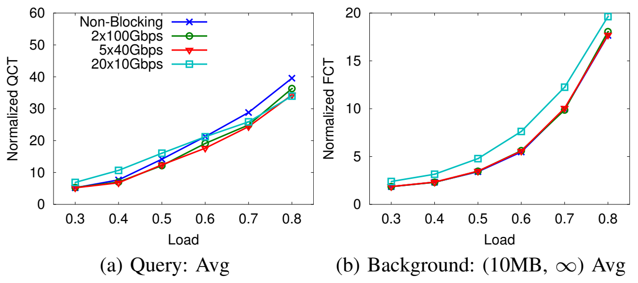


Figure 2. Average normalized query completion time (QCT) and flow completion time (FCT) for large (10MB, ∞) background flows. The completion time for each query (flow) is normalized to the minimum possible value for that query (flow); i.e., the value achieved if the flow (query) is transmitted at 10Gbps without any interference from other flows.

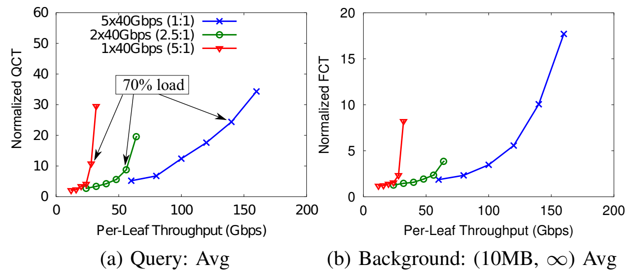


Figure 3. Comparison of 1:1, 2.5:1, and 5:1 oversubscribed topologies with respectively 200Gbps, 80Gbps, and 40Gbps of uplink capacity per-leaf switch. In each case, we vary the traffic intensity between 30–80% of the uplink capacity in 10% increments.

III. Results

In this section we present our simulation results in three main parts. First, we present simulations that compare the performance of networks with 10Gbps, 40Gbps, and 100Gbps fabric links. Next, we evaluate the impact of oversubscription at the leaf switches. Finally, we study the impact of the buffer size of the leaf and spine switches.

A. *Impact of link speed*

We begin by comparing three topologies with different link speeds between the leaf and spine switches: each leaf switch has either (i) 20×10Gbps, (ii) 5×40Gbps, or (iii) 2×100Gbps uplinks, amounting to 200Gbps total uplink capacity. Note that all three networks are non-oversubscribed since the leaf switches have 20 10Gbps front-panel ports. We compare these topologies with an ideal non-blocking network where the leaf switches are connected to a single spine switch using "infinite" capacity links. This represents what we would achieve if we built the entire datacenter network as one large output-queued switch.

Figure 2 shows the average query completion time (QCT) and the average flow completion time (FCT) for large ($>$ 10MB) background flows at 30% to 80% traffic loads. In the interest of space, we omit the plot for the small background flows which, similar to the plot for query traffic, does not show much difference between the different topologies. Indeed, it is only for the large flows that we find a notable difference for the 20×10Gbps topology with FCTs that are $\sim$12–40% higher than the other topologies. Since the FCT for large flows is determined by the bandwidth they achieve, this shows that ECMP load-balancing is inefficient when the fabric link speed is 10Gbps. However, increasing the fabric link speed to 40Gbps or 100Gbps improves the load-balancing efficiency and achieves performance that is almost identical to Non-Blocking. Interestingly, there isn't much difference between using 40Gbps links and 100Gbps links (which are more expensive) in the spine.

Further, though we clearly observe the Incast problem, particularly at high load (as much as 12% of the queries experience TCP timeouts at 80% load), there isn't much difference between the three topologies suggesting that the extent of Incast is predominately determined by the size of the buffers (10MB at each switch). We revisit this in §III-C.

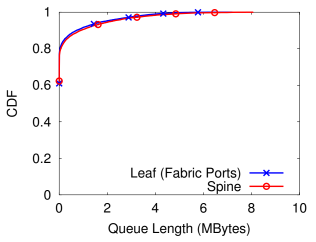
B. *Impact of oversubscription*

We now evaluate the impact of oversubscription at the leaf switches. We introduce oversubscription to the non-oversubscribed configuration of the previous section by removing some spine switches from the network. Specifically, we consider two oversubscribed configurations with (i) 2×40Gbps (2.5:1 oversubscribed) and (ii) 1×40Gbps (5:1 oversubscribed) uplinks at each leaf switch. For each topology, we vary the background traffic intensity from 30% and 80% of the *available capacity* in the spine (the query rate is 10 queries per second per server in all cases). For example, for the 2.5:1 oversubscribed topology which has 80Gbps of uplink capacity per-leaf, the traffic intensity varies between 24Gbps to 64Gbps per-leaf.

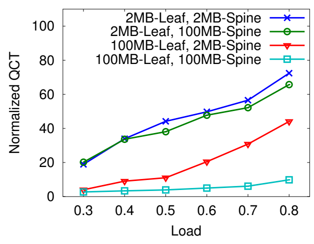
The average query and flow completion time for large background flows are shown in Figure 3 (the behavior for other background flows is similar). We find that oversubscription does not significantly degrade performance as long as the load is not very high. For loads of 60% and lower, the performance of the two oversubscribed topologies is inline with that achieved by the non-oversubscribed topology at the same traffic intensity. It is only for the two data points at 70% and 80% load (the two rightmost data points of each line) that the performance of the oversubscribed topologies breaks away from the overall trend and is notably worse.

