E-EON: Energy-Efficient and Optimized Networks for Hadoop

RENAN FISCHER E SILVA

Barcelona, Spring 2018

E-EON: Energy-Efficient and Optimized Networks for Hadoop

by

RENAN FISCHER E SILVA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

A thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Architecture

Departament d' Arquitectura de Computadors (DAC)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain

PhD Supervisor
Dr. Paul M. Carpenter

Tutor
Dra. Rosa M. Badía

Spring 2018

This page has been intentionally left blank.

Have the courage to follow your heart and intuition.
They somehow already know what you truly want to become...
Steve Jobs

Abstract

E-EON: Energy-Efficient and Optimized Networks for Hadoop

by Renan Fischer e Silva

Energy efficiency and performance improvements have been two of the major concerns of current Data Centers. With the advent of Big Data, more information is generated year after year, and even the most aggressive predictions of the largest network equipment manufacturer have been surpassed due to the non-stop growing network traffic generated by current Big Data frameworks.

As, currently, one of the most famous and discussed frameworks designed to store, retrieve and process the information that is being consistently generated by users and machines, Hadoop has gained a lot of attention from the industry in recent years and presently its name describes a whole ecosystem designed to tackle the most varied requirements of today's cloud applications. This thesis relates to Hadoop clusters, mainly focused on their interconnects, which is commonly considered to be the bottleneck of such ecosystem. We conducted research focusing on energy efficiency and also on performance optimizations as improvements on cluster throughput and network latency. Regarding the energy consumption, a significant proportion of a data center's energy consumption is caused by the network, which stands for 12% of the total system power at full load. With the non-stop growing network traffic, it is desired by industry and academic community that network energy consumption should be proportional to its utilization. Considering cluster performance, although Hadoop is a network throughput-sensitive workload with less stringent requirements for network latency, there is an increasing interest in running batch and interactive workloads concurrently on the same cluster. Doing so maximizes system utilization, to obtain the greatest benefits from the capital and operational expenditures. For this to happen, cluster throughput should not be impacted when network latency is minimized.

The two biggest challenges faced during the development of this thesis were related to achieving near proportional energy consumption for the interconnects and also improving the network latency found on Hadoop clusters, while having virtually no loss on cluster throughput. Such challenges led to comparable sized opportunity: proposing new techniques that must solve such problems from the current generation of Hadoop clusters.

We named E-EON the set of techniques presented in this work, which stands for Energy Efficient and Optimized Networks for Hadoop. E-EON can be used to reduce the network energy consumption and yet, to reduce network latency while cluster throughput is

improved at the same time. Furthermore, such techniques are not exclusive to Hadoop and they are also expected to have similar benefits if applied to any other Big Data framework infrastructure that fits the problem characterization we presented throughout this thesis.

With E-EON we were able to reduce the energy consumption by up to 80% compared to the state-of-the art technique. We were also able to reduce network latency by up to 85% and in some cases, even improve cluster throughput by 10%. Although these were the two major accomplishment from this thesis, we also present minor benefits which translate to easier configuration compared to the stat-of-the-art techniques. Finally, we enrich the discussions found in this thesis with recommendations targeting network administrators and network equipment manufacturers.

Acknowledgements

I start by thanking God for all the endeavors listed below in this Acknowledgment. I also thank Him for the health and capacity to start pursuing the Ph.D. degree and specially, for being able to successfully reach the end of this writing.

I want to express my deepest gratitude to my mentor, my boss and also my thesis advisor Paul, for having selected me for this project. Paul allowed me to have an incredible freedom in the way I decided to conduct my research. Yet, he was always there to discuss a new idea and for understanding my unorganized thoughts which would finally lead to presentable results. For his generosity granting his time on tight deadlines. Without him, this thesis wouldn't have half of the quality it has now.

*T*o Barcelona Supercomputing Center, its director Prof. Mateo Valero and also to all its employees which somehow supported the development of this thesis, my most sincere Acknowledgment. Having its financial support allowed me to have an worry-free employment time and to focus on what really matters.

*T*o my friends made in Barcelona. Thanks to all of you, I had a memorable time. It would be very hard to translate in these words the importance you had during these four years, but I guarantee I will always remember all the memories lived, stories, trips and fun time we shared. Thanks to all these experiences, I felt like these four years were a real sabbatical period in my life.

*T*he real support of all this work is my family. I am lucky to be part of a family with beautiful values, which they always taught me and made me the man I am today. Their visits warmed my heart every single time and I thank my mom, dad and twin brother for just being there during these years. It would be much more difficult without felling you all always around.

*T*o the pre-defense committee and specially to the external reviewers of this thesis, which helped me to improve the quality of this manuscript providing me constructive and detailed comments.

*F*inally, my Ph.D. adventure in Barcelona ends with the writing of this Acknowledgment. I am very grateful for all the personal, professional and academic growth and for all the experiences I was able to live during this season. I consider myself a grateful lucky man...

Author

The research leading to this thesis 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.

Contents

List of Figures

List of Tables

To my loved parents. . .

Chapter 1

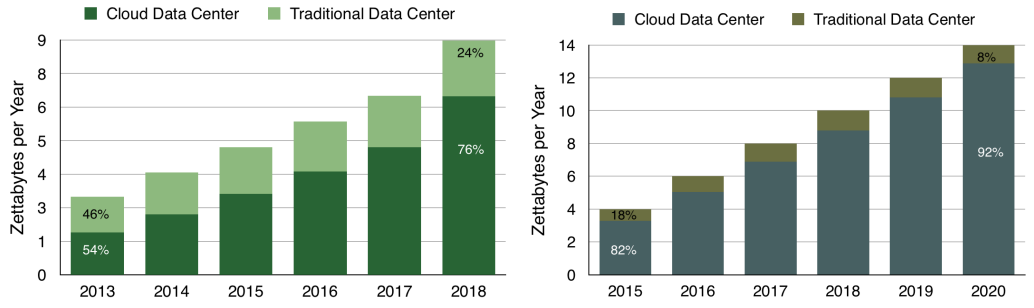
Introduction

In the last two years more data was created than the previous 5,000 years of humanity. It is also believed that in 2017, more data will be created in one year alone [1]. As a consequence, the significance of data is expanding across a wide-range of industries: biotech, energy, IoT, healthcare, automotive, space and deep sea explorations, cybersecurity, social media, telecom, consumer electronics, manufacturing, gaming and entertainment [1]. Such an enormous amount of generated data brings up a challenge on how to design effective infrastructures that are able to store and retrieve data correctly and within an expected time and yet to process and analyse all this data in a way that business and science can take advantage of it. This phenomenon started about a decade ago and presents a non-stop growing scale on both velocity and size. It is commonly referred as Big Data.

As, currently, one of the most famous and discussed frameworks designed to store, retrieve and process the information that is being consistently generated by users and machines, Hadoop has gained a lot of attention from the industry in recent years and currently its name describes a whole ecosystem designed to tackle the most varied requirements of today's cloud applications. Given its importance, the scope of this thesis relates to Hadoop clusters, mainly focused on their interconnects, which is commonly considered to be the system bottleneck. We conducted research focusing on energy efficiency and also on performance optimizations as improvements on cluster throughput and network latency. As a starting point, looking to the past may help when trying to predict the trends for the future. In the next section we discuss the past and also the future predictions for the network traffic generated on Data Centers which shows how significant cloud computing traffic has become and its distribution on global Data Centers.

