Measuring Burstiness in Data Center Applications

Jackson Woodruff
University of Edinburgh
J.C.Woodruff@sms.ed.ac.uk

Andrew W Moore
University of Cambridge
andrew.moore@cl.cam.ac.uk

Noa Zilberman
University of Cambridge
noa.zilberman@cl.cam.ac.uk

ABSTRACT

Buffer sizing is a tricky task — it depends on a large number of variables, ranging from congestion control to traffic engineering. Still, the most unpredictable contributors are the workloads running in the network. The link utilization and burstiness of these workloads dictate the buffer depth needed by a switch. But what is a burst? Do traditional definitions still apply in the age in which switches transfer terabits of data and billions of packets every second? Unless we assess bursts correctly, we are unlikely to size buffers appropriately. In this work, we present a measurement-led evaluation of the burstiness of different data center applications. We address the question of "what is a burst?" and assert that common techniques cannot answer this question in modern data centers. We quantify the change in burstiness of the studied applications across multiple vectors, including latency and network perspective, and generalize our results to the common case. Our observations can inform future buffer sizing efforts and guide switch configurations. Our dataset is openly available for the benefit of the community.

1 INTRODUCTION

What is the optimal size of a buffer? The obvious answer is "it depends." Traffic engineering, congestion control, network utilization and oversubscription are among many factors that affect the required size of a buffer [10]. Still, the most elusive factor that affects buffer sizing is the workload.

Previous studies within data centers [17] of the effect of bursts and microbursts on buffer utilization have shown significant differences between applications. These works [3, 7] considered the effects of bursts and microbursts on millisecond and microsecond granularity, and used (standard) switch functionality to analyze network data. Alizadeh *et al.* [1] define a microburst as a burst of traffic too short to be prevented by traditional congestion control protocols. This definition can be made more precise given a sub-nanosecond view of the network available today. Zhang *et al.* [17] define microbursts as small periods (<1 ms) of high utilization. During 1 ms, more than a megabyte of data can arrive, even on a 10G network link.

Time is deceiving. A 100G link with 40% utilization using 100B packets means 50kB of data over 10 µs, but those may arrive as a single burst of 50kB or with 100B packets spaced by 150B-equivalent gaps. In the first case, a buffer can be filled,[1] while in the second a queue may not build at all. Approaches such as in-network telemetry [8] may not be able to capture these transitional and momentary effects. Looking only at a watermark indication (e.g., as provided by counters on many NICs [6, 12]) is not the solution either; there is no way to distinguish between a singular and a recurrent event.

In this work, we try an address the question of *What is a burst?* Or rather, *How should we define a burst for buffer sizing purposes?* While we do not have full visibility into the switch, we do have the ability to look outside it by using packet traces. In this work we explore the use of packet traces with sub-nanosecond timestamp resolution to explore the meaning of burstiness using different data center applications. While we use a limited-scale, local setup, our environment does allow us to explore three interesting vectors: (1) What is the effect of the application? (2) What is the effect of server aggregation? (3) What is the effect of latency?

We make the following contributions:

- We collect a set of packet traces of different data center applications, with sub-nanosecond capture timestamp resolution. Our dataset and tools are available at [15, 16].
- We explore the definition of a burst using the different applications' packet traces.
- We explore the effect of latency and server aggregation on burstiness.
- We discuss generalizing our results and offer recommendations for buffer sizing.

The rest of this paper is organized as follows: Section 2 discusses our experimental setup and the applications used. Section 3 describes our dataset. Section 4 analyzes the burstiness of our applications. Section 5 discusses our results and the general case. Section 6 concludes.

BS'19, December 2-3, Stanford, CA

[1] Even if only to adjust to small port-frequency variations.

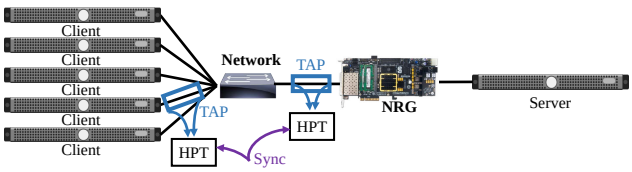

Fig. 1: The experimental setup used. One server is connected to the network through NRG. Traffic from the server and one client is captured using ExaNIC-HPT.

