Measuring Burstiness in Data Center Applications

Jackson Woodruff
University of Edinburgh

Andrew W Moore
University of Cambridge

Noa Zilberman
University of Cambridge

ABSTRACT

Buffer sizing is a tricky task, depending 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 workload dictate the depth of the buffer we need on a switch. But what is a burst? Do traditional definitions still apply at the age where 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 work evaluating the burstiness of different data center applications. We address the question of "what is a burst?" and assert that common techniques can not answer this question. We observe the change in burtiness 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 works and guide switch configurations. We are making our dataset 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: so many factors affect the required size of a buffer [1]. Still, the most illusive factor that affects buffer sizing is the workload.

Previous works [2] have studied the effect of bursts and microbursts within the data centers, among others also for studying buffer utilization, and have shown significant differences between applications. These works [3, 4] have considered bursts and microbursts on millisecond and microsecond granularity, and have used (standard) switch functionality to analyze network data. Alizadeh *et al.* [5] define a mircoburst as a burst of traffic too short for traditional congestion control protocols to prevent. Given the view that our data presents into the network, this definition can be made more precise. Zhang *et al.* [2] 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.

Time is deceiving. A 100G link with 40% utilization using 100B packets means 50KB of data over 10μs, 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 [6] may not be able to capture transitional and momentary effects. Looking only at a watermark indication, e.g. as provided by counters on many NICs [7, 8] 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 using packet traces. In this work we explore use packet traces with sub-nanosecond timestamp resolution to explore the meaning of burstiness using different data centre 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? and (3) What is the effect of latency?.

To summarize, in this work, we make the following contributions:

- We collect a set of packet traces of different data centre applications, with sub-nanosecond capture timestamp resolution. Our dataset is open and contributed to the community.
- We explore the definition of burst using the different applications' packet traces,
- We explore the effect of latency and number of sources on burstiness, within the collected traces.
- We discuss generalizing our results and offer some 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. Last, we conclude in Section 6.

2 EXPERIMENTAL SETUP

Ten machines running Ubuntu server 16.04.5 and Linux kernel 4.4.0-131-generic are used in this experiment. Up to seven machines are used to run the benchmarks, two to capture traces, and one as a management node. All the machines have 10 Gbit test NIC installed, either Intel 82599ES NICs or Solarflare SFC9220 NICs. The NIC of each machine is recorded during each benchmark run. Benchmark machines

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

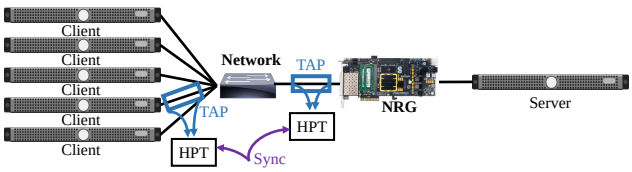

Fig. 1: The experimental setup used. One server is connected to the network through NRG. Traffic from the server and from one client is being captured using ExaNIC-HPT.

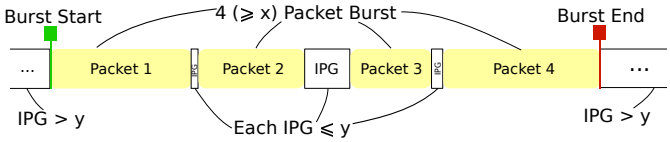

Fig. 2: How we define a burst in terms of x packets with an IPG of y ns. In the legend, xp$y$ns represent the burst parameters used.

are equipped with Intel Xeon E5-2637 v4 3.50GHz CPUs and 64 GiB of RAM.

Machines are connected using an Arista 7124FX switch. STP and LLDP are disabled to avoid uncontrolled traffic. Each machine is connected to the switch with a 3 m 10 G fiber. We use an ExaNIC HPT [9] to capture our traces. The ExaNIC 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.* [10], 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 both high resolution and maintaining precision even under high data rates.

We run two benchmarks, Memcached and Tensorflow, each with a single server/master and multiple clients/workers. Memcached is run using the Facebook ETC workload [11] and runs for 30 seconds. Tensorflow uses the MNIST dataset[2], and trains for 20,000 iterations. Configuration-specific information is recorded with the captured data.

