High Throughput and Low Latency on Hadoop Clusters using Explicit Congestion Notification: The Untold Truth

Renan Fischer e Silva, Paul M. Carpenter
Barcelona Supercomputing Center—*Centro Nacional de Supercomputación* (BSC–CNS)
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
Email: {renan.fischeresilva,paul.carpenter}@bsc.es

***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 performance and low latency. This is especially true for mixed use data centres that host both latency-sensitive applications and high-throughput workloads 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. We explain how this behavior decreases throughput, and propose a small change to the way that non-ECT-capable packets are handled in the network switches. 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-ECN and DCTCP can achieve the best performance using a simple marking scheme, in constrast to the current preference for relying on AQMs to mark packets.

***Keywords*-Hadoop, ECN, DCTCP, Throughput, Latency**

I. Introduction

Numerous Hadoop 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 [1]. As recently pointed out, 46% of IoT applications have low latency requirements on seconds, or even on milliseconds [2]. Also, 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 [3].

Not so long ago, a switch offering 1 MB of buffer density per port would be considered a deep buffer switch [4]. New products are arising and with them, a buffer density per port $10\times$ bigger [5]. All this can make the Bufferbloat problem [6] 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 infrastructure [7], which is constantly being pointed as the bottleneck to develop new type of solutions [8], [9]. In parallel with the increase in the capability of network switches, Hadoop also 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 [3], [10], [11]. For that, the network latency on current Hadoop clusters has to be decreased.

This work presents experimental results that show it is possible to reduce network latency on Hadoop clusters without degrading cluster throughput and performance. We expect to make it easy to understand the problem and wish to open new discussions and promote research towards new solutions.

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.

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 compares our approach with related work. Finally, Section 6 concludes the paper.

II. The Problem and Motivation

Network transport protocols, such as TCP, traditionally signal congestion to the sender by dropping packets. This mechanism is simple, but it reduces throughput due to potential time-outs and the need to re-transmit packets. Recent extensions, such as Explicit Congestion Notification (ECN) and Data Center TCP (DCTCP) avoid these overheads by indicating imminent congestion using marked packets. Such congestion control based on proactive signaling was conceived with the premise that it was better to identify congestion before dropping packets and waiting for the sender to react [12]. And the idea was not wrong!

When DCTCP was originally proposed, it was evaluated using a simple marking scheme. Although the marking scheme was, we believe, one of the key points of DCTCP, it was considered to be a straightforward aspect of DCTCP, and it was not debated enough. The authors claimed that the simple marking scheme could be easily mimicked on existing network switches that supported Random Early Discard (RED) [13]. RED is an Active Queue Management (AQM) typically implemented by switch manufacturers. They recommended setting the RED minimum and maximum intervals both to the same value of 65 packets, which they found to be necessary and sufficient to reach the full throughput of a 10 Gbps link.

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

A. A deeper look at TCP packet marking

The main role of the network switch buffers is to absorb burstiness in packet arrivals, which is often found in data center networks. A recent study from Cisco showed how deep (large) buffers help the switches to better absorb such burstiness. For Big Data applications such as Hadoop, Cisco investigated how the network affects job completion time, and found that the second most important characteristic, after network availability and resiliency, was the network's ability to handle bursts in traffic [14].

TCP connections will greedily use the available buffering on their network path. Therefore persistently full deep buffers can cause a problem known as Bufferbloat [6]. 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, which will access a replicated filesystem derived as a production from the batch workload.

After careful investigation considering snapshots from the egress port of network equipment, specifically on the queue level, we finally understood why previous work failed to achieve high throughput and low latency for Hadoop. Figure 1 illustrates the problem which is typical in Hadoop clusters. Limiting buffer utilization while explicitly avoiding early drops of ECT-capable packets that will persistently fill up the queues will allow low space to remain for other type of packets that may arrive in bursts. On Hadoop, limiting the buffer utilization will cause a disproportionate number of ACK packets to be dropped, even ACKs that contain ECE bits, which are useful to indicate congestion. The worst problem happens when a full TCP sliding window is dropped.

ACK packets are short (typically 150 bytes) but RED is typically implemented with thresholds being defined per-packet rather than per-byte. On the other hand, a true marking scheme would mark packets but never drop packets unless its

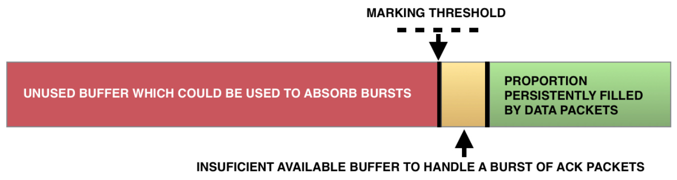

MARKING THRESHOLD
UNUSED BUFFER WHICH COULD BE USED TO ABSORB BURSTS
PROPORTION PERSISTENTLY FILLED BY DATA PACKETS
INSUFICIENT AVAILABLE BUFFER TO HANDLE A BURST OF ACK PACKETS

Fig. 1. Typical snapshot of a network switch queue in a Hadoop cluster

buffer was full. That is what we have found to unleash not only the potential of DCTCP on Hadoop clusters as we also verified that, especially for commodity switches, a classical TCP extended with ECN can outperform DCTCP.