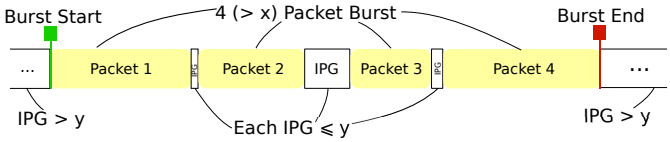

Fig. 2: How we define a burst in terms of x packets with an IPG of y ns.

2 EXPERIMENTAL SETUP

Our experiments use 10 machines running Ubuntu server 16.04.5 and Linux kernel 4.4.0-131-generic. Up to seven machines run the benchmarks, two more capture traces, and one acts as a management node. All the machines have a 10 GE test NIC installed, either Intel 82599ES NICs or Solarflare SFC9220 NICs. The NIC of each machine is recorded during each benchmark run. Benchmark machines are equipped with Intel Xeon E5-2637 v4 3.50GHz CPUs and 64 GiB of RAM.

An Arista 7124FX switch connects the machines. STP and LLDP are disabled to avoid uncontrolled traffic. A 3 m fiber connects each machine to the switch. We use an ExaNIC HPT [5] for traffic capture. The HPT captures at a resolution of 250 ps, which enables our high-resolution analyses.

We use NRG, a NetFPGA-based latency appliance, introduced by Zilberman *et al.* [19], to control the static latency between one machine (the server) and the rest of the setup. NRG is located between the server and the switch, and provides nano-second scale latency control. Unless noted otherwise, all latencies are applied both on the Tx and on the Rx side. Unlike other latency emulation tools (e.g., NetEm), NRG provides high resolution and maintains precision even under high data rates.

In this paper we discuss two benchmarks, Memcached and Tensorflow, each with a single server/master and multiple clients/workers. Memcached runs using the Facebook ETC workload [2], and it runs for 30 seconds. Memcached's server-side is saturated. Tensorflow uses the MNIST dataset,[2] and it trains for 20,000 iterations. The capture data records configuration-specific information. In addition, our dataset includes records of two more benchmarks: Apache web server and DNS.

3 DATASET

We study burstiness covering three axes: application, latency, and traffic direction. We collect a large dataset of experimental results for each application. Our study covers over 200 experiments, not all of them discussed in the paper, and our dataset includes additional, related traces. Our traces provide sub-nanosecond resolution and approximately 30 ns clock synchronization between traces captured on different cards. We note that storage and run time were the two limiting factors of this study. We also make available a Python processing framework as libpcap does not support sub-nanosecond timestamps.

Our analysis focuses on the server-side data. Because our benchmarks follow a client-server model, this reveals the burstiness that the switch experiences. All traffic passing through the switch is going from or to the server machine. Beyond the traces, we also collect metadata: benchmarking machines' logs (stdout, hardware configuration, syslogs), capture card output logs, and NRG statistics logs.

Our dataset is available at [16]. The software environment used is available at [15].

4 WHAT IS A BURST?

Bursts are traditionally defined as periods of high packet arrival rates [9]. Existing work on microbursts has taken this approach. Alizadeh *et al.* [1] define a microburst as burst of traffic too short to be prevented by traditional congestion control protocols. Zhang *et al.* [17] define microbursts as small periods (<1 ms) of high utilization. Their data were sampled at a resolution of 25 µs: we assert that this is not high enough resolution for data center networks as data rate increases, as even on a 10G network link, more than a megabyte of data can arrive within 1 ms.

We adopt a definition for bursts more appropriate for buffer sizing. Instead of considering the *amount* of traffic, we consider the traffic's *density*, which partially resembles *link utilization*. To define bursts, and microbursts, we consider subsets of the question: How many packets become a burst? How close should these packets be to each other? What is the link utilization required to call a train of packets a burst?