3 DATASET

In this work we study burstiness covering three axes: application, latency and number of client machines. We collect a large dataset of experimental results for each of the applications. The results presented in this paper cover over 200 experiments, but our dataset includes additional related traces (e.g., client side only measurement). Our traces provide sub-nano second resolution, and approximately 30 ns clock synchronization between traces captured on different cards. We note that storage and run time where the two limiting factors of this study: as libpcap does not support sub-nano second timestamps, we wrote our own Python processing framework for this analysis.

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

For each experiment we collect beyond client and server side traces, also metadata, including benchmarking machines' logs (stdout, hardware configuration, syslogs), capture card output logs and NRG statistics logs.

Our dataset is available at [12]. The software environment used for capture is available at [13] and the software used for analysis is available at [14].

4 WHAT IS A BURST?

Here, we describe a model that will fit clusters running homogeneous workloads [11], and less so cloud environments running many user applications. We still believe that our work can provide relevant and interesting insights.

Bursts are traditionally defined as periods of high packet arrival rate [15]. Existing work on microbursts have taken a similar approach. Alizadeh *et al.* [5] defined a microburst as burst of traffic too short for traditional congestion control protocols to prevent. Zhang *et al.* [2] defined microbursts as small periods (less than 1 ms) of high utilization during which time more than a megabyte of data can arrive. Their data was sampled at a resolution of 25 µs.

We adopt a different definition for bursts, more appropriate for buffer sizing. Instead of looking and the *amount* of traffic, we consider the traffic's *density*, which partially resembles *link utilization*. To define bursts, and micro-bursts, we consider subsets of the question: How many packets become a burst? how close should these packets be to each other? and 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 will explore how best to select these parameters. This definition allows for detection of microbursts on the scale of individual packets. We refer to "bit time" rather than microsecond or nanosecond, as those depend on data rate. A given amount of data transmitted over a microsecond at 10 Gbit s^{-1} will be ten times more dense than the same amount of data transmitted at 100 Gbit s^{-1}. In other words, a gap of y ns corresponds to less burstiness at 100 Gbit s^{-1}

than at 10 Gbit s^{-1}. As our setup runs at 10 Gbit s^{-1}, for the rest of this paper we refer to y in units of nanosecond, 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 begin run the applications on our setup of 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 will fit multiple 1.5KB packets, and provides a "high level" view. Similar analysis was conducted at 1 µs and 10 µs, and is included in the repository. We then consider the 95th, 99th, 99.9th, and 99.99 percentiles within this distribution. These are shown for each application in table 1. Bandwidth is the link utilization multiplied by 10 Gbit s^{-1}.

The values in the table are intended to show the stark differences between the applications. Even for the same application, client-to-server and server-to-client directions will be very different, e.g. memcached's replies (server-to-client) will 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, 99th percentile and beyond. We use the IPG as an indicator, and require a burst to have a minimum number of consecutive packets: two, four or eight, all within (or under) the given IPG.

Figure 3 shows the length of incoming (client to server) bursts of memcached queries. We see a long tail (up to 77 packets), A significant number of the packets in the long tail are within 287 ns of each other, as shown by the prominence of the long tail on the 8 packet, 287 ns line. We can also see that back-to-back packets are very common: 75% of bursts at least two packets long, but within 287 ns of each other are exactly two packets long. Generally, we can use this analysis

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

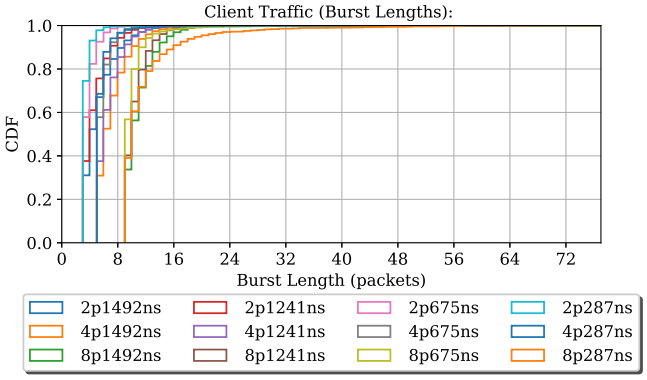

Fig. 3: The length of incoming bursts running Memcached. In the legend, *xpy*ns represent the burst parameters used.

to predict how much buffer space a particular channel should be allocated during a burst: for example, if we have seen 8 packets within 287 ns of each other, we should expect to need more than 8 packets *more* buffer space with a probability of about 10%.

