TCP Proactive Congestion Control for East–West Traffic: the Marking Threshold

Renan Fischer e Silva[a,b,*], Paul M. Carpenter[a]

[a] *Barcelona Supercomputing Center–Centro Nacional de Supercomputación (BSC–CNS)*
[b] *Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*

Abstract

Various extensions of TCP/IP have been proposed to reduce network latency; examples include Explicit Congestion Notification (ECN), Data Center TCP (DCTCP) and several proposals for Active Queue Management (AQM). Combining these techniques requires adjusting various parameters, and recent studies have found that it is difficult to do so while obtaining both high throughput performance and low latency. This is especially true for mixed use data centres that host both latency-sensitive applications and high-throughput workloads with east–west traffic such as Hadoop.

This paper studies the difficulty in configuration, and characterises the problem as related to ACK packets. Such packets cannot be set as ECN Capable Transport (ECT), with the consequence that a disproportionate number of them are dropped. The same issue can affect other non-ECT-capable traffic that may co-exist on the network. We explain how this behavior adversely affects throughput, and propose a small change to the way that non-ECT-capable packets are handled in the network switches. Using NS–2 simulation, we demonstrate robust performance for modified AQMs on a Hadoop cluster, maintaining full throughput while reducing latency by 85%. We also demonstrate that commodity switches with shallow buffers are able to reach the same throughput as deeper buffer switches.

Finally, we explain how both TCP using ECN and DCTCP can achieve the best performance using a simple marking threshold, in constrast to the current preference for relying on AQMs to mark packets. Overall, we provide recommendations to network equipment manufacturers, cluster administrators and the whole industry on how best to combine high-throughput and latency-sensitive workloads. This article is an extension of our previous work [1], which was published in Proceedings of the 19th IEEE International Conference on Cluster Computing (CLUSTER 2017).

Keywords: AQM, ECN, DCTCP, Throughput, Latency

1. Introduction

Once large data center operators have ownership of their network, they can optimize their end-to-end network connections. Doing so offers the potential to reduce latency, without degrading throughput. Over time, multiple such solutions have been proposed, for example the well-known DCTCP [2], an extension of the TCP protocol that reduces network latency and buffer utilization, without degrading throughput. In general, DCTCP is considered to be particularly promising, and the details of its deployment in production environments are being extensively discussed [3].

In certain settings however, recent studies have found that attempts to reduce latency often cause a degradation in throughput or performance [4, 5]. In particular, workloads with fully-distributed traffic, long-lived flows and high throughput requirements tend to fill up the buffers of the network equipment. Attempts to control buffer utilization have been found to reduce overall performance [4, 5]. This paper investigates why, and it proposes a solution.

Figure 1 demonstrates how Hadoop job execution time is affected when network latency is controlled using classical TCP extended with either ECN or DCTCP, both relying on Active Queue Management (AQM) to mark ECT-capable packets ("how it is"). It also shows how the performance "should be" if congestion control were performed by marking ECT-capable packets using a true marking scheme, as proposed in this paper. This figure was generated using the methodology of Section 3.

Although Hadoop is a network throughput-sensitive workload with less stringent requirements for network latency, there is an increasing interest in running batch

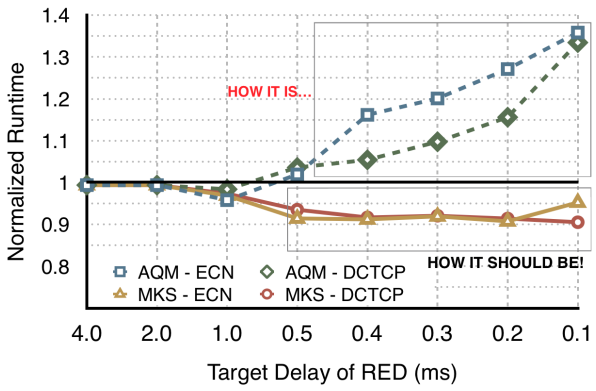


Figure 1: Hadoop job execution time affected by Active Queue Management

*Corresponding author
Email addresses: renan.fischeresilva@bsc.es (Renan Fischer e Silva), paul.carpenter@bsc.es (Paul M. Carpenter)

and interactive workloads concurrently on the same cluster. Doing so maximizes system utilization, to obtain the greatest benefits from the capital and operational expenditures. Recent studies have analysed how to reduce latency on systems with high-throughput workloads to enable heterogeneous classes of workloads to run concurrently on the same cluster [4]. Also, numerous MapReduce distributions are appearing with the aim of providing low-latency services, which may in future share the same infrastructure as Hadoop on a heterogeneous cluster with controlled latency [6]. As recently pointed out, 46% of IoT applications have low latency requirements on the order of seconds, or even milliseconds [7].

Current network switches offer much higher buffer densities due the employment of SDRAM memory. Many new solutions are targeting expensive equipment with deep buffers, in comparison with what was offered a few years ago. For example, not so long ago, a switch offering 1 MB of buffer density per port would have been considered a deep buffer switch [8]. New products are arising and with them, a buffer density per port 10× larger [9]. All this can make the Bufferbloat problem [10] even worse, with latency on these networks reaching up to tens of milliseconds for certain classes of workloads.

The shuffle phase of Hadoop, which involves an all-to-all communication among servers, presents a stressful load on the network. Recent tracers from Facebook show that the shuffle time varies between 50% and 70% of the total runtime [11]. Less computation and more communication leads to the network infrastructure constantly being the bottleneck to develop new type of solutions. In parallel with the increase in the capability of network switches, Hadoop has evolved from a batch-oriented workload to a more responsive and iterative type of framework. Currently it presents many different flavors and distributions, and reducing its latency has become of interest to the industry to allow new types of workloads that would benefit from the analysis capability of Hadoop and much more iterative solutions [4, 5]. For that, the network latency on current Hadoop clusters has to be decreased.

Our previous work was the first to investigate why earlier work had failed to maintain high-throughput while reducing the latency on clusters with throughput requirements such as the MapReduce framework and Hadoop [1]. These clusters present specific traffic patterns such as data and acknowledgement packets sharing the same bottleneck egress ports on typically oversubscribed links. This paper brings new results that confirm and strengthen the findings of our previous work.

This paper also extends the discussion to present recommendations to network equipment manufacturers and cluster administrators on how to reduce network latency on Hadoop clusters without degrading performance. We provide results that show the impact on Hadoop job execution time. We aim to make it easy to understand the problem and wish to open new discussions and promote research towards new solutions. We present experimental results in terms of cluster throughput and network latency.

In short, our main contributions are:

1. We analyse why extensions of TCP intended to reduce latency, e.g. ECN and DCTCP, fail to provide robust performance and effortless configuration.
2. We characterize the scenarios that provoke this problem and propose a small change to the way that non-ECT-capable packets are handled in the network switches.
3. We evaluate the proposed solution in terms of cluster throughput and network latency, as well as its expected impact on Hadoop job execution time.
4. We provide a set of recommendations to network equipment manufacturers and cluster administrators in order to benefit from this work.

The rest of the paper is organized as follows. Section 2 describes the problem and its solution. Section 3 describes our infrastructure and methodology and Section 4 presents the evaluation and results. Based on these results, Section 5 distills the most important recommendations. Section 6 compares our approach with related work. Finally, Section 7 concludes the paper.

2. The Problem and Motivation

Network transport protocols, such as TCP, traditionally signal congestion to the sender by dropping packets. This mechanism is simple, but dropped packets cause timeouts and packet retransmissions, which reduce throughput. Recent extensions, such as Explicit Congestion Notification (ECN) and Data Center TCP (DCTCP), signal congestion by marking packets instead of dropping them (as explained in Section 2.1). And this idea was not wrong!