C. *Impact of buffer size*

We now study how the buffer size in the leaf and spine switches impacts performance. We use the 2.5:1 oversubscribed topology of the previous section and evaluate four combinations with either 2MB or 100MB total buffer in the leaf and spine switches (a single queue can grab up to half the total switch buffer during congestion events). We use these values to make a clear contrast between the extremes of small and large buffers at the leaf and spine switches and to understand where buffers are more effective.



(a) Queue buildup at 60% load with 100MB leaf and spine buffer size



(b) Query: Avg

Figure 4. (a) CDF of queue occupancy at the leaf fabric (uplink) ports and the spine ports. (b) Average normalized query completion time for different buffer sizes at the leaf and spine switches. This is for a 2.5:1 oversubscribed topology with 2 × 40Gbps uplinks at each leaf switch.

Natural buffer occupancy: First, we study the natural buffer occupancy at the leaf and spine switches defined as the buffer occupancy when the switches have unlimited buffer space and thus do not drop packets. As a representative example, Figure 4a shows the distribution of the buffer occupancy at the leaf fabric ports (uplinks) and spine ports at 60% load with 100MB buffers (which is large enough to avoid packet drops) at both the leaf and spine switches. We find that the distribution of buffer occupancy is nearly identical at the leaf fabric ports and spine ports. This is because these ports run at the same speed and are subject to very similar traffic (in terms of intensity and flow size distribution). Of course, this stems from the nature of the all-to-all workload. For a different workload with predominately one-to-many communication patterns (among leaf switches), the natural buffer occupancy would be larger at the leaf fabric ports, while for many-to-one patterns, the spine ports would have more queue buildup. Nonetheless, this experiment demonstrates that without specific knowledge of traffic patterns, having significantly more buffering in one tier than the other is wasteful.

Where are larger buffers more effective? While the natural buffer occupancy indicates that the buffer sizes at the leaf and spine switches should not differ too much, it could be useful to have somewhat larger buffers in one tier versus the other depending on the extent to which they experience Incast [2] and traffic bursts. Indeed, comparing the average query completion time for the different buffer combinations in Figure 4b, we find that the "100MB-Leaf, 2MB-Spine" configuration performs much better than the "2MB-Leaf, 100MB-Spine" configuration (though both are worse than having 100MB buffers everywhere). Interestingly, the average query completion time with 2MB buffers at both the leaf and spine switches is not that much worse than with 2MB at the leaf and 100MB at the spine. This is explained by the number of packet drops for the different buffer sizes. For instance, as shown in Table I, at 60% load there are 261,350 dropped packets with 2MB buffers at both the leaf and spine switches, while increasing the spine buffer to 100MB only reduces the total drops to 238,499 packets.

	2MB-Spine	**100MB-Spine**
2MB-Leaf	261,350	238,499
100MB-Leaf	126,142	0

Table I
NUMBER OF PACKET DROPS AT 60% LOAD.

IV. CONCLUSION

The datacenter network is evolving to a Leaf-Spine architecture that aims to approximate a very large non-blocking switch. This paper investigated the data path performance of these network using high-fidelity simulations with realistic traffic workloads and real TCP stacks to shed light on different design tradeoffs such as link-speeds, oversubscription, and buffering. We found that using higher speed links in the fabric (between the leaf and spine tiers) relative to the edge (at the servers) greatly improves the efficiency of ECMP load-balancing, and that having significantly larger per-port buffer sizes in the leaf or spine tier compared to the other is wasteful. Further, it is better to apply additional buffering in the leaf tier than the spine tier to control Incast problems.

REFERENCES

[1] "Cisco Global Cloud Index: Forecast and Methodology, 2011–2016," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns1175/Cloud_Index_White_Paper.pdf.
[2] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *Proc. of ACM SIGCOMM'09*, 2009.
[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proc. of ACM SIGCOMM'10*, 2010.
[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. of ACM SIGCOMM'08*, 2008.
[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 *Proc. of ACM SIGCOMM'09*, 2009.
[6] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.
[7] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: meeting deadlines in datacenter networks," in *Proc. of ACM SIGCOMM'11, 2011.*
[8] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proc. of ACM SIGCOMM'12*, 2012.
[9] M. Alizadeh, S. Yang, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Deconstructing datacenter packet transport," in *Proc. of HotNets-XI*, 2012.
[10] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
[11] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *Proc. of USENIX NSDI'12*, 2012.
[12] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proc. of USENIX NSDI'10*, 2010.
[13] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. of ACM SIGCOMM'11*, 2011.
[14] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: reducing the flow completion time tail in datacenter networks," in *Proc. of ACM SIGCOMM'12*, 2012.
[15] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM2001)*, vol. 9, 2001.
[16] S. Jansen and A. McGregor, "Performance, Validation and Testing with the Network Simulation Cradle," in *Proc. of IEEE MASCOTS'06*, 2006.
[17] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *SIGCOMM Comput. Commun. Rev.*, Jan. 2006.
[18] "INET framework," http://inet.omnetpp.org/.