Figure 4 shows how the incoming bandwidth depends on the parameters used. Here, we can see that the observed bandwidths are generally grouped by the IPG used to specify the burst size. 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. For the most part, we can see that bandwidth utilization during bursts is fairly low, although long-tails do exist.

Tensorflow packets did not arrive at consistent intervals: A typical arrival pattern of IPGs (including inter-frame spacing) is: 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.

The results of this analysis are shown in figures 5 and 6. First, we can see that burst lengths are significantly longer than in Memcached (figure 5), and can reach more than a thousand packet. This is both as memcached is memory bounded, and as Tensorflow's two-phase algorithm [16] means that workers sends large batches of data - the weights that are the results of training. For these reasons, Tensorflow is a far burstier application than Memcached. We can see that the longest bursts more than 1100 packets long, which is on a different order of magnitude from the Memcached results, where the bursts are at most 77 packets long. This suggests that different applications should see different buffer sizes.

Application	Packet Size (Median)	95th		99th		99.9th		99.99th	
		Utilization	IPG	Utilization	IPG	Utilization	IPG	Utilization	IPG
Memcached	156B	7.72%	1508ns	9.14%	1257ns	15.60%	691ns	0.34%	303ns
Tensorflow	1518B	99.6%	20.8ns	99.62%	20.8ns	99.64%	20.8ns	99.65%	20.8n

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.

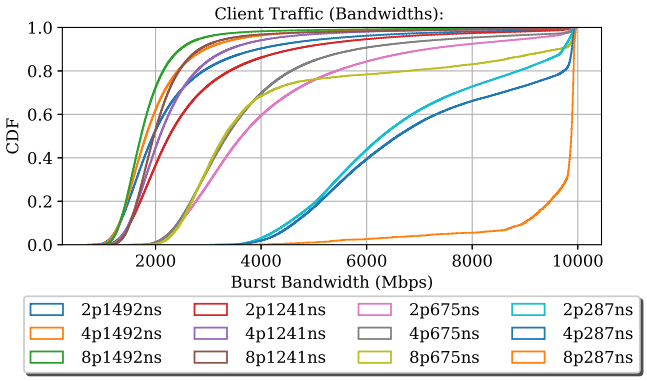

Fig. 4: The bandwidth of incoming bursts running Memcached. In the legend, *x*p*y*ns represent the burst parameters used.

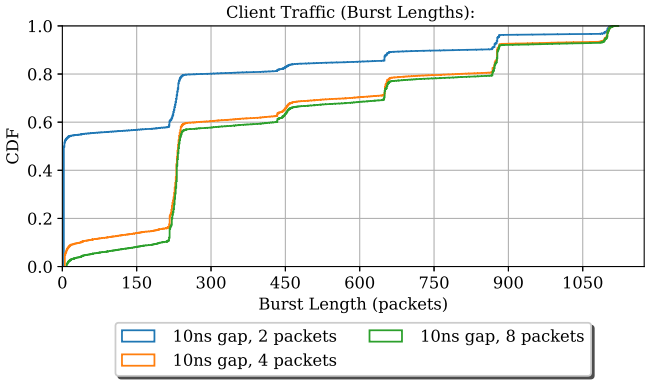

Fig. 5: Tensorflow incoming burst lengths.

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 much 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 sent from the server can be completely aggregated. There are also protocol asymmetries: the Memcached server typically generates longer responses than requests, meaning that outgoing bandwidth requirements are more taxing than incoming bandwidth requirements. These differing behaviours affect buffer sizing requirements.

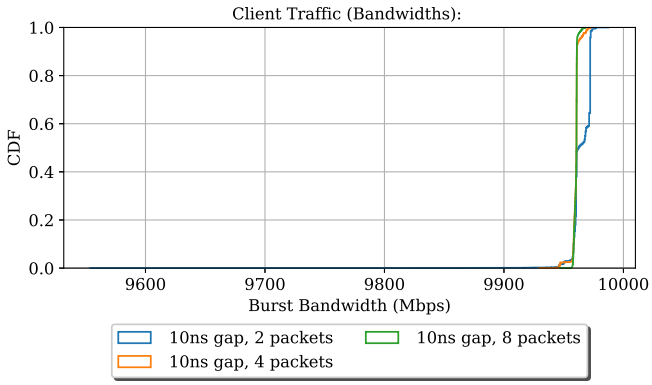

Fig. 6: Tensorflow incoming burst bandwidths.