When DCTCP was originally proposed [2], it was evaluated using a simple marking threshold. Although the marking threshold was, we believe, one of the key aspects of DCTCP, it was considered to be straightforward and was not debated enough. The authors mimicked the behavior of a simple marking threshold on an RED-capable switch by setting the Random Early

Discard (RED)[12] minimum and maximum intervals to the same value, and found that a value of 65 packets was necessary and sufficient to reach the full throughput of a 10 Gbps link. This meant that DCTCP could be implemented on existing and commonly available RED-capable switches.

The problem is that RED and other AQM queues that support ECN treat ECN Capable Transport (ECT)–capable packets differently from non-ECN-capable packets. The ECT-capable packets can be marked to indicate congestion, but in the same situation the non-ECT-capable packets would be *dropped*.

2.1. A deeper look at TCP packet marking

The main role of the network switch buffers is to absorb burstiness in packet arrivals, which are common in data center networks. A recent study from Cisco found that deep (large) buffers help the switches to better absorb such burstiness. For Big Data applications such as Hadoop, Cisco found that the second most important characteristic of the network, after availability and resiliency, was its ability to handle traffic bursts [13].

TCP connections will greedily use the available buffering on their network path, and persistently full deep buffers can cause a problem known as Bufferbloat [10]. For this reason, throughput-intensive applications, such as batch workloads like Hadoop, should not share the same infrastructure as low-latency applications, such as SQL or SQL in Hadoop.

The increasing latencies of Data Center networks has become a major problem, and with that, DCTCP has gained attention. A recent study [3] that extensively debated the most common pitfalls in DCTCP deployment pointed out that TCP and DCTCP traffic should never co-exist on the same infrastructure, because, while DCTCP data packets are ECT-capable and can be marked, classical TCP packets are not, so they will be disproportionally dropped. The authors even identified a possible problem with opening new connections. Since SYN packets are not ECT-capable, congestion could cause excessive dropping of SYN packets, which would prevent the establishment of new connections.

Targeting Hadoop clusters, recent studies used ECN and DCTCP in an attempt to improve network latency without degrading throughput or performance [4, 5]. In the latter study, the authors were able to provide useful configurations, but fine-tuning the AQM queues was considered to be non-trivial.

After carefully investigating snapshots from the egress port of network equipment at the queue level, we finally understood why previous work had failed to achieve high throughput and low latency for Hadoop.

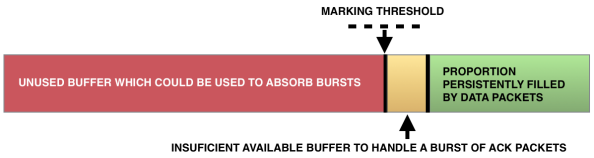


Figure 2: Typical representation of a network switch queue in a Hadoop cluster

Figure 2 illustrates the queue occupancy during a period of high utilization. Over time, TCP adapts to packet marking, causing data packets to persistently fill the queue capacity to some point just below the marking threshold. Bursts of data packets will be marked but not dropped, whereas non-ECT–capable packets (SYNs, ACKs and SYN-ACKs) would be dropped. This includes ACKs with the ECE bit set, which are used to echo congestion experienced back to the sender, and are necessary to adapt to congestion. The worst problem happens when a full TCP sliding window of ACK packets is dropped, which will trigger a TCP Retransmission Timeout (RTO), causing the sender's congestion window to be reduced to a single packet, limiting throughput.

We found that the solution was to use a true marking scheme that marks all ECT-capable packets once the threshold is crossed, and that does not drop any packets unless the buffer is full. By using a true simple marking threshold instead of mimicking one using an AQM, senders reduce their send rate proactively, keeping TCP's typical small-scale sawtooth behavior. The throughput of the network is maximised because there is much lower overhead for retransmitting packets. Another change is to measure the thresholds in bytes rather than numbers of packets, the latter being typical of RED implementations. This change is useful because ACK packets are short (typically 150 bytes), so there is usually enough space in the buffers to fit arriving ACKs.

The reason this problem has not been detected before is that previous experiments have considered simple topologies with many senders and *one* receiver. Such experiments create a network bottleneck in order to evaluate how TCP and DCTCP handle congestion, but they do not involve sharing of egress ports by data and ACK packets. We also investigated the reason for recent studies and evaluations finding that DCTCP outperformed TCP extended with ECN. We found that Partition/Aggregation workloads typically have network traffic dominated by short query messages. For short messages, for which the TCP congestion window never grows significantly, a severe cut of the

Table 1: ECN codepoints on TCP header

Codepoint	Name	Description
01	ECE	ECN-Echo flag
10	CWR	Congestion Window Reduced

Table 2: ECN codepoints on IP header

Codepoint	Name	Description
00	Non-ECT	Non ECN-Capable Transport
10	ECT(0)	ECN Capable Transport
01	ECT(1)	ECN Capable Transport
11	CE	Congestion Encountered

congestion window on the first indication of congestion can significantly decrease throughput.

Currently, it seems that data centers no longer consider using TCP extended with ECN. In a recent evaluation [3], the authors considered the only possibility for ECT-capable flows to be DCTCP and that TCP and DCTCP traffic cannot share the same infrastructure. In this paper we do present results using TCP extended with ECN, which we refer to as TCP–ECN, and find that in a specific situation it actually outperforms DCTCP.

When running Hadoop, the shuffle phase, which accounts for 50% to 70% of the total execution time [11], will engage a large part of the cluster, if not the whole cluster, in many-to-many communication. Employing either TCP–ECN or DCTCP with misconfigured AQM to mark ECT-capable packets will cause data and acknowledge (ACK) packets to share the same bottlenecks, and at minimal pressure on the buffers, packets that are not ECT-capable will be dropped. This effect can be devastating for TCP as new connections will be prevented from being established [3] and ACKs will be constantly dropped, significantly degrading the cluster's throughput.

2.2. Proposed and evaluated solutions

We propose two distinct solutions. The first is to modify the AQM implementation to allow an operational mode that, if ECN is enabled, protects packets whose ECE-bit is set in the TCP header. As seen in Table 2, current AQM implementations only check for an ECT(0) or ECT(1) codepoint in the IP header, in order to decide whether to mark or early drop the packet. If an ECT(0) or ECT(1) codepoint is found, the CE-bit is marked so a replied ACK can echo the congestion experienced back to the sender by setting the ECE-bit set in the TCP header. Protecting packets that have the ECE-bit set means that ACKs that echo congestion experienced back to the TCP sender are protected from early drop. The same behavior will also protect SYN and SYN-ACK packets, which are necessary to initialize a new TCP connection. When ECN is configured, SYN packets have their ECE-bit marked in their TCP header to signal an ECT-capable connection. SYN-ACK packets are replied having both ECE and CWR bits set by the receiver so that the sender can finally enable an ECT-capable connection. In short, when ECN is configured, the first proposal will ensure that not only ECT-capable packets, but also SYN, SYN-ACK and ACK packets that signal congestion (with the ECE-bit set) will not be early dropped. As we demonstrate with our results this approach achieves lowest latency while also alleviates the performance loss on cluster throughput.

Our second proposal is to simply implement a true marking threshold on switches, independently of the buffer density per port. This solution will allow cluster throughput to be improved beyond the baseline of a DropTail queue. While the translated latency of this approach will be slightly higher than our first proposal, cluster throughput is maximized even on commodity switches which offer shallow buffer density per port. The next section describes the experimental environment to evaluate these proposals.

3. Methodology

This section describes the experimental methodology for our work. The NS–2 simulator has been extended with CoDel [14] and DCTCP [15] implementations and is driven by the MRPerf MapReduce simulator [16]. The NS–2, MRPerf, CoDel and DCTCP implementations are open source, so using the parameters described in the next subsection, this simulation methodology has the advantage that it can be easily reproduced by independent researchers and future work can be carried out on it.