1.1 Cloud Computing Traffic

This thesis was started in early 2014, and back then we had access to Cisco's Global Index forecast encompassing the period between 2013 and 2018 [2]. As seen in Figure 1.1a, the trend was clear; Cloud Computing traffic had already became dominating on Data Centers and in comparison to traditional Data Center traffic, it already represented 60% of the total Data Center traffic in 2014, with predictions indicating that it would reach more than 3/4 of the total global traffic by the year of 2018. Cloud Computing traffic is associated with cloud consumer and business applications; e.g. Gmail. On the other hand, traditional Data Center traffic is associated with noncloud consumer and business applications such as the traditional email servers which are considered nowadays obsolete.



(A) Between 2013-2018 (B) Between 2015-2020

FIGURE 1.1: Traditional vs. Cloud Data Center Traffic Distribution

Currently, looking to Cisco's latest prediction [3], Data Center traffic has already grown at a faster rate than previously expected. The same happened with Cloud Computing traffic. As seen in Figure 1.1b, by 2015 Cloud Computing traffic already represented more than 80% of the Data Center traffic, and the predictions point it will have reached more than 90% by 2020.

Another traffic characterization from Cisco relates to the source and destination of Data Center traffics. It can be one of 3 types; Data Center-to-user, Data Center-to-Data Center and finally Within Data Center. Examples from the type of services are:

- Data Center-to-User: Such traffic is generated by services offered to users such as web, email and Video on Demand (VOD).
- Data Center-to-Data Center: Inter Data Center traffic is generated by tasks such as replication, Content Delivery Networks (CDN) and intercloud links.

- Within Data Center: Consists in storage, production and development data. Big Data is a significant driver of its traffic. It is also defined as east-west traffic, which is traffic among servers inside the same data center.

As illustrated in Figure 1.2, the north-south traffic which, on the contrary to the east-west traffic, is network traffic that leaves the Data Center, corresponds to only a small fraction of global generated traffic. As seen in Figure 1.3a, only 1/4 of the global generated traffic actually leaves the Data Center and by the latest report, as seen in Figure 1.3b, the Data Center traffic distribution is likely to remain equal for the next years.

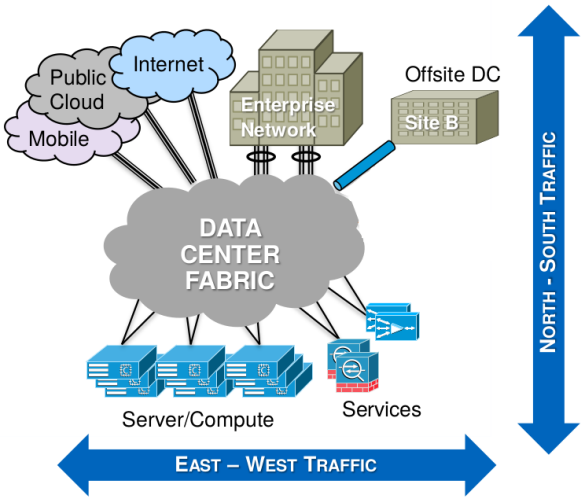
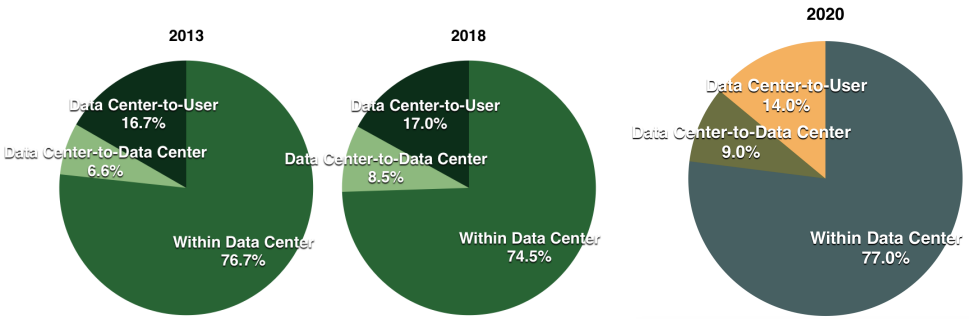


FIGURE 1.2: North-South and East-West traffics on a graphical representation sample (reproduced from [4])



(A) Between 2013-2018 (B) Between 2015-2020

FIGURE 1.3: Global Data Center Traffic Distribution by Destination

It is important to mention that intra-rack traffic is not accounted by Cisco for such traffic distribution. If intra-rack traffic was considered, traffic within Data Center would account for more than 90% of the Global Data Center Traffic distribution [3]. Hadoop,

which is the considered workload on this thesis, schedules its tasks regarding where data resides. If the task is small enough to be executed by servers residing within the same rack, inter-rack communication will not be necessary. Therefore, this workload generates a considerable amount of intra-rack and inter-rack traffic which fits with the current trends and give us the opportunity to make a significant contribution to improve figures related to energy consumption and performance optimization of this ecosystem.

1.1.1 A Brief Story About the Hadoop Ecosystem

As told by Marko Bonaci [5], the story about how Hadoop emerged can be defined as a quest to make the entire Internet searchable. The origin has two important moments. The start of the Lucene project and a paper published by Google which described their solution for processing large data sets on large clusters.

The Lucene project started in 1997 by the Yahoo! employee Doug Cutting when he created a full text search library. By the year of 2001, the project was already moved to Apache Software Foundation, having a thriving Apache Lucene community by the end of the same year. Still in 2001, a subproject of Lucene named Apache Nutch was created by Mike Cafarella, in an effort to index the entire Web.

The problem with Apache Nutch is that it was not scalable, basically because it was running on 4 machines and data interchange between them had to be done manually. Any attempt to increase the number of machines would have resulted in an exponential rise of complexity. An underlying cluster management platform was made extremely necessary so the core problem of indexing the Web could be tackled again.

A couple of years after, more precisely in October 2003, Google published a paper presenting the Google File System [6]. It took some time but Cutting and Cafarella were able to follow Google's blueprints and after what was considered a remarkable job, Nutch Distributed File System (NDFS) was available bringing some key features as hiding operational complexity from users and that it could be built out of inexpensive commodity hardware components.

After having solved the operational problems to have a fully distributed file system, Cutting and Carafella started exploring various data processing models trying to reach the highest level of parallelism. Again, Google came up with the idea to do so and in December 2004 they published the now famous MapReduce: Simplified Data Processing on Large Clusters [7]. The framework was designed with the goal of processing large data sets in a reduced amount of time. Google's MapReduce algorithm solved 3 problems in 2004: parallelization, distribution and fault-tolerance.

By February 2006, Cutting removed the implemented code from GFS and MapReduce out of the Nutch code base and created a new project under the Lucene project which was named Hadoop. It consisted of Hadoop Common core and libraries, the HDFS (finally with the popular Hadoop File System name) and the MapReduce workload. After that, its adoption was a matter of time with companies as Yahoo!, Facebook, Twitter, LinkedIn and many others performing very serious work with Hadoop and contributing back to its open source ecosystem. By 2012, Yahoo!'s Hadoop cluster was already composed with 42000 nodes and the number of Hadoop contributors reached 1200.

If in its early age Hadoop was originally designed for batch-processing, after almost a decade evolving the plataform to fit industry requirements, Hadoop is today capable of online processing which can help reducing the demand from SQL-on-Hadoop gap and also enabling IoT to finally spur Hadoop's case. Deeper details from the framework are covered on the next Chapter where we present the background and related work.