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). An example of this is that for incoming packets, if we see a burst of 8 packets long, there is a 25% chance we will see more than 11 packets in the burst. However, in the outgoing traffic, if we see a burst of 8 packets long, there is a 25% chance there will be more than 15 packets in the burst. The bandwidth graph shows similar large 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 burstiness effect of aggregation of messages is stronger close to the sending machine.

4.5 How does latency affect burst size?

We used the NRG described earlier to control the latency visible to the applications to observe the effect of latency on bursts. Figure 9 shows this effect. We can see that at 50us latency, there is a small difference in the burst lengths: in

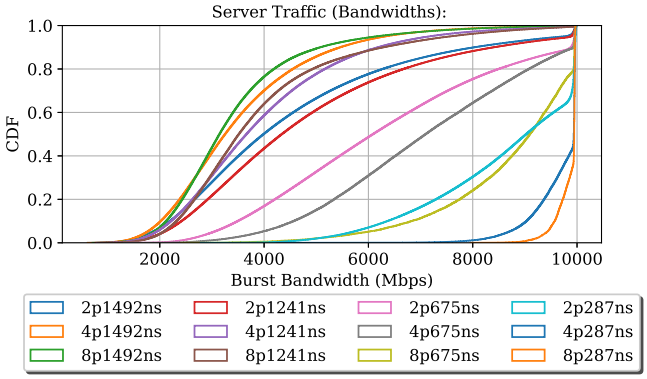

Fig. 7: The outgoing burst bandwidth running Memcached. In the legend, *x*p*y*ns represent the burst parameters used.

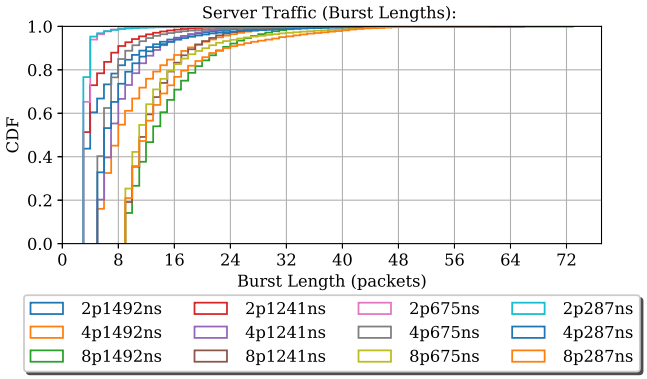

Fig. 8: The outgoing burst length running Memcached. In the legend, *x*p*y*ns represent the burst parameters used.

particular, bursts are less likely to be very long. In terms of buffer design, this suggests that switches at different locations within the network should have different buffer sizes. For example, this suggests that a ToR switch should require more buffer-space per connection than a switch connecting multiple racks together.

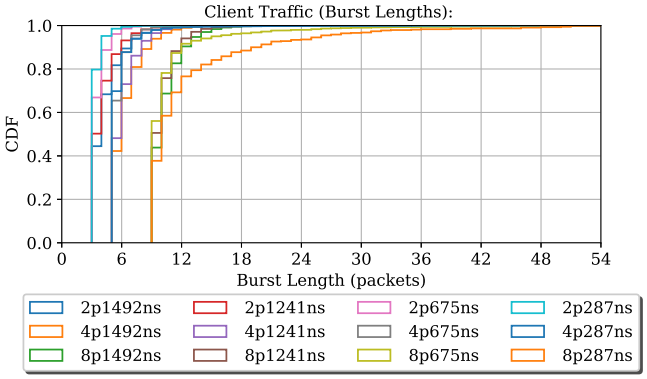

Fig. 9: The length of incoming bursts in Memcached with 50 µs of artificial delay. In the legend, *x*p*y*ns represent the burst parameters used.