4.1 Methodology

We define a burst as some sequence of x packets, each arriving within y bit times of each other. Figure 2 shows this. This definition yields a parameter space in x and y. In section 4.2, we explore how best to select these parameters. This definition allows for detection of microbursts on the scale of individual packets. Because the meaning of a microsecond or a nanosecond depends on data rate, we instead refer to "bit

[2] http://yann.lecun.com/exdb/mnist/

time". A given amount of data transmitted over a microsecond at 10 Gbps will be 10 times denser than the same amount of data transmitted at 100 Gbps. In other words, a gap of y ns corresponds to less burstiness at 100 Gbps than at 10 Gbps. As our setup runs at 10 Gbps, for the rest of this paper we refer to y in units of nanoseconds, and note that this should be scaled for other data rates.

Next, we explore how changing x and y affects the lengths and link utilization of bursts for given traces. To calculate the average bandwidth during a microburst, we define:

$$BW = \frac{\sum_{\text{Packets in Burst}} \text{Packet Size(s)}}{t_{end,last} - t_{start,first}}$$

And link utilization is $BW/Link\ Capacity$.

4.2 Parameter Selection

Our approach requires two parameters: the inter-packet gap (IPG) between consecutive packets, and minimum number of consecutive packets with this IPG (or smaller) for the sequence of packets to be considered a burst.

To choose these parameters, we run the applications with five clients/workers and one server, and analyze the collected traces. We directly extract IPG,[3] and calculate bandwidth within 100 µs windows. This window size fits multiple 1.5KB packets, and provides a "high-level" view. The repository includes similar analyses, conducted at 1 µs and 10 µs. We then consider the 95th, 99th, 99.9th, and 99.99th percentiles within this distribution. Table 1 shows these for each application. Bandwidth is the link utilization multiplied by 10 Gbps. For clarity, the IPG used in the table, and through the rest of this section, accounts for an average 16 ns inter-frame gap (IFG) and preamble[4].

The values in the table show stark differences between applications, both in terms of utilization and of IPG. In addition, even for the same application, client-to-server and server-to-client traffic is different, e.g., Memcached's replies (server-to-client) have ×2.75 higher 99th percentile utilization than Memcached's requests (client-to-server).

4.3 Burst Length

Using our high-resolution traces, we explore the question of *What is the length of a burst?* by looking at the packet trains at the tail, determined by bandwidths at the 99th percentile and beyond. We use the mean IPG during these tails, and require a burst to have a minimum a number of consecutive packets: more than two, four or eight, each within the given IPG.

[3]Note that this is not inter-arrival time.

[4]The repository provides data without this constant, e.g., for deficit idle counter analysis.

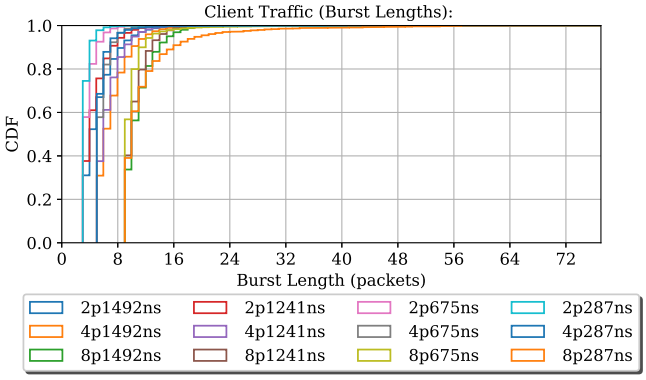

Fig. 3: The length of incoming bursts in Memcached. xp$y$ns represents the x and y burst parameters used.

Figure 3 shows the length of incoming (client-to-server) Memcached query bursts. The more stringent the requirement on number of packets within a burst, the higher the probability a burst is very long. This is partly by definition, as all bursts of more than 2 packets and y IPG will emcompass all bursts of more than 4 and 8 packets of the same IPG. In Memcached, bursts can be short: for more than 2 packets within 287 ns of each other (2p287ns) the probability of minimum burst size (three packets) is 75%. However, the tail can be quite long (up to 77 packets) as shown on the eight packet, 287 ns line (8p287ns).

Figure 4 shows how the incoming bandwidth depends on the parameters used. Observed bandwidths are generally grouped by the IPG parameter. At 8 packets with 287 ns IPG, we have identified a distinct group of bursts: 75% of these bursts have bandwidth greater than 9.5Gbps, a series of truly back-to-back packets. We can see that bandwidth utilization during other burst definition-groups is fairly low, although long-tails do exist.