1.2 The Energy Consumption Problem

One of the greatest concerns in the design of data centers is the need to reduce energy consumption. In recent years, the number of data centers has multiplied, and worldwide, they are now responsible for a significant proportion of global electricity consumption [8]. Recently, in 2014, U.S. data centers were responsible for 1.8% of total U.S. electricity consumption. At an average cost of 10 cents per kWh, the annual energy cost of U.S. data centers is about $7 billion per year [9]. Another study even showed that the cost of energy of current data centers had exceeded the cost of the hardware [10].

A significant proportion of a data center's energy consumption is caused by the network. D. Abts et al. [11] recently showed that a typical data center network consumes 12% of the total system power at full load, and even more when the CPU and memory are not fully utilized, which is common in data centers. Another study put the total energy consumption for network switches at 30% [12], divided among top of rack switches (15%), which typically use 1GbE links; and aggregation switches (10%) and core switches (5%), both typically employing 10GbE links. The proportion of energy consumed by the network is likely to increase, as processors and other components continue to improve in energy efficiency and energy proportionality.[1] There is still opportunity to reduce network energy consumption through energy proportionality, since interconnect links, which consume up to 65% of the total network power [13], always consume full power,

[1]The term "energy proportional" means that a component's energy consumption should be proportional to its utilization.

even when the link is idle [14]. Broadcom estimates that it could translate into a reduction of CO_2 emissions by up to 2.85 million metric tons per year only in U.S [15].

The Energy Efficient Ethernet (EEE) standard, approved by IEEE in 2010, improves Ethernet energy proportionality by defining a link sleep mode known as Low Power Idle (LPI). Although the standard defines the low-level mechanisms for entering and leaving LPI mode, its designers chose to promote competition between vendors by not defining how to decide when to enter and leave sleep mode. EEE was initially analysed for Small Office/Home Office (SOHO) environments, but ongoing efforts are analysing its deployment for data center applications, including video streaming [16] and scientific computing [17]. Since EEE can incur significant performance overheads, many system vendors still advise their customers to disable it in production use [18–20], at least until its impact on real applications is better understood.

MapReduce[7] presents a specific traffic pattern, including all-to-all communication in the shuffle phase, between mappers and reducers. As mentioned before, more than 75% of the total traffic nowadays remains inside the data center [2]. Therefore there is still opportunity to reduce network energy consumption through energy proportionality, since interconnect links, which consume up to 65% of the total network power [13], always consume full power, even when the link is idle [14].

1.3 Opportunity for Performance Optimization

With the advent of Big Data, data center applications are processing multi-terabyte datasets, in parallel on large clusters, across hundreds to thousands of nodes. Big data workloads based on Hadoop or similar frameworks generate significant communication among servers within the same data center. In particular, as explained below, the shuffle phase of MapReduce involves an all-to-all communication, which presents a stressful load on the network.

Although Hadoop is a network throughput-sensitive workload with less stringent requirements for network latency, there is an increasing interest in running batch 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 [21]. Also, numerous Hadoop distributions are appearing with the aim of providing low-latency services, which may in future share the same infrastructure as MapReduce workloads on a heterogeneous cluster with controlled latency [22]. As recently

pointed out, 46% of IoT applications have low latency requirements on seconds, or even on milliseconds [23].

Current network switches offer much higher buffer density 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 be considered a deep buffer switch [24]. New products are arising and with them, a buffer density per port 10× bigger [25]. All this can make the Bufferbloat problem [26] even worse, with latency on these networks reaching up to tens of milliseconds for certain classes of workloads.

1.4 Thesis Contributions

This thesis discusses two major problems of Hadoop clusters. Our research focused on energy consumption and performance optimization at the interconnect level (on network switches and server NICs). To do so, we faced several challenges throughout the thesis development which led us to reach a deeper level of knowledge so we became able to analyse, characterize and propose changes to state-of-the-art techniques that if adopted by the industry and network equipment manufacturers, they will push for the improvement of Hadoop's network performance while also reducing its energy footprint.

As it refers to the title of this work, E-EON stands for Energy Efficient and Optimized Networks for Hadoop. We selected E-EON to reference the set of techniques presented on this work which can be used to reduce the network energy consumption and yet, to reduce network latency while cluster throughput is improved at the same time. Also, such techniques are not exclusive to Hadoop and they are also expected to have similar benefits if applied to any other Big Data workload and cluster that fits the problem characterization we presented throughout this thesis. We provide more details from the contributions bellow:

1. We were the first to evaluate Energy Efficient Ethernet on Hadoop clusters. We found that good energy savings can be obtained by leaf interconnects, but at the aggregation level packet coalescing is necessary to reduce the energy footprint to near ideal level. Our study produced two publications; the first was a best paper nominee [27] and the second a journal extension [28].

2. Our second contribution stands for the first analysis of Active Queue Management and TCP protocol extensions that enable the use of ECN such as ECN standalone and DCTCP. We were able to reduce Hadoop's network latency to 85% while

maintaining cluster throughput on 95% from the baseline. Yet, we concluded that it is not a trivial task to configure such queues. This study was awarded one publication [29].

3. The third contribution from this Thesis comes from a deeper study to understand why AQMs are so difficult to configure on Hadoop clusters. We identified the reasons for that and we characterized the traffic pattern for it to happen. Later on we suggested modifications on how switch queues handle ECN marked packets. We were able to reduce network latency while on some cases even improving throughput by 10%. We also solved the configuration problem providing effortless configuration. Any other Big Data workload that follows the same traffic pattern will benefit from this study, which carried on one publication [30] and has another work currently under review and for publication during the year of 2018.

4. Finally, our last contribution resides on identifying the right conditions to combine Energy Efficient Ethernet using Packet Coalescing together with Active Queue Management. From such work is possible to achieve very low energy consumption combined with very low latency networks. Such study yielded a conference paper [31].

Below are publications list related to the contributions of this thesis.

- **[LCN15] Renan Fischer e Silva** and Paul M. Carpenter, *Exploring Interconnect Energy Savings Under East-West Traffic Pattern of MapReduce Clusters.* on the 40th IEEE Conference on Local Computer Networks 2015, best paper nominee [27].
- **[LCN16] Renan Fischer e Silva** and Paul M. Carpenter, *Controlling Network Latency in Mixed Hadoop Clusters: Do We Need Active Queue Management?* on the 41st IEEE Conference on Local Computer Networks 2016 [29].
- **[CLUSTER17] Renan Fischer e Silva** and Paul M. Carpenter, *High Throughput and Low Latency on Hadoop Clusters using Explicit Congestion Notification: The Untold Truth* on the 19th IEEE International Conference on Cluster Computing 2017 [30].
- **[LCN17] Renan Fischer e Silva** and Paul M. Carpenter, *Interconnect Energy Savings and Lower Latency Networks in Hadoop Clusters: The Missing Link* on the 42nd IEEE Conference on Local Computer Networks 2017 [31].
- **[Transactions on Networking] Renan Fischer e Silva** and Paul M. Carpenter, *Energy Efficient Ethernet on MapReduce Clusters: Packet Coalescing To Improve 10GbE Links* on the IEEE/ACM Transactions on Networking, October 2017 [28].
- **[- 2018] Renan Fischer e Silva** and Paul M. Carpenter, *TCP Proactive Congestion Control Revamped: the Marking Threshold*, under review.

1.5 Thesis Organization

The rest of this thesis is organized as follows. The next chapter presents background and literature, which discusses the necessary background to follow this thesis work. Chapter 3 presents the general methodology used during our experimentations. Chapters 4, 5, 6 and 7 discuss the four contributions of this thesis as presented above. Finally Chapter 8 concludes this thesis.