The topology selected for this work was the leaf–spine architecture [17], illustrated in Figure 3, which seems to be recommended for Hadoop, as seen in various references for cluster design [8, 18, 19].

Each leaf switch is connected to the spine switch using a single 10GbE link. The over-subscription ratio on the access layer is equal to 2:1. This matches cluster design recommendations for MapReduce clusters, which indicate that the over-subscription ratio in the access layer should be no greater than 4:1 [13, 20]. We

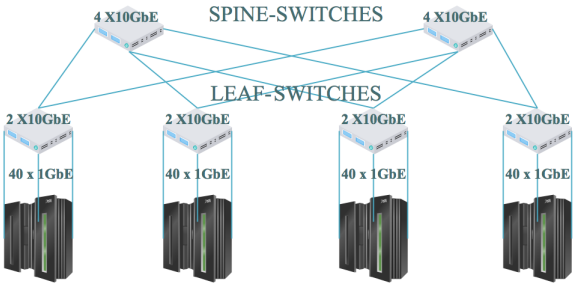


Figure 3: Leaf–Spine Cluster Topology

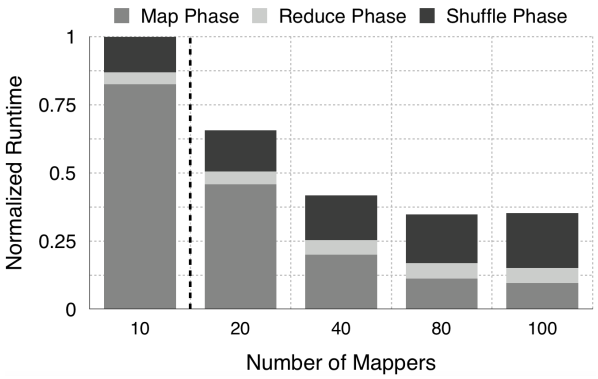


Figure 4: Proportion of time in Map, Reduce and Shuffle phases (normalized to 10 mappers)

also used the multiPath option from NS–2, which simulates Equal Cost Multi-Path (ECMP) routing through two equal cost routes. The ECMP feature is essential for a representative analysis of this cluster topology, which offers multiple routes and loaded over-subscription.

We provide results for both shallow and deep buffer switches using two different values for the queue sizes: 200 packets or 2000 packets per port. For data packets using the maximum payload size of 1500 bytes, these queue sizes translate to 293 KB or 2.43 MB respectively.

Table 3 shows the configuration of the simulated workloads using the MapReduce simulator MRPerf. One node was reserved for Hadoop housekeeping, to serve as namenode and jobtracker, with the remaining nodes used as worker nodes for processing map and reduce tasks. A single Terasort job is configured to sort 6.4 GB (random elements) with 100 mappers. Terasort is a popular batch benchmark commonly used to measure MapReduce performance on a Hadoop cluster. In order to ensure a representative benchmark, we characterized the shuffle phase as shown in Figure 4, which shows the amount of time spent on computation (map and reduce phases) and communication (shuffle phase), normalized to the case with 10 mappers and and fixing the number of reducers to the recommended (159 nodes), which is the number of workers in order to use the full system [**?**]. As can be seen from Figure 4, using 100 mappers allows a short completion time, and it is consistent with a study of traces obtained at Facebook, which shows that most of the jobs were small and that the shuffle phase represented more than 50% of the execution time of a job [11]. Shuffle is considered to be the MapReduce phase with greatest stress on the network because of its all-to-all communication between mappers and reducers. The amount of communication, most of which is in the shuffle stage, is close to proportional to the workload size, but it increases as the network becomes the bottleneck. Since the communication pattern is also repetitive, we can obtain representative figures using only one task, as the network and cluster utilization were proportionally designed based on available tracers.

MRPerf was recently introduced as an option for researchers that wish to carry out research of Hadoop workloads using the well-known and trusted NS–2 packet-level simulator. MRPerf has been extensively evaluated [21], and although it has limitations in precisely measuring some steps from the MapReduce framework, MRPerf is still recognized to achieve high accuracy for the simulation of the impact of network topologies and also for simulating the behaviors related to underlying networks. Our main contribution from this work is not related to the computation phase of MapReduce, but to its shuffle phase, which involves data movement across the cluster. The real gain for the MapReduce runtime will depend on the exact proportions of time spent in computation and communication.

Table 3: Simulated Environment

Category	Parameter	Value
Simulated hardware		
System	Number nodes	160
	Number racks	4
Node	CPU	Intel Xeon 2.5 GHz L5420
	Number cores	2
	Number processors	2
Network	Each node	*1GbE:* 1 —
	Each leaf switch	*1GbE: 40 10GbE:* 2
	Each spine switch	— *10GbE:* 4
Buffers	Commodity switches	200 packets - max 293 KB per port
	Expensive switches	2000 packets - max 2.43 MB per port

Table 4: Auto Random Early Detection Settings

Target delay	1 GbE thresholds	10 GbE thresholds
100 µs	12.5–37.5	125–375
200 µs	25–75	250–750
300 µs	37.5–112.5	375–1125
400 µs	50–150	500–1500
500 µs	62.5–187.5	625–1875
1 ms	125–375	1250–3750*
2 ms	250–750	2500*–7500*
4 ms	500–1500	5000*–15000*

Table 5: Controlled Delay (CoDel) Settings

Aggressiveness level	Target delay	Interval	Reference
1	500 µs	20 ms	[22]
2	800 µs	1.5 ms	[5]
3	400 µs	1 ms	[5]
4	300 µs	0.75 ms	[23]

We considered two AQMs to mark ECT-capable packets, which are RED and CoDel. Implementations of Random Early Detection are found on Linux, Solaris, and FreeBSD [24]. It is also implemented by network equipment vendors including Cisco [25] and Juniper Networks [26]. RED uses configurable thresholds to decide when to mark packets when combined with ECN, and drops packets based on a probability that grows with the queue occupancy. The first threshold is when the queue starts to mark or early drop packets, both based on probability. The second threshold defines when a packet will be either always marked or always dropped, with probability of 100%. Table 4 shows the used settings for the autoRED configuration. Since the maximum capacity of deep buffers was limited to 2000 packets, some values will not produce an effect for the 10 GbE links. For 1 ms, the second threshold will not be reached, so early drops will still be done based on probability, which will not reach 100%. For 2 ms and 4 ms the egress queue will behave as a DropTail queue, never marking or early dropping packets.

Controlled Delay is recommended by the Bufferbloat initiative [10, 22]. The user needs to configure the target delay, which is the tolerable delay-per-packet from the time it is queued until it is transmitted, and the interval, which is how often the delay-per-packet of transmitted packets is evaluated. If any packet has a delay greater that the target, then the interval is shortened, otherwise it is reset at the end of its cycle. Table 5 shows the configured settings for the CoDel queue. On NS-2, RED offers the auto-configure mode that allows a clear swipe on its configuration values for experimenting multiple data points and different trade-offs. For CoDel, such trade-off is not as simple, as for example, the interval impacts on how rigorous the target delay cutoff is preserved. Based on recommended settings from literature, we sorted each pair of interval and target delay as from 1, which is pair of settings that strikes lesser buffering restrictions; to 4, which is the configuration that forces more buffering cutoff.

We modified both AQM queues to simulate, in addition to their normal behavior, the two operational modes described in the previous section. First, we protected all the packets whose ECE-bit is set in the TCP header. Finally, we repeated the same set of experiments expanding both AQM queues to correctly mimic a true simple marking scheme. We could identify the problem as related to the extra ACKs which are neither ECT-capable or have the ECE-bit set on their header. To characterize the problem, we repeated the same experiments and kept the drop capability on these queues. Yet, we also forced the queues to protect the following packets from an early drop: ECT-capable packets, packets that have ECE-bits set in the TCP header and all the remaining ACK packets. Therefore, all the following packets are protected from an early drop: all data packets (which are ECT-capable), all SYN and SYN-ACK packets (which will have ECE-bit on their TCP header) and all the ACK, whether or not they are echoing congestion experienced using the ECE-bit.