In Tensorflow, packets did not arrive at consistent intervals. Packets arrived at intervals of: 1 ns, 1 ns, 1 ns, 8 ns. This keeps the mean IPG low (and hence, the bandwidth high), but means that the 4 ns we derived above does not accurately capture the bursts. To accommodate this insight, we have used larger IPGs (10ns) in the analysis of Tensorflow.

Figures 5 and 6 show the results of Tensorflow analysis. First, we can see that bursts are significantly longer than in Memcached (Figure 5) and can reach more than 1,000 packets. This is due to Tensorflow's two-phase algorithm [14], which means that workers send large batches of data — the neural network's edge weights. These large batches mean Tensorflow is network bound for periods of time. In contrast, Memcached is memory bound, which limits data rate (and in turn, the burstiness). For these reasons, Tensorflow is a far burstier application than Memcached. The longest bursts are a on a different order of magnitude from the Memcached

Application	Packet Size (Median)	95th		99th		99.9th		99.99th	
		Utilization	IPG	Utilization	IPG	Utilization	IPG	Utilization	IPG
Memcached	156B	7.72%	1492ns	9.14%	1241ns	15.60%	675ns	30.34%	287ns
Tensorflow	1518B	99.6%	4.8ns	99.62%	4.8ns	99.64%	4.8ns	99.65%	4.8n

Table 1: The tail bandwidths reached by each application using a 100 µs window. The table indicates client/worker to server communication. The IPG specified indicates the median IPG for a given utilization.

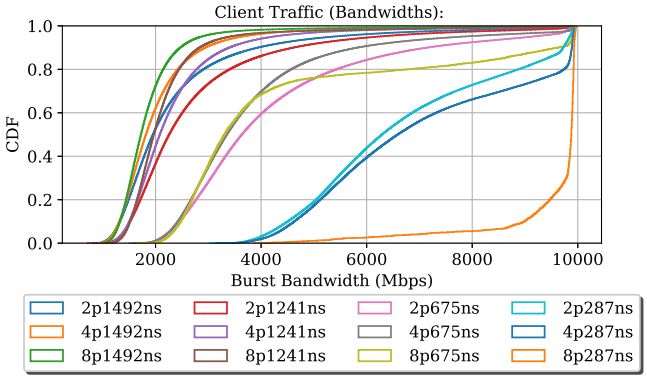

Fig. 4: The bandwidth of incoming bursts in Memcached. xp$y$ns represents the burst parameters used.

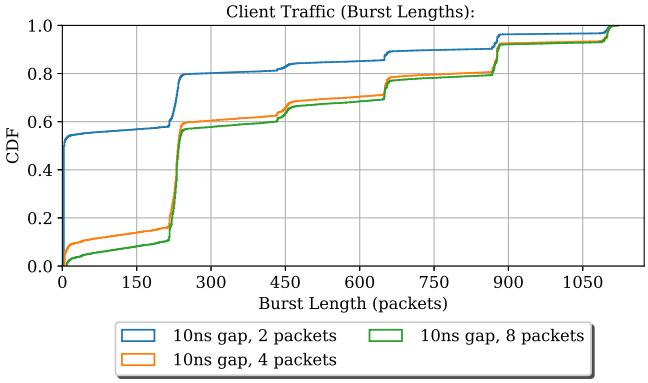

Fig. 5: Tensorflow incoming burst lengths.

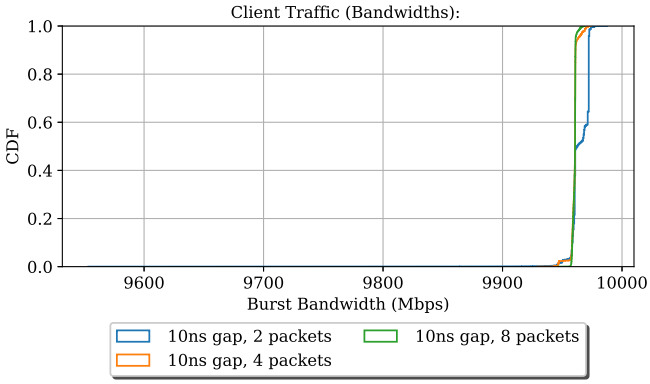

Fig. 6: Tensorflow incoming burst bandwidths.

results, where the bursts are at most 77 packets long. This suggests that applications should see different buffer sizes. Furthermore, a thousand packets are over a megabyte in size, and are not a microburst; no one should expect a buffer similar in size to such a burst. Memcached's 77 packets are roughly 12KB, a number closer to commonly used per-port buffers.