Chapter 2

Background and Related Work

In this chapter we cover the background studied during the development of this work. It describes the most important problems encountered in modern data center networks and this chapter also describes the main solutions, both current practices and state-of-the-art. Some of the problems presented here may already have been solved for the current generation of Data Centers but they still need additional study of how they will behave on a particular workload or scenario. This chapter also summarizes the Hadoop framework and MapReduce programming model.

2.1 Energy Efficient Ethernet

IEEE 802.3az Energy Efficient Ethernet (EEE) was approved by IEEE in September 2010 [32]. Since Ethernet is the dominant technology for wire-line LANs, the power saving mechanisms of EEE are expected to bring considerable energy savings [16]. EEE has already been deployed, but many system vendors advise their customers to disable it in production use [18–20], since it has a poorly understood impact on real world application performance, with no visibility of the performance–energy tradeoff.

Cisco published a study of Energy Efficient Ethernet that showed a 16% reduction in system power for synthetic Ethernet traffic [18]. The same study recommends that EEE should be used only for edge devices. Yamaha Audio advises their customers to disable Energy Efficient Ethernet for audio and video streaming [19]. Dell also presents a troubleshooting section related to EEE [20].

The Energy Efficient Ethernet standard defines the low-level mechanisms for entering and leaving sleep mode, known as Low Power Idle (LPI). Figure 2.1 shows the timeline of a link, which is initially active. Transitioning into LPI mode requires time T_s. While the

link remains in LPI mode, the transmitter sends periodic refresh signals, each of duration T_r, to allow the receiver to continue to adapt to channel characteristics and to recognise if the link is physically disconnected. Before transmitting a frame, the link must first be woken from LPI mode, and doing so requires time T_w, which is approximately 4 µs for 10GbE and 16 µs for 1GbE, similar to the time to transmit a small number of 1,500-byte Ethernet frames. Power consumption is at full when the link is active and during wake and sleep transitions, but in LPI mode, the average power consumption, including refresh, is reduced to about 10%. The EEE standard does not define the strategy for deciding when to enter and leave low-power mode. This subject is an active area of research.

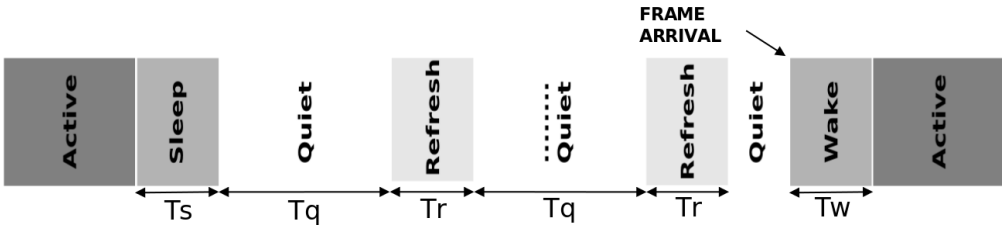


FIGURE 2.1: Timeline of a link using Energy Efficient Ethernet

TABLE 2.1: EEE single-frame efficiency

Speed	Min. T_w (µs)	Min. T_s (µs)	1,500-byte frame T_{frame} (µs)	1,500-byte frame Efficiency (%)	150-byte frame T_{frame} (µs)	150-byte frame Efficiency (%)
100Base-TX	30.5	200	120	34.2	12	4.9
1000Base-T	16.5	182	12	5.7	1.2	0.6
10GBase-T	4.48	2.88	1.2	14.0	0.12	1.6

The energy efficiency of EEE is therefore impacted by a) the idle power consumption (about 10%) and b) the wake/sleep overheads. The idle power consumption is a fixed cost, unaffected by the strategy for entering and leaving low-power mode, which affects only the wake/sleep overheads. When the load is moderate to high, the energy overhead of transitioning in and out of LPI mode can be amortised over a number of packets. When the load is moderate to low, however, it may be necessary to wake and sleep the link to transmit a single frame, incurring high energy and latency penalties in relative terms [33]. The extreme case is illustrated in Table 2.1, which summarises the energy efficiency, assuming that the link wakes and sleeps to transmit a single frame. The numbers are achieved by dividing the time spent to transmit a single frame by the total time to leave LPI mode and return to this state.

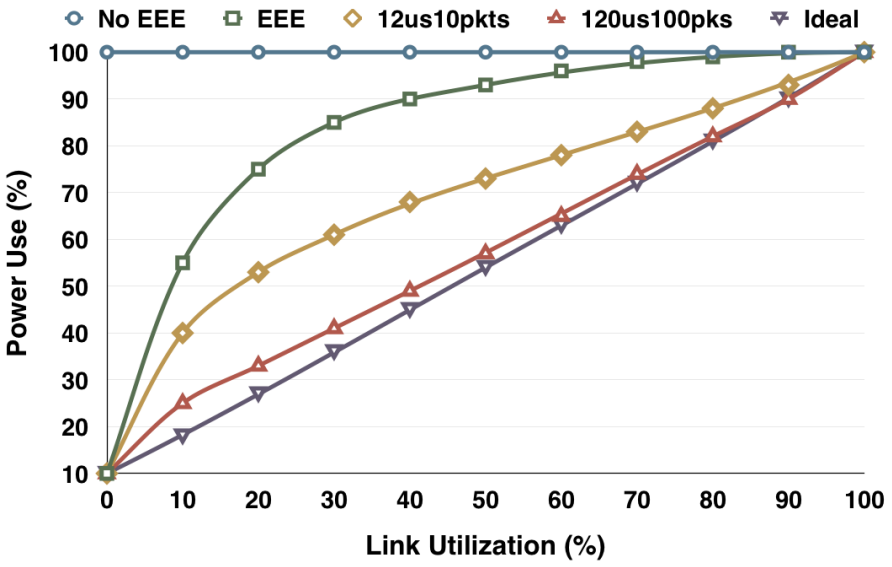


FIGURE 2.2: Normalized link power consumption as a function of utilization, assuming Poisson arrivals (redrawn from [14])

To understand the context, Figure 2.2 (reproduced from Christensen et al. [14]) indicates the efficiency of EEE, for Poisson arrivals. The ideal power consumption (assuming that the idle power consumption cannot be avoided) is given by the line from 0% link utilisation (for which the power consumption is 10%) to 100% utilisation and power. The figure also shows how the baseline, which is legacy Ethernet, is always at full power irrespective of link utilization. Standard EEE works well at very low link utilisation because, although the relative efficiency of each packet is poor, the low packet frequency means that the total power overheads are also low. It is in fact at moderate loads that Standard EEE has poor energy efficiency. At 10% link utilization it already uses almost half of the baseline power of legacy Ethernet and at 20% link utilization it reaches 70% of the baseline power.

Previous studies show that the energy savings depend on the traffic pattern (which often deviate significantly from Poisson) and network load [14, 33]. Proposals include Power Down Threshold [17], or stall timer, which initiates sleep after a defined period of inactivity, typically about 50 µs. Another technique, packet coalescing (also known as packet aggregation), intentionally delays any packet that arrives while the link is in LPI mode. If additional packets arrive within a short time, then the link can be woken once to transmit them back-to-back, amortising the wake and sleep energy over multiple packets [14, 33].

Packet coalescing introduces a significant and variable latency, and it is not clear which workloads can tolerate the extra latency and burstiness. It is usually characterised using two parameters: the *trigger*, which is the maximum number of packets to hold

(or alternatively, the buffer size in KB) and the *timer*, or holding time, which is the maximum time to hold a packet. The right configuration is critical for maximum energy savings and low performance overhead [34]. Christensen et al. [14] suggest using either a timer of 12 µs and a trigger of 10 packets or 120 µs and 100 packets. Their results, also included in Figure 2.2, as before, assume that the traffic is characterised as Poisson arrivals. For Poisson network traffic, packet coalescing was able to reach a power use much closer to ideal, specially at the latter setting.