In short we provide results for either TCP–ECN and DCTCP flows using AQMs configured with ECN to protect the following packets from an early drop:

- **Default** behavior which protects only ECT-capable packets.
- **ECE-bit** which protects ECT-capable packets and packets which have ECE-bit set on their TCP header (SYN, SYN-ACK and a proportion of ACKs).
- **ACK + SYN** which protects ECT-capable, SYN, SYN-ACKs, and ACK packets, irrespective of whether or not they have the ECE-bit set in their TCP header.

The three performance metrics considered are: the *runtime* which is the total time needed to finish the

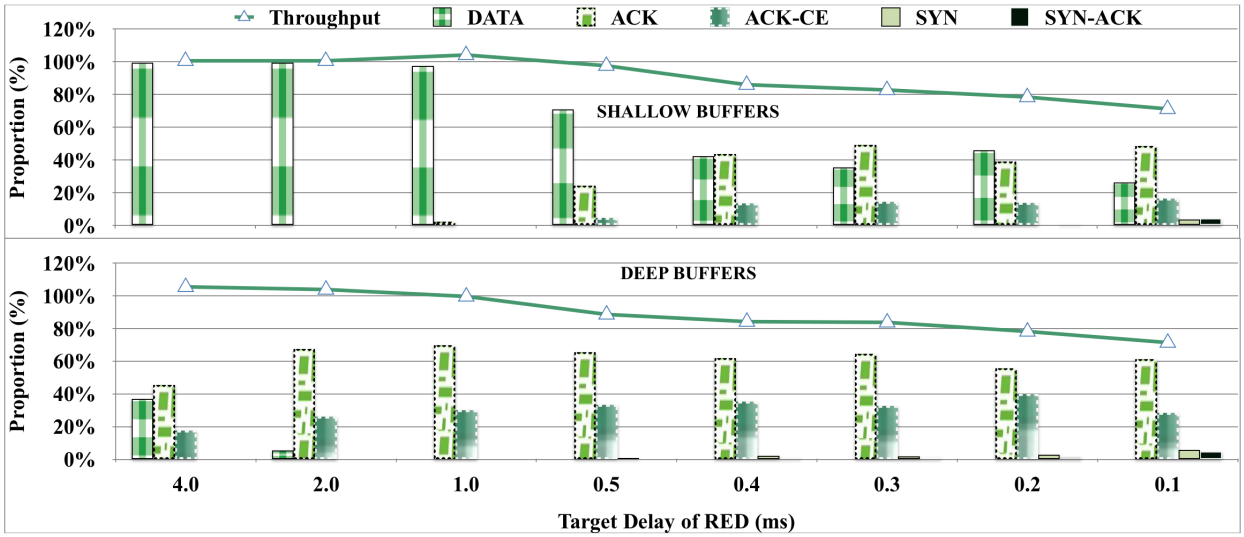


Figure 5: Proportion of total dropped packets (%) X Target delay of RED (ms) - using ECN

Terasort workload, which is inversely proportional to the effective throughput of the cluster; the *average throughput* per node and the *average end-to-end latency* per packet.

4. Results

This section presents the quantitative results in terms of the runtime, throughput and latency for Hadoop, using the methodology described in the previous section. Results are shown for both RED and CoDel, in order to demonstrate that the approach advocated in this paper is independent of the precise AQM mechanism.

4.1. Proportion of Dropped Packets

Section 2 explained that the difficulty in configuring proactive congestion control is related to the dropping of ACK packets. We start the results section by exploring this claim. In this respect, Figure 5 and Figure 6 show

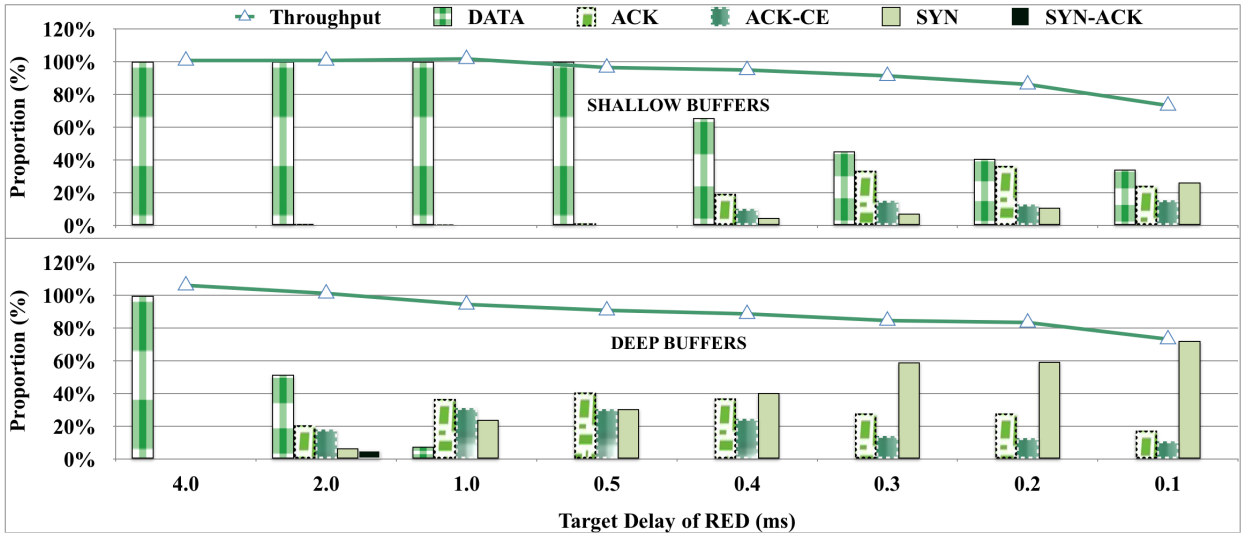


Figure 6: Proportion of total dropped packets (%) X Target delay of RED (ms) - using DCTCP

the distribution of dropped packets by type of packet (data, ACK, ACK-CE, SYN or SYN-ACK) as a function of the RED target delay, for ECN and DCTCP respectively. In each figure, the top plot is for shallow buffers and the bottom plot is for deep buffers. These results use standard RED queues. In order to provide context, the charts also present the measured throughput (the throughput plotted in these figures is discussed in the following subsection).

For tolerant proactive congestion control (high RED target delay), shown on the left-hand side of these figures, we see that, with the exception of deep buffers on ECN,[1] essentially all of the dropped packets are DATA packets. This is because TCP congestion control is operating normally. Since ACK packets are short (150 bytes), there is usually enough space in the buffers to fit arriving ACKs. On the other hand, when the sender increases its congestion window (on each round trip), and when the congestion window becomes too large, a burst of DATA packets is dropped when it reaches a congested buffer. The sender, however, will continue sending data, and when the receiver receives packets out of order, it will issue duplicate ACK packets, causing the sender to enter fast recovery mode. TCP throughput, also indicated on the figure, remains at 100%.