4.4 The effects of server aggregation

There is a key difference between data arriving at a server and data sent from the server: the data arriving at the server is at best semi-synchronized, while the data from the server can be completely aggregated. There are also protocol asymmetries: the Memcached server typically generates longer responses than requests, so outgoing bandwidth requirements are more taxing than incoming bandwidth requirements. These differing behaviors suggest that under some scenarios asymmetric buffer sizing policies can more effectively use available resources.

Figures 7 and 8 show the effects of this server-side aggregation. We can see both higher bandwidths during bursts (Figure 7) and longer burst lengths (Figure 8). As an example, for incoming packets, if we see an eight packets long burst, there is a 25% chance that we will see more than 11 packets in the burst. However, in the outgoing traffic, if we see a burst eight packets long, there is a 25% chance that there will be more than 15 packets in the burst. The bandwidth graph shows similar increases in utilization.

Thus, there are two arguments here that buffer sizing in switches should not be symmetric: application workloads are not symmetrical, and, further, the effect of aggregation of messages is stronger close to the server/aggregation side.

4.5 How does latency affect burst size?

To observe the effect of latency on bursts, we use the NRG tool described earlier to control the latency visible to the applications. Our experiments range in additive latency from 0us to 1ms, with a minimum step of 25us, as we see little effect below that [11, 19]. The base latency of our setup is roughly 10us. Figure 9 shows the effect at 50us latency (100us RTT). There is a small difference in the burst lengths; in particular, bursts are less likely to be very long. In Memcached, as latency increases, the probability of long bursts decreases.

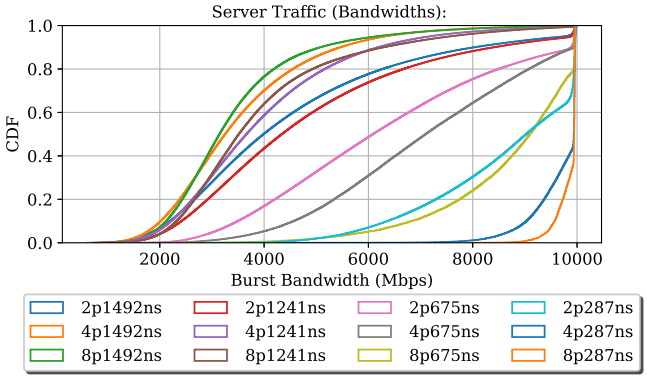

Fig. 7: The outgoing burst bandwidth in Memcached. *x*p*y*ns represents the burst parameters used.

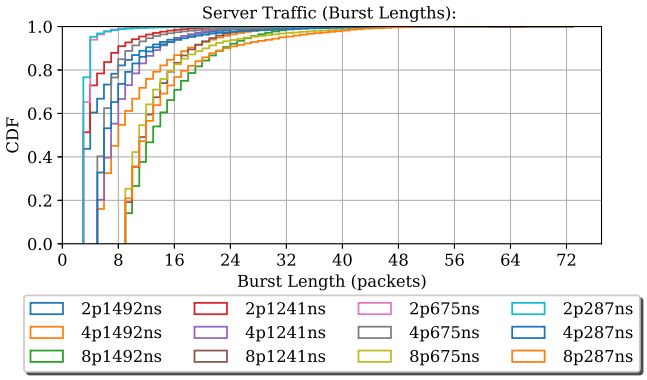

Fig. 8: The outgoing burst length in Memcached. *x*p*y*ns represents the burst parameters used.

In Tensorflow, we see little difference as latency increases. This means that even with much longer latencies we still see bursts of over a thousand packets. These figures are omitted as they resemble previous ones. An important note to consider is that this analysis is based on bursts with a very small IPG of 10ns. At larger IPGs, differences may be visible.