Although Figure 2.2 illustrates well the problem and the solution for it, the best setting always depends on the distribution of the traffic on the network. For example, another publication uses substantially different values [33], of 1 ms and 10 ms as timers, in both cases with 1,000 packets as trigger. We show the effect for MapReduce workloads in Chapter 4, where we modify the parameters, including configuring different devices to use different settings. Even if the application is not expected to be latency sensitive, larger holding times and larger numbers of packets lead to greater burstiness, which we found to cause Ethernet packet loss. This is especially problematic for commodity data centers, whose switches have relatively small buffers. Spending more money on high-end switches could reduce or eliminate this problem, but it is unlikely to lead to a low cost or low energy solution. Next section resumes the MapReduce programming model.

2.2 MapReduce and Hadoop

In 2004 Google introduced the MapReduce programming model for reliable fault-tolerant processing of huge data sets on large commodity clusters [7]. MapReduce is a pipelined data processing framework. The pipeline is broken down in three different phases: map, shuffle and reduce. Each node of the cluster runs a Daemon, which also defines the type of service run by each server in the cluster.

JobTracker is an essential Daemon which runs on the *NameNode*. The NameNode is responsible for the Hadoop housekeeping.

TaskTracker is another Daemon from Hadoop framework which runs on every *DataNode*. DataNodes will run Map and Reduce tasks, which are executed by the TaskTracker of the nodes. TaskTrackers will be in constant communication with the JobTracker signalling the progress of the task in execution.

MapTasks will be issued by the JobTracker at the initial phase of the pipeline.

Map phase: Initially, the JobTracker is responsible for dividing the dataset from a given job into many data splits. Such splits are independent chunks, which are processed in

parallel by map tasks assigned to TaskTrackers. Each TaskTracker can lunch several MapTasks, each one related to one split of data. The Mappers process the original input and convert it into intermediate results organized in <*key,value*> pairs and stored into a temporary output file, which is also split into several data partitions where each data partition will be intended for being processed by a ReduceTask .

The JobTracker also issues ReduceTasks to be executed by the TaskTrackers. ReduceTasks play their role during the second and third phases of the MapReduce pipeline.

Shuffle phase: Each ReduceTask fetches their corresponding data partitions from the output files of every MapTask. It leads to the so-called shuffle stage, which involves all-to-all data transfers among nodes, with the merged data finally passed to the ReduceTasks.

Reduce phase: Each ReduceTask finally processes the merged segments. The final result is then stored into the distributed file system known as HDFS (Hadoop Distributed File System).

One important reason that the MapReduce framework has became so popular is the possibility for programmers to rely on the framework to be responsible for many complex implementation details while writing their MapReduce applications. The programmer is given a data abstraction in terms of *map* and *reduce* operations on key/value pairs, and the framework takes care of the implementation details including automatic parallelization, task scheduling with data locality, monitoring, redundant distributed data storage, and re-executing failed tasks. The input and output data for the MapReduce jobs are stored on the distributed filesystem which uses disks attached to the same nodes used for computation. Also, the JobTracker tries to schedule tasks to run on nodes where data is already present, resulting in high data locality [35]. Several open-source MapReduce frameworks have been developed over the years, with by far the most popular one being Apache Hadoop [35]. The next section presents the challenges found on Data Center networks.

2.3 TCP in Modern Data Centers

The Transmission Control Protocol (TCP) is one of the main protocols from the Internet Protocol (IP) suite and provides reliable stream–oriented connections, which complements the IP protocol and is also referred as TCP/IP [36]. TCP is a mature protocol that has been extensively studied over a number of years and in a wide range of networks as LANs, WANs, data centers, campus networks and enterprise networks [37].

TCP was initially designed for Wide Area Networks (WANs) [38], and certain aspects of its design, such as the minimum Retransmission Timeout (RTO) of 200 ms are better suited to WANs than data center LANs.

Recent studies show that 97% of the traffic in current data centers is carried by IP packets, either as TCP or UDP segments depending on the workload [39]. In 2010, Microsoft Research published a study of 150 TB of network traces that showed that, for their data center, TCP segments made up more than 99% of their internal traffic [40].

Problems that arise in such a low-latency environment include (a) *TCP Incast* [38], a dramatic loss in throughput for many-to-one communication patterns, where congestion leads to packet loss, (b) *TCP Outcast* [41], where (surprisingly) the throughput to a congested node may be much lower from nearby nodes than from more distance ones, and (c) *Bufferbloat* [26], where congestion causes excessive packet buffering, leading to high and variable latency.

The goal of any transport protocol as TCP is to maximize the usage of the network. TCP, or any other congestion protocol, will be probing the network, trying to find how many packets the network can carry until it loses the packet and then back-off. On other words, TCP is always pushing the network into congestion and then backing-off. Using deeper buffer equipments will automatically increase the average delay per packet in order to obtain some gain on throughput and bursty tolerance. It is a trade-off between adding extra latency to the network while having more tolerance to bursty communication and also obtaining a higher throughput.

There are two situations where network equipment most benefit from larger buffers. Firstly, in upper-layer devices, at the aggregation and core layers, when bursty traffic on multiple incoming links is redirected to the same outgoing port, the switch will have to queue the packets before transmitting them. Secondly, in the access layer; i.e. in the Top-of-Rack (ToR) switches, incoming traffic going to the server nodes may arrive on a link that has higher bandwidth than the link to the server; e.g. packets arrive at 10GbE but must be transmitted at 1GbE.

In an ideal case, data center networks should accommodate long flows, which require high throughput, and also allow short flows to have low latency in scenarios where buffers are heavily used. Doing so may not be possible on some workloads, and trade-offs have to be considered to adjust each case to the best possibility.

2.4 Controlling latency and buffer occupancy

This sections presents the mechanisms studied here to control network latency and buffer occupancy.

2.4.1 Active Queue Management

AQM schemes have been proposed to manage buffer occupancy to keep the average latency of the buffers below a determined threshold. The goal is eliminate the problem found on DropTail queues that tend to penalize bursty flows and also introduce high latency into the network. Bursty flows are penalized when TCP global synchronization happens. Instead, an AQM scheme aims to keep the delay controlled by providing feedback to the end points through appropriately dropping packets. Another goal of these smarter queues is to support the use of Explicit Congestion Notifications, found on TCP network protocol, that allow end points to react before congestion happens. On this thesis we selected two AQMs to compare, Random Early Detection (RED) [42] and Controlled Delay (CoDel) [43].

Random Early Detection (RED) was proposed in 1993 [42] and since then, it has been widely studied and adopted. Implementations of RED are found on Linux, Solaris, and FreeBSD [44]. It is also implemented by network equipment vendors including Cisco [45] and Juniper Networks [46]. RED uses configurable thresholds to decide when to mark packets if combined with ECN, and drops packets based on a probability that grows with the queue occupancy.

Controlled Delay (CoDel) was proposed in 2012 and since then, it has gained more attention. Its usage is recommended by the Bufferbloat initiative [26] [47]. It promises to be easier to configure than RED, which has several parameters and variants to be configured. CoDel claims it has no parameters to set at all, but still, the user needs to configure the target *delay*, which is the tolerable delay per-packet when queued until it is transmitted, and the *interval* how often the delay per packet of transmitted packets within the interval is evaluated. If any packet has a delay grater that the target, the interval is shortened, otherwise it is reset at the end of its cycle.