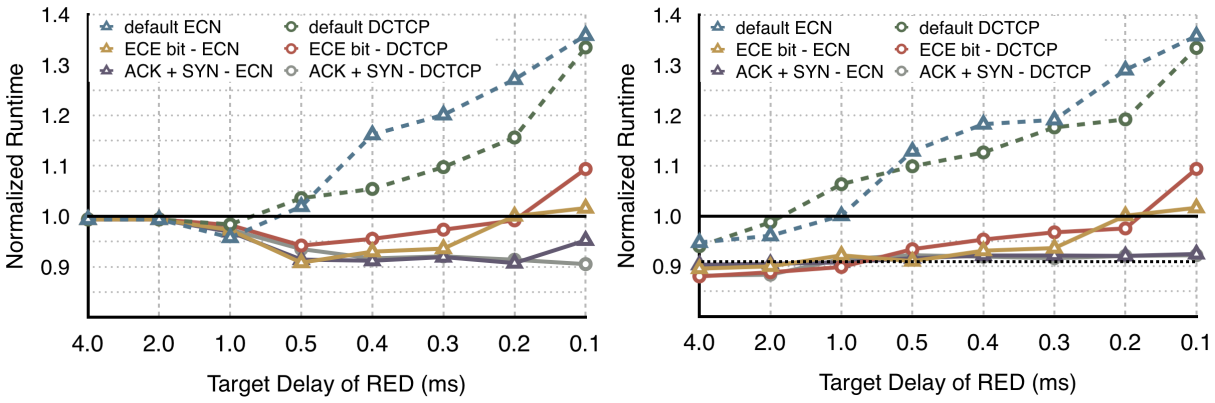
As the aggressiveness of the proactive congestion control is increased, moving to the right of the figures, the distribution of dropped packets changes. Specifically, a greater proportion of the dropped packets are ACK packets (with and without the ECE-bit set), SYN packets, or SYN-ACK packets. As explained in Section 2 and explored quantitatively in the next section, these packets are more important to the functioning of TCP congestion control.

There are some differences between ECN and DCTCP, which can be seen by comparing Figure 5 and Figure 6. The largest difference is that when proactive congestion control becomes active, ECN tends to drop mainly ACK packets whereas DCTCP tends to drop many more SYN and SYN-ACK packets. This is because DCTCP uses delayed ACKs, which tends to avoid bursts of ACK packets, which would be dropped under conditions of heavy network congestion. Especially when using deep buffer switches, we see that DCTCP has a larger proportion of dropped SYN packets (70% of all dropped packets), so the biggest issue is the opening of new connections.

For the next set of results, we repeated these experiments protecting only ACK packets with the ECE-bit set. In this case, we see that a large proportion of the dropped packets are ACK packets without the ECE-bit. The queue ends up on a state of prioritizaing DATA, ACK-ECE, SYN and SYN-ACK packets with low space to remain for normal ACKs that may arrive in bursts.

Finally, we also repeated these experiments with proactive congestion control done right, i.e. when protecting all ACK and SYN packets. We find in this case, that the absolute number of dropped packets is reduced dramatically, to about 1% of the prior value. In this case TCP congestion control exhibits normal behaviour, in that it probes the network to find out how many packets the network can carry, and it backs off on the first sign of congestion. Since congestion is signalled with ECN

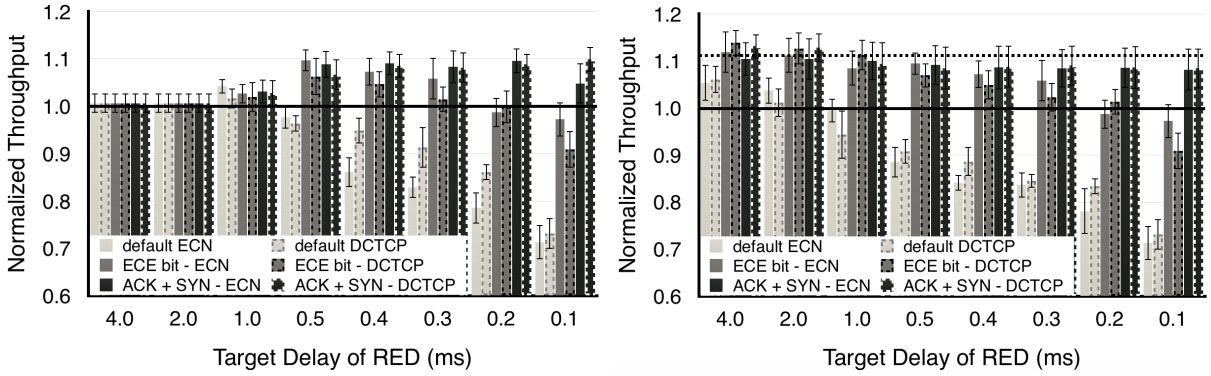
[1] For deep buffers on ECN proactive congestion control is already active even with the most tolerant RED target delay.



(a) Shallow Buffers

(b) Deep Buffers

Figure 7: Hadoop Runtime - RED



(a) Shallow Buffers

(b) Deep Buffers

Figure 8: Cluster Throughput - RED

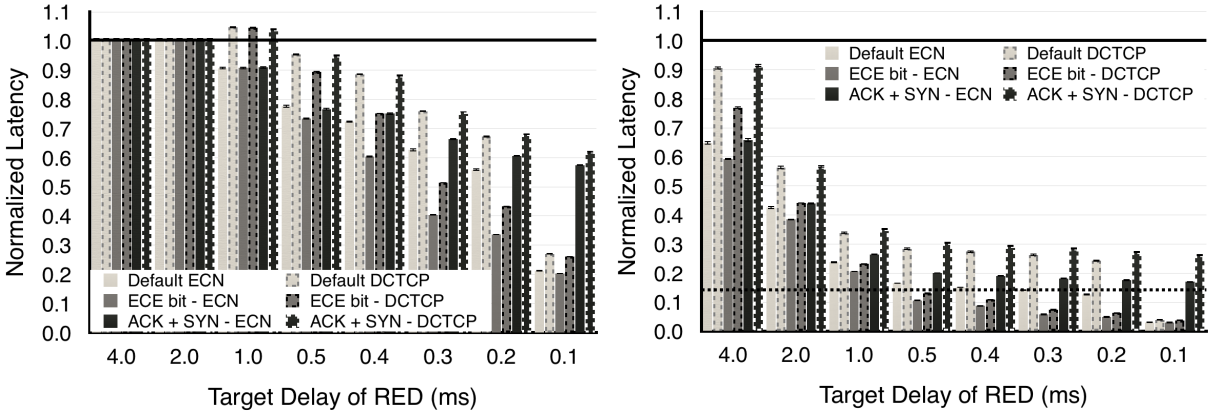
or DCTCP instead of packet drops, the network only rarely has to drop a packet. TCP throughput remains high and latency is reduced significantly. We explore runtime, throughput and latency in the following set of results.

4.2. Normalized runtime, throughput and latency

The following results are normalized relative to an ordinary DropTail queue. In the case of runtime and throughput, results are always normalized with respect to DropTail with shallow buffers. For these results, the dashed line on the deep buffer plots indicates the (better) runtime or throughput obtained using DropTail with deep buffers. In order to analyse the bufferbloat problem separately for deep and shallow switches, network latency is normalized to the latency of DropTail with the same buffer lengths. On the deep buffer results, we indicate with a dashed line the (much lower) latency obtained using shallow buffer switches.