By using a true simple marking scheme instead of trying to mimic one using an AQM, senders are able to reduce their send rate proactively while keeping the typical sawtooth behavior of TCP on a small scale. The throughput of the network is maximised because there is much lower overhead of retransmitting packets.

On Hadoop, whose shuffle phase involves many-to-many communication, employing either TCP-ECN or DCTCP will degrade the cluster throughput when relying on misconfigured AQM to mark ECT-capable packets. This problem happens because on Hadoop a large part of the cluster, if not the whole cluster, will be engaged during the Map/Reduce communication phase known as shuffle, where data is moving across all the nodes. Therefore, data packets and ACKs will typically share the same bottlenecks, and at the minimal pressure on the buffers, packets that are not ECT-capable will be dropped. This effect can be devastating for TCP as not only new connections will be prevented from being established [15] but also ACKs will be constantly dropped. ACKs have an important role to ensure proper signalling of congestion. Congestion should be signalized soon enough, before packets are dropped, to avoid timeouts and retransmission, and ECN uses the ACK packets to echo congestion experienced back to the sender. Also, ACKs are used to control the TCP sliding window, which controls how many packets can be in flight so the receiver can absorb and process them. If a whole TCP sliding window is lost, it will also cause TCP to trigger RTO and its congestion window will be reduced to a single packet, affecting throughput.

We demonstrate in Section IV that if signalized correctly, congestion, which is the steady state of the network during the shuffle phase of Hadoop, can be dramatically reduced. Meanwhile, the performance of TCP can be improved, specially for commodity switches as long as any important packet which is not ECT-capable is allowed to be kept on the buffer that remains available when using tight marking thresholds.

B. Proposed and evaluated solutions

Regarding the problem described previously, we propose two distinct solutions. Our first proposal consists in modifying the AQM implementation to allow an operational mode which, if ECN is enabled, protects the packets that contain ECE-bit on their TCP header, as seen on Table I. As seen in Table II, current AQM implementations only check for ECT(0) or

TABLE I
ECN CODEPOINTS ON TCP HEADER

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

TABLE II
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

ECT(1) bits on the packets IP header, when deciding between marking or early dropping the packet. If a ECT(0) or ECT(1) bit is found, CE-bit is marked so a replied ACK can echo the congestion experienced back to the sender with the ECE-bit set on their TCP header. Protecting packets which have the ECE-bit set means a partial proportion of ACKs will be prevented from an early drop, which are those ACKs marked with ECE-bits to echo a congestion experienced signal back to the TCP sender. It will also protect SYN and SYN-ACK packets, which are necessary to initialize a TCP connection. When ECN is configured, SYN packets have their ECE-bit marked on its TCP header to signalize a 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, ECT-capable packets and also SYN, SYN-ACK and the ACKs which have ECE-bit set won't be early dropped. As we demonstrate with our results this approach is the one which achieves lowest latency while also alleviates the performance loss on throughput.

Our second proposal is to finally implement a true simple marking scheme 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 a slightly higher than our first proposal, cluster throughput is maximized even on commodity switches which offer shallow buffer density per port. Next section describes the experimental environment to evaluate our proposals.

III. Methodology

This describes the experimental methodology for our work. We replicated the methodology used in recent work [10], using the NS–2 packet-level network simulator [16], so we are able to demonstrate the robustness of our findings. Therefore, the NS–2 simulator has been extend with DCTCP [17] implementation and is driven by the MRPerf MapReduce simulator [18].

We also modified RED queue to simulate, in addition to their normal behavior, the two operational modes described on the previous section. First, we protected all the packets that contain ECE-bit in their TCP header. Finally, we repeated the same set of experiments expanding the RED queue to correctly mimic a true simple marking scheme. We could identify the problem related the 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 which have ECE-bits on the TCP header and all the remaining ACK packets. 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:

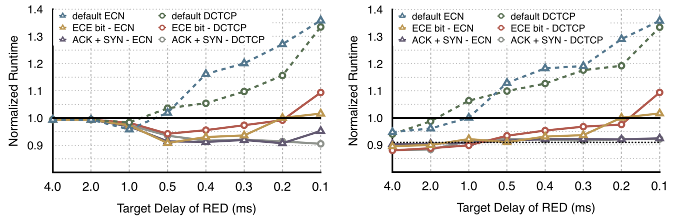
- **Default** behavior which protects only ECT-capable packets.
- **ECE-bit** which protects ECT-capable packets and packets which have ECE-bit set on their TCP header (SYN, SYN-ACK and a proportion of ACKs).
- **ACK + SYN** which protects ECT-capable, SYN, SYN-ACKs, and finally all ACK packets, irrespective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