2.4.2 Explicit Congestion Notifications

ECN are helpful to indicate a pre-state of congestion on the network and allow senders to proactively react before it happens. Instead of waiting for the buffer to drop packets and trigger the fast recovery state of TCP, which can lead to a RTO timeout and causes

TCP to reset to its slow start state, the sender reduces its congestion window by the number of marked packets, alleviating the pressure under the buffer that signalized the congestion, which helps to reduce latency and specially jitter. Therefore, proactive congestion control is helpful for two reasons. First, the default value of 200 ms is not suitable for low–latency networks. Finally, even if the RTO is configured to a much smaller value on these networks (typically 1 ms) RTO timeouts lead to the slow start phase which can temporally decrease throughput.

Yet, on data center networks, using proactive congestion control by enabling ECN was found to reduce throughput of applications while keeping the latency and buffer occupancy low, which may not be desired on frameworks with high throughput requirements. As an alternate solution, DCTCP has been proposed which involves some modifications on the TCP network protocol to specifically fit data center network requirements: high throughput and small latency.

2.4.3 Data Center TCP

DCTCP is an extension to the TCP network protocol proposed by Microsoft Research Center as an alternate solution to specifically reduce the latency on data center networks without affecting throughput. In comparison with ECN, DCTCP extends the ECN processing to estimate the fraction of bytes that encounter congestion, rather than simply detecting that some congestion has occurred. In short, while ECN reduces the congestion window by the number of ACK packets echoed to the source, DCTCP modifies how the source handles the received ACKs to estimate a more gentle reduction on the congestion window, which is more suitable for low–latency networks. DCTCP then scales the TCP congestion window based on this estimate. This method achieves high burst tolerance, low latency, and high throughput with shallow-buffered switches [40].

On its evaluation using commodity switches, DCTCP was able to deliver even better performance than ECN itself. Currently, network equipment manufactures are iterating into their lineup and recommending the usage of deep buffer switches for Big Data frameworks, specially Hadoop, which demands more analyses of how DCTCP performs on such type of workloads and using these new network equipments.

The next section presents related work to specific network traffic patterns using Energy Efficient Ethernet followed by the related work to controlling latency mechanisms found on Data Center networks.

2.5 Related Work

2.5.1 EEE Under Specific Network Traffic Patterns

De la Oliva et al. conducted a study of the effect of Energy Efficient Ethernet on a video streaming service using UDP traffic [16]. Their simulation results showed that UDP video streaming could achieve good energy savings without the need for advanced techniques such as packet coalescing. They mention, however, that using TCP rather than UDP would have led to lower energy savings, due to TCP acknowledgements and TCP congestion control mechanisms.

In the field of High-Performance Computing (HPC), Saravanan et al. established that although scientific applications have high peak communication demand and therefore need a high-performance interconnect, the average traffic is usually low [17]. This work led to an adaptive control mechanism for Energy Efficient Ethernet that maximises energy savings subject to a bound on the percentage increase in execution time [48]. Dickov et al. presented an analysis of data compression for InfiniBand network energy savings [49]. They also introduced a novel power reduction software manager for InfiniBand links [50, 51]. Both techniques of Dickov et al. are implemented in the MPI software layer, so they are only applicable to workloads written using MPI.

There are several differences between our approach and the above related work in HPC. Firstly, HPC workloads have complex dependencies and require low latency, leading to the conclusion that packet coalescing would not be useful [17]. Both Dickov and Saravanan use high-level simulation models that abstract away fine-grain details, whereas we use a detailed packet-level simulator. We found that in our context, especially with switches with shallow buffers, packet-level phenomena, such as Ethernet packet loss and TCP/IP congestion avoidance, have a critical effect on both performance and energy. Accurate quantitative results could therefore only be obtained using a packet-level simulator, described on our Methodology on the next chapter.

2.5.2 Controlling latency and Buffer Occupancy on Data Center networks

Heterogeneous clusters are becoming relatively more common on modern data centers as an attempt to reduce cost and avail the built infrastructure. As an example, Apache Myriad is a open source project that enables Apache Hadoop to run side-by-side with other type of applications, dynamically sharing cluster resources [52].

Several vendors are positioning themselves for the Big Data market and have introduced network equipment with the promise of increased performance for Big Data applications. Arista Networks is marketing their new 7048T, 7280E and 7500R switch series with large buffers as recommended solutions for optimum performance for Hadoop [24]. Cisco published a study that found that network latency has little impact on job completion time, among other factors such as availability and resiliency, burst handling queuing, over-subscription ratio and data node network speed [53]. In the same study, burst handling queuing capability was considered as the second most important factor that affects job completion time.

A lot of attention so far has focused on RED, which is widely used as a baseline for the evaluation of new AQMs. Also it has based versions implemented by network vendors as Cisco [45] and Juniper Networks [46].

The CoDel IETF draft [54] suggests that CoDel would be useful in environments other than the normal Internet, including in data center switches, particularly for MapReduce clusters. As described, a CoDel queue tuned for such an environment promises to minimize packet drops, while keeping throughput high and latency low.

DCTCP was presented before CoDel, reason why we believe an evaluation using CoDel was not considered at that moment. Its evaluation used only a variant of the RED queue, with recommended values of the minimum and maximum thresholds both equal to 65 packets. When DCTCP was compared to RED, it was suggested that although RED combined with ECN would dramatically reduce network throughput in data centers, DCTCP would maintain high throughput (while providing low queue occupancy and low delay). The explanation was that, instead of dramatically cutting the TCP congestion window by the number of marked acknowledges, DCTCP reduces the window gently to maintain high throughput.

Wu et al. presented a comprehensive study on the tuning of ECN for data center networks, which they described as ECN* [55]. Their new approach performs as well as DCTCP, but it requires modifications to the ECN marking scheme in the network switches (the transport protocol and end points are not modified). They proposed marking packets when they leave the network queue ("dequeue marking") instead of when they enter the network queue (classical "enqueue marking", as used in RED). Dequeue marking seems to deliver similar results to DCTCP's gentle congestion control. This proposal also came before the introduction of CoDel, which also marks packets when dequeue occurs, suggesting that CoDel would also deliver better performance than RED. Big Data workloads or deep buffer switches were not considered, missing a more profound analysis of the scenarios that typically present bufferbloat phenomena. When specifically compared using NS–2 simulations with synthetically generated traffic with

a Pareto distribution, CoDel delivered lower latency than RED but the latter was still considered as a good candidate for an AQM [56].

Incast was shown to have little significant impact on the performance of Hadoop, assuming a well-tuned Hadoop cluster. Incast is characterized by many-to-one communication where the buffer on the receiver pipe is heavily pressured so packets are lost. But its effect is specially devastating on partition/aggregation workloads which perform small queries, because the default RTO (Retransmission Timeout) penalty of 200 ms represents a big overhead on the overall performance. Hadoop presents many-to-one network communication in its shuffle phase, but on a well-configured system there is not significant overhead caused by incast. Also, buffers of network equipments are highly used, especially during the shuffle and write phases [57]. As we demonstrate in our results, Hadoop is highly throughput-sensitive. Therefore, in contrast to the recommendation to use a smaller minRTO in data centers, specially when using big buffer equipments which tolerate more bursty communication, reducing the RTO from 200 ms to 1 ms can impact on a fake RTO penalty. For example, an in-flight packet could still be queued in a buffer and since on bufferbloat scenarios the average latency per packet can be higher than such small RTO, TCP would trigger its timeout even if the packet is not dropped. For this reason our simulations use the default TCP minRTO of 200 ms and the overall results can be verified on the next chapters. Next chapter covers the methodology used to obtain the results presented on this thesis.