4.3. Random Early Detection (RED)

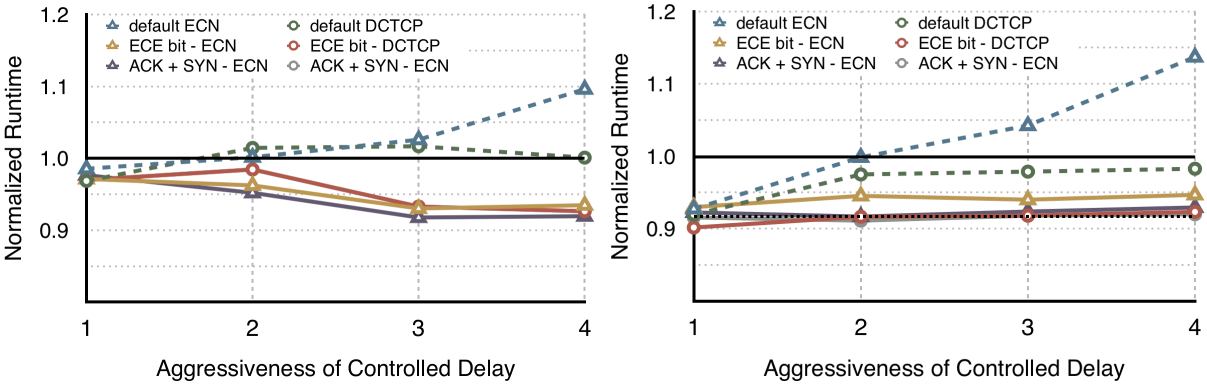
We start by presenting the effect of configuring the target delay of RED and how its different thresholds (see the previous section) affect Hadoop runtime for switches with shallow buffers. Figure 7a shows the runtime for shallow buffers and that for shallow buffers the best runtime is achieved either at a moderate target delay of 500 µs for both ECE-bit and ACK+SYN with ECN, or also using more aggressive settings to achieve the same with DCTCP. Comparing with Figure 8a we



(a) Shallow Buffers

(b) Deep Buffers

Figure 9: Network Latency - RED



(a) Shallow Buffers

(b) Deep Buffers

Figure 10: Hadoop Runtime - CoDel

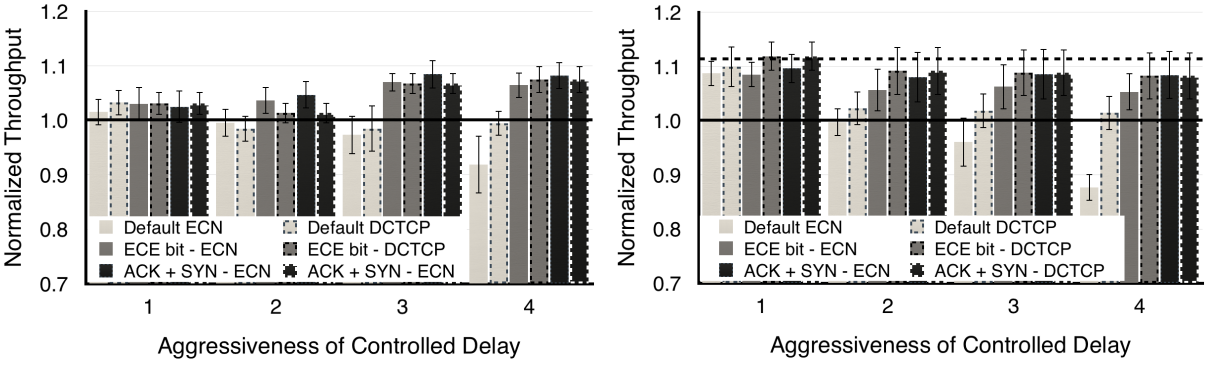
see how ACK+SYN was in terms of throughput, which increases by about 10% when target delay settings become aggressive. It shows that senders are able to control congestion if it is signaled soon enough. The best results and robustness of throughput is also translated to a network latency never lower than 50% compared to the baseline, as confirmed on Figure 9a.

For deep buffers, we start with Figure 8b. We can clearly see that as any congestion control is performed using ECE-bit or ACK+SYN cluster throughput achieves its maximum values using loose settings. As seen in Figure 9b, although the network latency was reduced by almost 60%, it is still about three times higher than the latency found on the DropTail queue of shallow buffer switches. The values to be considered should be the ones starting on 500 µs. Finally, Figure 7b shows Hadoop runtime reaching a robust 10% speed-up, which is about the same performance reached by the DropTail queue from deep buffer switches. Now we analyze the results using the modified CoDel.

4.4. Controlled Delay (CoDel)

Using a modified CoDel enabled us to verify that regarding how the logic of the queue decides when to mark and drop packets also influences the results obtained. Although there is some variation in comparison to what was obtained using RED, we also verified the robustness of both solutions using CoDel.

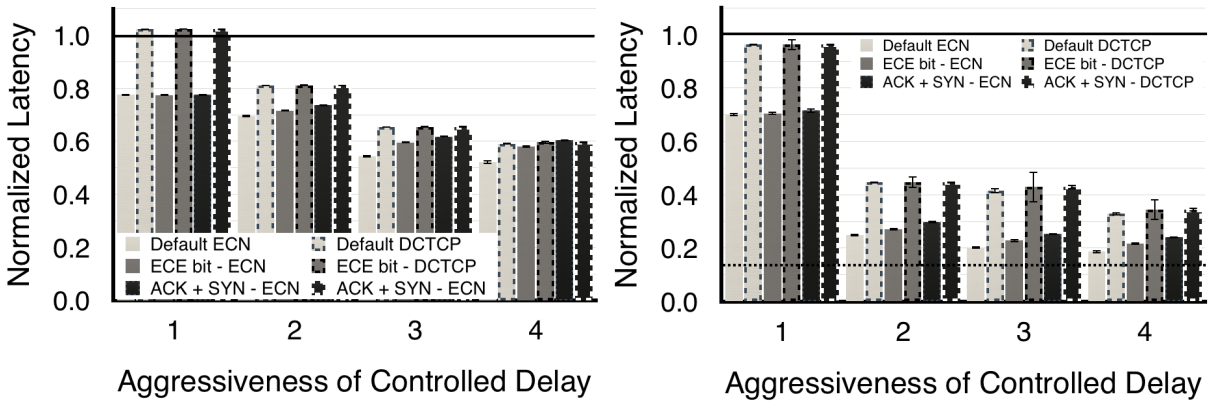
We start again by presenting the effect of delay control applied to CoDel as regarding the aggressiveness of the settings described on previous section. Figure 10a shows that ACK+SYN for ECN achieved the fastest



(a) Shallow Buffers

(b) Deep Buffers

Figure 11: Cluster Throughput - CoDel



(a) Shallow Buffers

(b) Deep Buffers

Figure 12: Network Latency - CoDel

Hadoop runtime, about 8% faster than the DropTail baseline. Cluster throughput was also stable as verified in Figure 11a, even used the most aggressive settings it remained above 5% improvements for both proposed solutions and all the flows. Analysing Figure 12a shows CoDel reduced its latency by half of what RED achieved, which translate on a network latency reduced by 40%.

As seen in Figure 10b, CoDel also has impacts on Hadoop runtime when more aggressive settings are used. The dashed line shows the best results which were from the larger DropTail queue and the proposals had a slightly performance degradation. The small drop on cluster throughput is also verified in Figure 11b. At last, the small reduction in throughput was compensated for by a reduction in network latency of about 80%, almost reaching the same network latency found on shallow buffer switches (pointed by the dashed line) in Figure 12b. Specially for CoDel, ECN solutions had much lower network latency than DCTCP solutions, which matches with results related by previous work (see related work).

4.5. Summary of Results

From the results described here we could verify that RED is much more sensitive to aggressive settings. It was translated to an increase in Hadoop runtime of about 35%, which was much higher than CoDel's worst performance loss of about 10%. Still, the performance degradation was importantly reproduced so we could also verify that independently from the egress queue logic utilized, the problem resides on the disproportional number of ACK packets which are early dropped. For CoDel, ECE-bit proposal is enough to achieve near the best results on terms of Hadoop runtime, cluster throughput and network latency. For RED, the implementation of a true marking scheme, which on our simulations was reproduced by protecting the packets that caused the performance loss (described on previous section), allow the network administrator to choose between the former mentioned ECE-bit, which is a really low latency solution that still early drops some ACK packet that do not have ECE-bit on their header; or the true marking scheme which offers a more conservative reduction on network latency, with no early drops. For commodity clusters particularly employing shallow buffer switches, it translates by expressive gains on cluster throughput and Hadoop runtime. Finally, for clusters employing deep buffer switches which already present improved throughput as their baseline performance, the task of configuring switches egress queues as an attempt to reduce network latency becomes much easier with no throughput degradation on misconfigured AQMs. Next section presents discussions and recommendations regarding our results.