5 DISCUSSION AND RELATED WORK

In this work, we are trying to answer the question *What is a burst?* in the context of current day data center applications. We focus our attention on application-level bursts, meaning we consider one application at a time and not a mix of applications. 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 micro-bursts at 10G, and can be scaled to higher bandwidths. Others have used different definitions of microbursts. For example Zhang *et al.* [2] 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. Further, the 25 µs resolution data used by Zhang *et al.* means their approach is limited to window sizes large enough for hundreds of packets at 10G, far 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 as it relies only on the IPG between packets and a static packet count. There remain questions about what the most useful measure of burst size is: in terms of packets or bits, and the answer to this question may well depend on the device, e.g., if buffer is assigned on fixed-size units, or if only bus-width alignment affects buffer size.

Burst size should not dictate buffer size on it's own. Many other factors come into play, such as resource availability,

QoS guarantees, and SLAs. Nevertheless, burst size is an important indicator for buffer sizing as it highlights the peak throughputs network devices can expect.

Limitations of the study. In order to achieve reproducibility, our study has limited scale. We run our experiments with at most seven machines, and only consider data center traffic. Further, we only run a small number of applications: not representative of today's heterogeneous data center cloud environment, not to mention 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 use here. Finally, our work does not have network-wide visibility: our conclusions are drawn exclusively from the perspective of the server.

Future Work. Our work contains relevant information for data center buffers. Internet traces such as CAIDA [17] provide an opportunity for similar analysis of Internet traffic. Traces from real data centers also provide an opportunity for further analysis. However, they are largely lacking due to privacy concerns among other things. Facebook have released a trace [11], 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

In this work, we have presented an analysis of the burstiness of two data center applications: Memcached and Tensorflow. Each application shows it's own characteristics relevant to its buffer size requirements. We found that Tensorflow exhibits far burstier behaviour, with the median burst length approximately 16 times longer and argue that required buffer sizes are application dependent. Exploring the effect of server aggregation, we found that the server synchronization increases burst-related bandwidth spikes and argue that buffer sizes need not be symmetrical on ingress and egress ports. Finally, we explore the effect of latency on burst size using memcached and found that additional latency decreased burstiness. Thus, we argue that switches handling low-latency paths (e.g. top-of-rack switches) should have larger buffers per flow than those handling higher latency paths (e.g. core switches). 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.

REFERENCES

[1] N. McKeown, G. Appenzeller, and I. Keslassy, "Sizing Router Buffers (Redux)," *ACM SIGCOMM computer communication review*, pp. 69–74, 2019.

[2] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proceedings of the 2017 Internet Measurement Conference*, pp. 78–85, ACM Press, 2017.

[3] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, (New York, New York, USA), p. 267, ACM Press, 2010.

[4] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09*, (New York, New York, USA), p. 202, ACM Press, 2009.

[5] M. Alizadeh, A. Greenberg, D. Maltz, and J. Padhye, "Data Center TCP(DCTCP)," *ACM SIGCOMM computer communication review*, vol. 40, no. 4, pp. 63–74, 2010.

[6] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, 2015.

[7] Solarflare, "Solarflare Server Adapter User Guide," 2014.

[8] Intel, "Intel Ethernet Controller X710/XL710 and Intel Ethernet Converged Network Adapter X710/XL710 Family: Linux* Performance Tuning Guide," 2016.

[9] Exablaze, "ExaNIC HPT."

[10] N. Zilberman, M. Grosvenor, D. A. Popescu, N. Manihatty-Bojan, G. Antichi, M. Wójcik, and A. W. Moore, "Where has my time gone?," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10176 LNCS, pp. 201–214, 2017.

[11] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, p. 53, 2012.

[12] J. Woodruff, A. W. Moore, and N. Zilberman, "Measuring Burstiness in Data Center Applications: Dataset." https://www.cl.cam.ac.uk/research/srg/netos/projects/latency/buffer2019/, 2019.

[13] J. Woodruff, "Capture environment." https://github.com/j-c-w/BandwidthPerf.

[14] J. Woodruff, "Trace analysis environment." https://github.com/j-c-w/EXPCAP_Process.

[15] W. E. Leland and D. V. Wilson, "High time-resolution measurement and analysis of lan traffic: Implications for lan interconnection," in *IEEE INFCOM '91. The conference on Computer Communications. Tenth Annual Joint Comference of the IEEE Computer and Communications Societies Proceedings*, pp. 1360–1366 vol.3, April 1991.

[16] "Tensorflow Achitecture." https://www.tensorflow.org/guide/extend/architecture.

[17] "CAIDA." http://www.caida.org/.