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

IV. Results

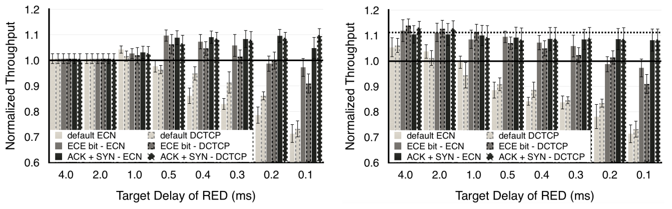
All results are normalized relative to an ordinary DropTail queue. In the case of runtime and throughput, results are always normalized with respect to DropTail with shallow buffers. For these results, the dashed line on the deep buffer plots indicates the (better) runtime or throughput obtained using 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.

We start by presenting the effect of configuring the target delay of RED and how its different thresholds affect Hadoop



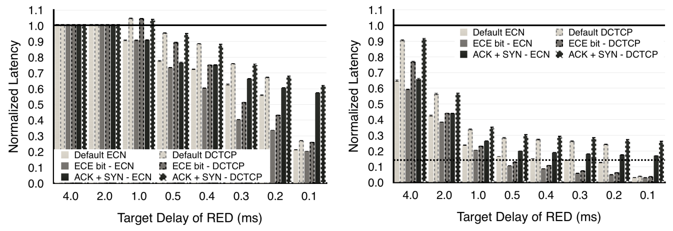
(a) Shallow Buffers (b) Deep Buffers

Fig. 2. Hadoop Runtime - RED



(a) Shallow Buffers (b) Deep Buffers

Fig. 3. Cluster Throughput - RED



(a) Shallow Buffers (b) Deep Buffers

Fig. 4. Network Latency - RED

runtime for switches with shallow buffers. Figure 2a 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 3a we 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 signalled soon enough. The best results and robustness of throughput is also translated to a network latency never lower than 50% to the baseline, as confirmed on Figure 4a.

For deep buffers, we start with Figure 3b. 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 4b, 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 2b shows Hadoop runtime reaching a robust 10% speed-up, which is about the same performance reached by the DropTail queue from deep buffer switches.

V. Related Work

The original DCTCP paper [12] suggested that a simple marking scheme 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 [19] 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 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 [15] pointed to the same problem that happens on a saturated egress queue when trying to open new connections.

Targeting Hadoop clusters, recent studies used ECN and DCTCP in an attempt to improve network latency without degrading throughput or performance [3], [10]. In the latter study, the authors were able to provide useful configurations, but fine-tuning the AQM queues was considered to be non-trivial. The next section concludes this paper.

VI. 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 comparable results with 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 a true simple marking scheme 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 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 of about 85%. The results presented in this paper are not exclusive but can also be expected to be reproduced on other type of workloads that present the characteristics described in our problem characterization.

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.

VII. Acknowledgment

The research leading to these results has received funding from the European Unions 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] G. Mone, "Beyond hadoop," *Commun. ACM*, vol. 56, no. 1, pp. 22–24, Jan. 2013. [Online]. Available: http://doi.acm.org.recursos.biblioteca.upc.edu/10.1145/2398356.2398364

[2] "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.

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

[4] A. Bechtolsheim, L. Dale, H. Holbrook, and A. Li, "Why Big Data Needs Big Buffer Switches. Arista White Paper," Tech. Rep., 2011.

[5] Cisco, "Network switch impact on big data hadoop-cluster data processing: Comparing the hadoop-cluster performance with switches of differing characteristics," Tech. Rep., 2016.

[6] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, no. 11, pp. 40:40–40:54, Nov. 2011. [Online]. Available: http://doi.acm.org/10.1145/2063166.2071893

[7] R. F. e Silva and P. M. Carpenter, "Exploring interconnect energy savings under East-West traffic pattern of MapReduce clusters," in *40th Annual IEEE Conference on Local Computer Networks (LCN 2015)*, Clearwater Beach, USA, Oct. 2015, pp. 10–18.

[8] Y. Chen, A. Ganapathi, R. Griffith, and 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, July 2011, pp. 390–399.

[9] R. F. e Silva and P. M. Carpenter, "Energy efficient ethernet on mapreduce clusters: Packet coalescing to improve 10gbe links," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–12, 2017.

[10] R. F. E. Silva and 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)*, Nov 2016, pp. 415–423.

[11] R. F. e Silva and P. M. Carpenter, "Interconnect energy savings and lower latency networks in hadoop clusters: The missing link," in *Accepted to 2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, Oct 2017.

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

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug 1993.

[14] Cisco Systems, Inc, "Big Data in the Enterprise - Network Design Considerations White Paper," Tech. Rep., 2011.

[15] 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)*. Oakland, CA: USENIX Association, 2015, pp. 145–157. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/judd

[16] "Network Simulator NS-2," http://www.isi.edu/nsnam/ns, accessed: 2017-01-26.

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

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

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