5. Discussion and Recommendations

The results presented in this paper show that Hadoop can tremendously benefit from a true simple marking threshold. These results are exclusive to Hadoop, but we encourage future work to evaluate other types of workload that present the following characteristics:

- Use of ECN: ECN configured on TCP flows (either TCP–ECN or DCTCP) and switch queues configured to mark ECT-capable packets.

- East–west traffic pattern, which implies that data and ACK packets are often present in the same egress queue.
- Long-lived TCP flows with bursty communication.

Previous work pointed out that classical TCP and DCTCP flows should never coexist on the same infrastructure. We identified a worse problem when a controlled infrastructure having only ECN configured flows (either TCP–ECN or DCTCP) will collapse, due to a disproportional drop of ACKs. Redesigning the topology and organization of the cluster such that egress queues are never shared between data and ACK packets would make the problem disappear. The cluster topology and workload organization therefore play an important part in reproducing the problem described here, which explains why previous work failed to reproduce or identify this problem. On the other hand, MapReduce workloads were designed so that a large proportion of the cluster, or all of it, will be engaged in a fully distributed computation phase. This requires many-to-many communication among servers, in the so-called shuffle phase, in which the network will be consistently congested.

DCTCP has a strong premise, which is to maintain low latency while delivering high throughput, even higher than classical TCP. This was demonstrated in previous evaluation, and these results still hold for the circumstances in which it was evaluated. A workload composed of short TCP flows will benefit from a more gentle cut on the congestion window. TCP takes about one round trip to double its congestion window during its Slow Start phase. When a more aggressive congestion window cut happens on a short-lived flow, it might not have another round trip to return to a satisfactory congestion window to achieve full throughput. Therefore, DCTCP is necessary on workloads that follow the OLTP model (OnLine Transactions Processing), which is generally composed of short-lived TCP flows such as query messages (flows typically shorter than 1 MB). Since TCP–ECN presents a more aggressive congestion window cut, it can degrade throughput on such type of workloads.

For long-lived flows, of more than 1 MB, on DCTCP evaluation itself, there was virtually no difference in performance compared with TCP–ECN. We verified that for Hadoop, for which the network is often congested, TCP–ECN will benefit from a more aggressive cut on the congestion window, offering a lower latency than DCTCP.

A true simple marking schema simplifies the difficult task of configuring and tuning AQMs to work on MapReduce workloads. Yet, we identify different approaches that fit better for shallow or deep buffer switches. For shallow buffers switches, the more aggressive settings are useful to signal congestion as soon as possible. As the buffer density per port is limited, it is better to signal the congestion before it is necessary to drop packets. For deep-buffer switches, using moderate aggressiveness settings allows more data to be persistent on egress queues while they can still proactively signal congestion to the senders, which finally translates to an improved TCP throughput. We now distill our most important recommendations.

Recommendations for equipment vendors. Existing switches allow only limited configurability of the egress queues, and it is not possible for the network administrator to disable packet drops. This means that if AQMs are combined with ECN flows, it is not possible to achieve the best results in terms of throughput and network latency for the shuffle phase of a Hadoop infrastructure. The performance of Hadoop workloads would be improved if the switches offered an operational mode where the egress queues mark ECT-capable packets using a simple marking threshold while avoiding drops of non-ECT-capable traffic, and it may be possible to achieve via a firmware update. This option should also be offered for commodity switches, even if the switch offers a lower buffer density per port than 1 MB, as our results demonstrate its effectiveness for switches with a buffer density per port no higher than 300 KB. For Hadoop, ECN flows (TCP–ECN or DCTCP) will benefit from this operational mode where no packets are dropped until overflowing the egress queue limit while the congestion control is fully performed by TCP endpoints.

Recommendations for network administrators. Until a true simple marking threshold is implemented on network switches, we recommend the following two solutions.

Some high-end switches offer the option to forward non-ECT-capable traffic on the egress queue instead of dropping them, as we discuss in the next section (related work). Unfortunately, this option is only available on top-of-the-line switches which offer a very high buffer density per port. Although this solution will perform as well, in terms of throughput and network latency, as the results obtained in this work, it is unlikely to lead to a low-cost solution.

For switches that do not offer the option to forward non-ECT-capable traffic, the network latency can be reduced to an optimum level at the cost of a loss in

throughput. The performance of a standalone queue was out from the scope of this work, as it was already covered by previous work, described in the following section. Therefore, for commodity or legacy switches that offer AQM and ECN features, as an immediate low-latency solution, we recommend using a standalone AQM queue, disabling ECN from TCP flows and from the queue configuration. The next section covers the related work, which enrich our recommendations presented here.

6. Related Work

The original DCTCP paper [2] suggested that a simple marking threshold could be mimicked using switches that already support RED and ECN. More recent studies, such as a comprehensive study of tuning of ECN for data center networks [27] also recommended that switches would be easier to configure if they had one threshold instead of the two found on RED. They also recommended to use the instantaneous rather than the averaged queue length. They also pointed out the problem with SYN packets not being ECT-capable, but the problem with disproportional dropping of ACKs was not mentioned. Another recent study, which extensively discussed common deployment issues for DCTCP [3] pointed to the same problem that happens on a saturated egress queue when trying to open new connections.

None of these previous studies considered the use of a class of workload, such as Hadoop, that involves multiple long-lived TCP flows in multiple directions. Such workloads cause the same egress queue to contain large numbers of both data and ACK packets. As previously explained, on Hadoop, data is transferred across the whole cluster during its shuffle phase and that was why they were not able to identify the problem described on this work.

The performance of TCP–ECN and DCTCP on Hadoop clusters was recently investigated. Grosvenor et al. [4] evaluated the ability of TCP–ECN and DCTCP to reduce Hadoop's latency while running latency-sensitive services in the background. The goal of their work was to present a new method to classify the network traffic, for which the queue utilization was important, so some packets could jump the queues. In that work, TCP–ECN was considered to have a performance degradation of more than 2× the baseline. DCTCP achieved better results, but they still showed a performance degradation of 20% with respect to the baseline. We believe that the reason for such a difference between TCP–ECN and DCTCP was the fact that for TCP–ECN they set a much lower marking threshold using RED, equal to 40 packets, while for DCTCP they maintained it at 65 packets, as recommended in DCTCP's original paper [2]. We believe the basis for comparison between TCP–ECN and DCTCP should be the same marking threshold, as DCTCP already reduces its congestion window more gently. A more recent study [5], which compared TCP–ECN and DCTCP using identical thresholds of 70 packets, also showed a performance degradation of about 20%, with respect to the baseline, which used deep buffer switches. In that study TCP–ECN was considered to achieve a lower latency than DCTCP, showing that in a congested environment with long-lived TCP flows, both TCP–ECN and DCTCP can achieve similar throughputs, but the more aggressive cut in the congestion window in the case of TCP–ECN leads to a lower-latency solution.

A new feature is being offered on modern high-end switches which offer the possibility to forward non-ECT-capable traffic while having AQM configured to use ECN. This feature allows the non-ECT-capable traffic to bypass the AQM thresholds and grow until the egress queue limit [28]. Yet, this feature is only available on new high-end switches, which for example, have a buffer density per port near 10 MB. From the results presented in this work, we showed the benefits from a true simple marking threshold using the instantaneous queue length being implemented on switches with much lower buffer density per port. Once current network equipment receive such update it could translate to improvements in both throughput/performance and network latency, with an affordable solution achieved through a simple firmware update.