5 DISCUSSION AND RELATED WORK

We address the question *What is a burst?* in the context of current-day data center applications. We focus our attention on application-level bursts, considering one application at a time. We leverage sub-nanosecond resolution traces to explore burst lengths defined in terms of packet IPG. Our work provides an analysis of the resolution needed to record microbursts at 10G, and can be scaled to higher bandwidths.

Others have used different definitions of microbursts. For example, Zhang *et al.* [17] use a windowed approach to detect periods of high utilization. However, their windowed approach breaks down for bursts longer or shorter than the window. As we have shown, these are both very common.

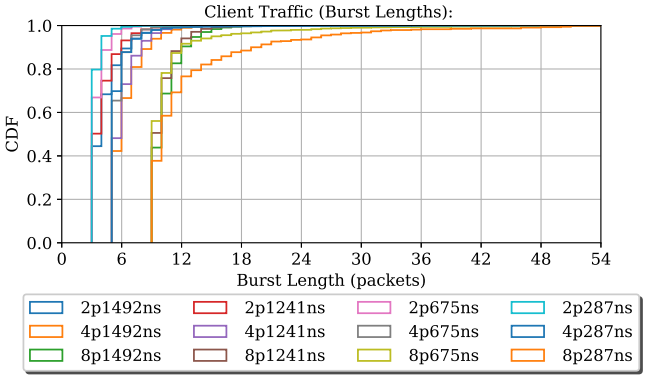

Fig. 9: The length of incoming bursts in Memcached with 50 µs of artificial delay. *x*p*y*ns represents the burst parameters used.

Further, the 25 µs resolution data used by Zhang *et al.* limits their approach to window sizes large enough for hundreds of packets at 10G, more than enough to fill a small buffer.

At higher bandwidths, higher resolutions will be required to enable similarly detailed analyses. Our work suggests that differences in burstiness can be seen on the scale of 10 bytes, which is a mere 0.8 ns at 100G. Scaling our mechanism for detecting bursts is important for more than just higher bandwidths. Modern network devices can have more than 256 ports. Our detection mechanism is scalable to devices with many ports because it relies only on the IPG between packets and a static packet count. The question over what constitutes the most useful measure of burst size remains. Should the measurement be made in terms of packets or bits? The answer may well depend on the device, e.g., whether the buffer is assigned in fixed-size units, of if buffer assignment is flexible and bus-width alignment is the only restriction.

Burst size on its own should not dictate buffer size. Many other factors come into play, such as resource availability, QoS guarantees, and SLAs. Nevertheless, burst size is an important indicator for buffer sizing because it highlights the peak throughputs network devices can expect.

Limitations of the study. To achieve reproducibility, our study has limited scale. We run our experiments with at most seven machines. Further, we only run a small number of applications, which are not representative of today's heterogeneous data center cloud environment, much less the Internet. This is largely due to resource constraints; the applications we ran produced more than 2 TB of data which required significant processing time. This data restriction similarly limited the number of iterations we used. Finally, our work does not have network-wide visibility: our conclusions are drawn exclusively from the perspective of the server. Although this reveals instances of burstiness, we don't

have true switch-internal visibility. Further understanding can be gained by combining server- and client-side traces.

This work describes an analysis that will fit clusters running homogeneous workloads [2]; it is less suited to cloud environments running many user applications. Nevertheless, we believe that our work can provide relevant and interesting insights into buffer sizing.

Future Work. Our dataset includes more applications and network-parameter variations than covered in this paper, and we intend to explore these in an extended version. Our work contains relevant information for data center buffers. Internet traces such as CAIDA [4] provide an opportunity for similar analysis of Internet traffic. Traces from real data centers also provide an opportunity for further analysis. However, such data are largely lacking due to privacy concerns among other issues. Facebook has released a trace [2], but this trace is sampled to 1/30,000 packets, making it infeasible to use for IPG-based analysis. Even without traces from further afield, we intend to explore a wider range of applications in more complete contexts, using synchronized client-server traces to observe burstiness.