Chapter 3

Methodology

This chapter presents the overall methodology used throughout the development of this thesis to generate the necessary results so we could carry on with our deeper analisis and either confirm previous insights or even obtain new findings based on results which came to be contrary to the naive assumption.

3.1 Simulation Environment and Workloads

We evaluate the impact of Energy Efficient Ethernet as a function of the network topology, workload, and control algorithm, using the NS–2 packet-level network simulator [58]. This simulator has been extended with a model of Energy Efficient Ethernet [59], which has been previously validated [14] and used extensively in previous work [60]. The network simulator is driven by the **MRPerf MapReduce** simulator [61], which allows researchers to carry experiments on the MapReduce framework while using NS–2 to simulate the underlying networks.

MRPerf was presented not so long ago as an option for researchers which desire to use the already consolidated NS–2 to carry research on this well-known and trusted packet level simulator. MRPerf was recently extensively evaluated [62], 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 contributions from this work is not related to the computation phase of MapReduce, but to its shuffle phase, which is when data is moved across the cluster. Therefore the improvements on TCP throughput measured here can be translated to a production environment. The real gain on the MapReduce runtime will depend on how much is the extent of each workload regarding the proportion of computation and communication.

We also evaluated control delay mechanisms as a function of the network topology, hardware configuration, and transport protocol, using the NS-2 packet-level network simulator [58]. The simulator has also been extended with CoDel [63] and DCTCP [64] implementations. Extensive evaluation has been done with RED, CoDel and DCTCP using NS-2 which brings us to a fair comparison with our results. NS–2, MRPerf, CoDel and DCTCP are open source and EEE module can be obtained contacting referenced authors [59], so using the parameters described in this chapter and also in each specific methodology from each of the next chapters, our simulation methodology has the advantage that it can be reproduced and future work can be carried out on it. We could not use real hardware because the EEE control algorithm is implemented in NIC and switch firmware, and no hardware was available for which we were able to change the packet coalescing settings.

3.2 Hardware configuration

An important question is the power consumption of the 1GbE and 10GbE links. 1GbE (1000BASE-T) cards were originally expected to consume about 1 W, but current NICs using 110 nm silicon technology require just 0.5 W [65]. On the other hand, 10GbE (10GBASE-T) NICs are still considered to be power hungry. The previous generation, at 40 nm, consumed about 5 W, while the current generation at 28 nm is expected to consume between 2 W and 4 W [66]. Our main contributions are related to energy savings in the 10GbE links, so we conservatively chose relatively power efficient 10GbE links. In summary, we assume 0.5 W per port for 1GbE and 2.5 W per port for 10GbE. On real hardware, numbers can still vary depending on the vendor's to be considered. We restricted the power consumption to the NICs as considering power consumption for the servers could translate to an even wider range of values and great imprecision since different architectures and solutions could be adopted (going from low end commodity servers to high end more expensive ones). Therefore we reduced the scope of this work to estimate the energy consumption of the links themselves, which can be verified on the next chapters.

We provide results for both commodity and more expensive switches. Hadoop clusters often use inexpensive commodity switches, which have small (shallow) buffers. Small buffers can cause excessive packet loss, leading to the incast and outcast problems described in Chapter 2. These problems can be alleviated using expensive switches with larger (deep) buffers.

The over-subscription ratio on the 10GbE links is never higher than 4:1. Specifically in the following chapter we explored multiples network over-subscription as 1.2:1, 2.5:1 or

4:1. For the rest of the thesis we considered fixed network over-subscription after we understood its impact in Chapter 4. Selecting a network over-subscription not higher than 4:1 matches Cisco's recommendation that MapReduce clusters should be deployed with an over-subscription ratio of 4:1 or lower at the access layer [53].

3.3 Topology

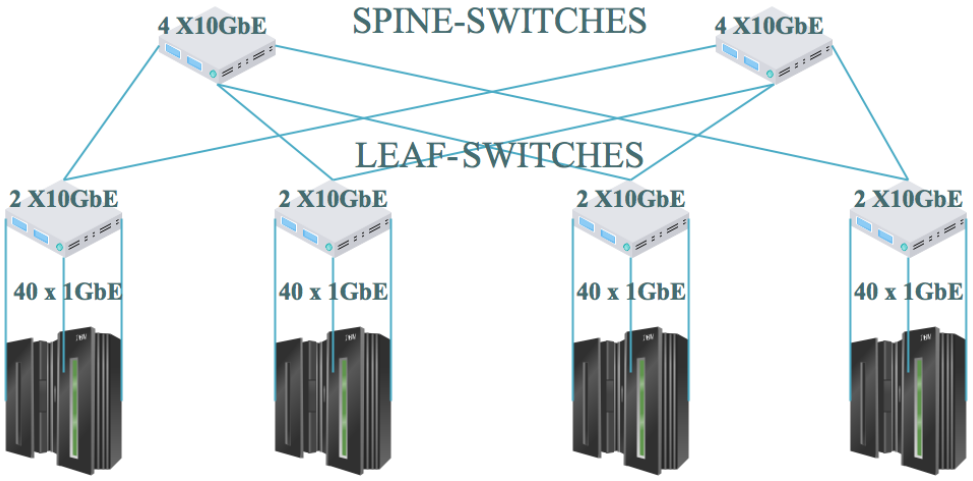


FIGURE 3.1: Leaf-spine Cluster Topology

The data center architecture selected for this work was the leaf-spine architecture [67], as seen in Figure 3.1. Contrary to the traditional hierarchical three–layer network, which is recommended for north-south traffic, the leaf-spine architecture is the recommended architecture for warehouse scale computers [68] when focusing on east-west traffic among servers of the same cluster [69]. The topology brings spine and leafs switches similar to the access and aggregation switches found on the classical k-ary fat tree [70]. The advantage of leaf-spine when compared to the k-ary fat free is its full mesh connectivity between leaf and spine switches, yet it follows indeed a two-tier clos architecture. Each leaf switch, also known as top-of-rack (ToR) switch, is connected to the spine switch, regularly named aggregation switch on the fat tree model, using a single 10GbE link. It can "scale out" to a fairly large numbers of servers by adding more switches, although it also brings a concern to the amount of cables and the cost of the network equipment required, once each leaf switch must be connected to each spine switch [69].

It is also important to mention that the leaf-spine topology can also be categorized as a 2-level fat-tree topology, i.e. without the aggregation layer between the edge and the core layer, and it is not organized in pods [71] [72]. Since the focus of this work was to analyze MapReduce workloads in depth we decided to consider only the leaf-spine topology for

this thesis, which seems to be recommended for Hadoop, as seen in various references for cluster design [24] [73] [68]. Also, the NS-2 simulator has scalability problems due to its packet-level nature and therefore, considering the addition of the super-spine layer on top of the spine layer would require many more nodes to be added so we could organize the cluster on a larger folded clos topology (organized in three tiers). Such extra layer would require the addition of many more nodes to justify the larger topology and therefore we limited the cluster size on our experiments. Nevertheless, from the results obtained on this work we expect similar phenomena in many other distributed applications that are throughput sensitive and use the 2–layer fat-tree topology, which is comparable to the leaf-spine topology, or either use the more traditional 3–layer fat-tree topology, typically employed on large clusters.

On our cluster we used the multiPath option from NS-2 that simulates the equal cost multi-path routing through two equal cost routes. Equal Cost Multi Path (ECMP) feature is essential for a representative analysis of this cluster topology, which offers multiple routes and loaded over-subscription. Recent work [74] shows the benefit of using multipath TCP achieving improved network utilization and better reliability. Since multipath TCP is not yet commonly adopted in the mainstream and we want to specifically investigate the impact of latency control mechanisms and DCTCP on Hadoop we decided to bound the scope of this work using only ECMP feature.