7. Conclusions

In this paper, we presented a novel analysis on how to reduce network latency on MapReduce clusters without degrading TCP throughput performance. We characterized the problem which previous work failed to identify. We demonstrated why it is inadvisable to use Active Queue Management to mark ECT-capable packets on MapReduce workloads. We presented results comparable to recent works that tried to reduce the network latency found on MapReduce clusters, and which failed to identify the real problem when DCTCP or TCP–ECN flows rely on AQMs to mark ECT-capable packets.

We also demonstrate that one true simple marking threshold not only simplifies the configuration of marking ECT-capable packets, but it also translates to a more robust solution. Doing so, we were able to avoid the

20% loss in throughput reported by previous work, and especially on commodity switches which offer considerably low buffer density per port, we even achieved a boost in TCP performance of 10%, in comparison to a DropTail queue. Yet, our gains in throughput were accompanied with a reduction in latency between 70% and 85%, depending on the considered buffer density per port. The results presented in this paper are not exclusive but can also expected to be reproduced on similar bases in other type of workload that presents the same characteristics described on the problem characterization found on our discussion and recommendations.

Finally, we showed that a true simple marking scheme should not only be supported in deep buffer switches. Commodity switches, as typically employed in MapReduce clusters, could also achieve promising results in terms of throughput and network latency. The results in this paper can help reduce Hadoop runtime and allow low-latency services to run concurrently on the same infrastructure.

8. Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007–2013) under grant agreement number 610456 (Euroserver). The research was also supported by the Ministry of Economy and Competitiveness of Spain under the contracts TIN2012-34557 and TIN2015-65316-P, Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272), HiPEAC-3 Network of Excellence (ICT- 287759), and the Severo Ochoa Program (SEV-2011-00067) of the Spanish Government.

References

[1] R. F. e. Silva, P. M. Carpenter, High throughput and low latency on Hadoop clusters using explicit congestion notification: The untold truth, in: 2017 IEEE International Conference on Cluster Computing (CLUSTER), 2017, pp. 349–353. doi:10.1109/CLUSTER.2017.19.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center TCP (DCTCP), in: Proceedings of the SIGCOMM 2010 Conference, SIGCOMM '10, ACM, New York, NY, USA, 2010, pp. 63–74. doi:10.1145/1851182.1851192.
URL http://doi.acm.org/10.1145/1851182.1851192

[3] G. Judd, Attaining the promise and avoiding the pitfalls of TCP in the datacenter, in: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), USENIX Association, Oakland, CA, 2015, pp. 145–157.
URL https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/judd

[4] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, J. Crowcroft, Queues don't matter when you can jump them!, in: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), USENIX Association, Oakland, CA, 2015, pp. 1–14.
URL https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/grosvenor

[5] R. F. E. Silva, P. M. Carpenter, Controlling network latency in mixed Hadoop clusters: Do we need active queue management?, in: 2016 IEEE 41st Conference on Local Computer Networks (LCN), 2016, pp. 415–423. doi:10.1109/LCN.2016.70.

[6] G. Mone, Beyond Hadoop, Commun. ACM 56 (1) (2013) 22–24. doi:10.1145/2398356.2398364.
URL http://doi.acm.org.recursos.biblioteca.upc.edu/10.1145/2398356.2398364

[7] MapR Takes Road Less Traveled to Big Data, https://davidmenninger.ventanaresearch.com/mapr-takes-road-less-traveled-to-big-data-1, accessed: 2017-01-26.

[8] A. Bechtolsheim, L. Dale, H. Holbrook, A. Li, Why Big Data Needs Big Buffer Switches. Arista White Paper, Tech. rep. (2011).

[9] Cisco, Network switch impact on big data Hadoop-cluster data processing: Comparing the Hadoop-cluster performance with switches of differing characteristics, Tech. rep. (2016).

[10] J. Gettys, K. Nichols, Bufferbloat: Dark Buffers in the Internet, Queue 9 (11) (2011) 40:40–40:54. doi:10.1145/2063166.2071893.
URL http://doi.acm.org/10.1145/2063166.2071893

[11] Y. Chen, A. Ganapathi, R. Griffith, R. Katz, The case for evaluating MapReduce performance using workload suites, in: 2011 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, 2011, pp. 390–399. doi:10.1109/MASCOTS.2011.12.

[12] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413. doi:10.1109/90.251892.

[13] Cisco Systems, Inc, Big Data in the Enterprise - Network Design Considerations White Paper, Tech. rep. (2011).

[14] Controlled Delay (CoDel) Active Queue Management NS-2 code, http://pollere.net/Codel.html, accessed: 2017-01-26.

[15] Data Center TCP NS-2 code, http://simula.stanford.edu/~alizade/Site/DCTCP.html, accessed: 2017-01-26.

[16] G. Wang, A. R. Butt, P. Pandey, K. Gupta, Using realistic simulation for performance analysis of Mapreduce setups, in: Proceedings of the 1st Workshop on Large-Scale System and Application Performance, LSAP '09, ACM, New York, NY, USA, 2009, pp. 19–26. doi:10.1145/1552272.1552278.
URL http://doi.acm.org/10.1145/1552272.1552278

[17] Cisco, Cisco data center spine-and-leaf architecture: Design overview, Tech. rep. (2016).

[18] Extreme Networks, Extreme networks: Big data a solutions guide, Tech. rep. (2014).

[19] Cisco, Cisco's massively scalable data center: Network fabric for warehouse scale computer, Tech. rep.

[20] Hortonworks, Cluster planning guide, http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.5.3/bk_cluster-planning/content/hardware-for-slave.1.html, accessed: 2017-01-26.

[21] Y. Liu, M. Li, N. K. Alham, S. Hammoud, HSim: A MapReduce simulator in enabling cloud computing, Future Generation Computer Systems 29 (1) (2013) 300 – 308, including Special section: AIRCC-NetCoM 2009 and Special section:

Clouds and Service-Oriented Architectures. doi:http://dx.doi.org/10.1016/j.future.2011.05.007.
URL //www.sciencedirect.com/science/article/pii/S0167739X11000884

[22] Technical introduction to bufferbloat, http://www.bufferbloat.net/projects/bloat/wiki/Bufferbloat, accessed: 2017-01-26.

[23] S. N. Ismail, H. A. Pirzada, I. A. Qazi, On the effectiveness of CoDel in data centers, Tech. rep.

[24] References on RED (Random Early Detection) Queue Management, http://www.icir.org/floyd/red.html, accessed: 2017-01-26.

[25] Configuring weighted random early detection, https://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfwred.pdf, accessed: 2017-01-26.

[26] RED Congestion Control, https://www.juniper.net/techpubs/en_US/junos12.2/topics/concept/random-early-detection-congestion-control-overview.html, accessed: 2017-01-26.

[27] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, Y. Zhang, Tuning ECN for Data Center Networks, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '12, ACM, New York, NY, USA, 2012, pp. 25–36. doi:10.1145/2413176.2413181.
URL http://doi.acm.org/10.1145/2413176.2413181

[28] Cisco, Cisco Nexus 9000 Series NX-OS Quality of Service Configuration Guide, Release 6.x.

Renan Fischer e Silva received both B.Sc. and M.Sc. degrees in Computer Science from the Universidade Federal do Paraná (UFPR), Brazil, in 2007 and 2010, respectively. He obtained his Ph.D. degree in Computer Architecture from the Universitat Politècnica de Catalunya (UPC) in 2018. Before his PhD he was with industry for seven years. He is currently a fellow researcher at the Barcelona Supercomputing Center (BSC). His research interests include data center infrastructure, local area networks, and big data workloads.

Paul Carpenter is a senior researcher at the Barcelona Supercomputing Center (BSC). He graduated from the University of Cambridge in 1997, and he received his Ph.D. in Computer Architecture from the Universitat Politècnica de Catalunya (UPC) in 2011. Prior to starting his Ph.D., he was Senior Software Engineer at ARM in Cambridge, UK. His research interests include system architecture, energy proportional interconnects, and programming models.