6 CONCLUSION

This work analyzes burstiness using a definition based on the gap between packets, rather than the amount of aggregated data. Our work shows that even at 10G, sub-microsecond precision is required to understand application burstiness. We present a methodology for understanding burstiness with high-resolution packet traces. Our traces help understand the traffic load that buffers in data center switches face on small timescales.

Our analysis of two data center applications, Memcached and Tensorflow, shows that each application behaves differently in terms of burstiness. Results indicate that Tensorflow exhibits far burstier behavior, with the median burst length approximately 16 times longer than Memcached. We argue that required buffer sizes are application dependent.

Our exploration of the effect of server aggregation demonstrates that the server synchronization increases burst-related bandwidth spikes. We argue that in cluster workloads buffer sizes need not be symmetrical on different ports.

Finally, our investigation into the effect of latency on burst size finds that additional latency decreased burstiness for Memcached, but did not affect maximum burst size for Tensorflow. Thus, we propose that switches handling low-latency paths (e.g., top-of-rack switches) may need larger buffers per flow than those handling higher-latency paths (e.g., core switches). This approach resembles other works pushing buffering to the edge [18].

7 ACKNOWLEDGEMENTS

We thank Malcolm Scott for his invaluable help with the experimental setup, and Murali Ramanujam for his help in setting up the benchmarks. This project was partially funded by the Leverhulme Trust (ECF-2016-289) and the Isaac Newton Trust. We also thank Xilinx for their support.

REFERENCES

[1] M Alizadeh, A Greenberg, DA Maltz, and J Padhye. 2010. Data Center TCP (DCTCP). *ACM SIGCOMM CCR* 40, 4 (2010), 63–74.
[2] Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. 2012. Workload analysis of a large-scale key-value store. *ACM SIGMETRICS Performance Evaluation Review* 40, 1 (2012), 53.
[3] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *IMC '10*. ACM.
[4] CAIDA. [n. d.]. *The CAIDA UCSD Anonymized Internet Traces*. https://www.caida.org/data/passive/passive_dataset.xml/.
[5] Exablaze. [n. d.]. *ExaNIC HPT*.
[6] Intel. 2016. Intel Ethernet Controller X710/XL710 and Intel Ethernet Converged Network Adapter X710/XL710 Family: Linux Performance Tuning Guide. (2016).
[7] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The nature of data center traffic. In *IMC'09*.
[8] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. 2015. In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM*.
[9] W. E. Leland and D. V. Wilson. 1991. High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In *IEEE INFCOM '91*. 1360–1366 vol.3.
[10] Nick McKeown, Guido Appenzeller, and Issac Keslassy. 2019. Sizing Router Buffers (Redux). *ACM SIGCOMM CCR* (2019), 69–74.
[11] Diana Andreea Popescu, Noa Zilberman, and Andrew William Moore. 2017. *Characterizing the impact of network latency on cloud-based applications' performance*. Technical Report. Computer Laboratory technical reports, UCAM-CL-TR-914.
[12] Solarflare. 2014. Solarflare Server Adapter User Guide. (2014).
[13] O. Tange. 2011. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine* 36, 1 (Feb 2011), 42–47.
[14] TensorFlow. [n. d.]. *Tensorflow Achitecture*. https://www.tensorflow.org/guide/extend/architecture.
[15] Jackson Woodruff. 2019. Measuring Burstiness in Data Center Applications: Code Repository. https://github.com/cucl-srg/Measuring-Burstiness. (2019).
[16] Jackson Woodruff, Andrew W. Moore, and Noa Zilberman. 2019. Measuring Burstiness in Data Center Applications: Dataset. https://www.cl.cam.ac.uk/research/srg/netos/projects/latency/buffer2019/. (2019).
[17] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. 2017. High-resolution measurement of data center microbursts. In *IMC'17*. ACM, 78–85.
[18] Noa Zilberman, Gabi Bracha, and Golan Schzukin. 2019. Stardust: Divide and conquer in the data center network. In *NSDI'19*. 141–160.
[19] Noa Zilberman, Matthew Grosvenor, Diana Andreea Popescu, Neelakandan Manihatty-Bojan, Gianni Antichi, Marcin Wójcik, and Andrew W. Moore. 2017. Where has my time gone? *PAM'17*, 201–214.