Regarding differences in topology seen in the next chapters, we evaluated different cluster sizes throughout the development of this work. The Energy Efficient Ethernet module available for NS-2 is implemented in a different layer than the duplex-links used with AQMs. During the development of this work we found that the EEE implementation available for NS-2 does not support multiple routes and therefore, it was not possible to conduct experiments with EEE and simulating larger cluster topologies. The simulation of large topologies is a task that is only possible when the ECMP feature is available and this way it translates into the multiple paths being fully utilized. For such reason, in Chapter 4 and in Chapter 7 we simulated a smaller cluster with 2-racks and 1 aggregation switch. We understand it could be considered a limitation so we compensated it by analyzing three different network over-subscription ratios as mentioned before. Therefore, we strongly believe we were able to obtain realistic figures for the results related to Energy Efficient Ethernet. In Chapter 5 and in Chapter 6 we utilized the full cluster as seen in Figure 3.1.

Chapter 4

Energy Efficient Ethernet on MapReduce Clusters

This chapter presents the first analysis from Energy Efficient Ethernet on MapReduce clusters.

4.1 Summary

This chapter estimate the suitability of EEE for applications that follow the MapReduce programming model, in terms of both performance and energy. As mentioned in Chapter 1, EEE was initially analysed for Small Office/Home Office (SOHO) environments, but ongoing efforts are analysing its deployment for data center applications, including video streaming [16] and scientific computing [17]. Since EEE can incur significant performance overheads, many system vendors still advise their customers to disable it in production use [18–20], at least until its impact on real applications is better understood. Yet, there is still opportunity to reduce network energy consumption through energy proportionality, since interconnect links, which consume up to 65% of the total network power [13], always consume full power, even when the link is idle [14].

With this study we found optimum energy savings for 10GbE links only when packet coalescing is enabled. With packet coalescing, switches intentionally delay outgoing packets while the link is in LPI mode, so that they can be transmitted back-to-back with subsequent packets. The packet coalescing settings, however, must be carefully chosen to avoid an excessive loss in performance.

This chapter also presents the discussion to evaluate how to adjust the static packet aggregation settings as a function of the traffic load. At low load, packet aggregation

settings must avoid excessively increasing the latency, and thereby affecting performance. At high load, more aggressive settings are needed to obtain the greatest energy savings. At last, we quantified these recommendations. Throughout our experimentation using suggested settings from literature we feed network equipment manufactures with insights and recommendations for best settings of network cards deployed at the access and aggregation level of data center networks.

In short, the contributions presented on this chapter are threefold:

1. The first evaluation of the performance impact and energy savings from using EEE on a MapReduce cluster.
2. Analysis of packet coalescing, including the tradeoff between performance, energy and load.
3. Dissolution of the different energy profiles of 1GbE and 10GbE links and recommended settings for each.

The rest of this chapter is organized as follows: Section II describes the experimental methodology. Section III presents the quantitative results and analysis, from which Section IV distils the most important recommendations. Finally, Section V concludes the chapter.

4.2 Methodology

This section describes the experimental methodology employed in this chapter. Part of our methodology which is common from this thesis is described in Chapter 3. The specific methodology for this Chapter is described here in this section.

4.2.1 Hardware configuration

The simulated hardware is shown in Table 4.1. We simulate a two-rack cluster with up to 80 nodes, each node having the throughput of a two-core Xeon at 2.5 GHz and a single 1GbE link to the top-of-rack (ToR) switch. Each top-of-rack switch is connected to the aggregation switch using a single 10GbE link. The over-subscription ratio on the 10GbE links is equal to 1.2:1, 2.5:1 or 4:1. Lower over-subscription ratios improve network performance at higher cost [75], so we explored multiple points along this performance–cost tradeoff.

Table 4.1: Simulated Environment

Category	Parameter	Value	
Simulated hardware			
System	Number of nodes	24, 50 or 80	
	Number of racks	2	
Node	CPU	Intel Xeon 2.5 GHz L5420	
	Number of cores	2	
	Number of processors	2	
Network	Each node	*1GbE:* 1	—
	Each top-of-rack (ToR) switch	*1GbE:* ⟨*# Nodes*⟩/2	*10GbE:* 1
	Aggregation switch	—	*10GbE:* 2
Buffers	Shallow buffers	128 KB per port	
	Deep buffers	10 MB per port	
Link power	1GbE	0.5 W	
	10GbE	2.5 W	
Simulated workload			
MapReduce Configuration	Number of job trackers	1	
	Number of workers	23, 49 or 79	
	Maps per node	2	
	Reduces per node	2	
Jobs	Maps per job	*Small jobs:* 10	*Batch jobs:* 2×⟨*# Workers*⟩
	Reduces per job	*Small jobs:* 1	*Batch jobs:* 2×⟨*# Workers*⟩
	Block size per job	*Small jobs:* 64 MB	*Batch jobs:* 2×128 MB
TCP buffer	Default	Max. 64 KB per connection	
	Optimized	Max. 1 MB per connection	

Manufacturers rarely disclose the buffer sizes in the product data sheet, so we followed the best public source we could find [76], giving 128 KB per port for the shallow buffer switches and 10 MB per port for switches with deep buffers.

An important question is the power consumption of the 1GbE and 10GbE links. 1GbE (1000BASE-T) cards were originally expected to consume about 1 W, but current NICs using 110 nm silicon technology require just 0.5 W [65]. On the other hand, 10GbE (10GBASE-T) NICs are still considered to be power hungry. The previous generation, at 40 nm, consumed about 5 W, while the current generation at 28 nm is expected to consume between 2 W and 4 W [66]. Our main contributions are related to energy savings in the 10GbE links, so we conservatively chose relatively power efficient 10GbE links. In summary, as shown in Table 4.1, we assume 0.5 W per port for 1GbE and 2.5 W per port for 10GbE. On real hardware, numbers can still vary depending on the vendor's to be considered.

4.2.2 Workloads

Table 4.1 also shows the configuration of the simulated workloads. We reserve one node for Hadoop housekeeping, to serve as namenode and jobtracker, with the remaining nodes used as worker nodes for processing map and reduce tasks. We chose two workloads, *small* and *batch* (large). The small workload consists of a sequence of small jobs, each with ten map tasks and one reduce task. The average CPU utilization is about 40%, except in Section 4.3.3.4, where it is varied between 5% and 55%. The default 40%

load is consistent with a study of traces obtained at Facebook, which shows that most of the jobs were small, with few maps and one reduce tasks, and that the cluster as a whole had a relatively low utilization of about 40% [77]. The large workload is closer to batch processing for big data applications [78], and we engage the whole system using a single large job, with the number of map and reduce tasks both equal to twice the number of worker nodes.

TABLE 4.2: Simulated benchmarks

Benchmark	% of jobs	Aggregate size		
		Input (MB)	Shuffle (MB)	Output (MB)
Small jobs				
TeraSort	33%	640	640	640
Search	33%	640	0.033	0.033
Index	33%	640	114	114
Batch (large) jobs				
TeraSort (23 nodes)	100%	5888	5888	5888
TeraSort (49 nodes)	100%	12544	12544	12544
TeraSort (79 nodes)	100%	20224	20224	20224

Table 4.2 lists the benchmarks that were used for the evaluation. Each benchmark comprises a sequence of one or more MapReduce jobs, each released at a particular time. The small workload contained a mixture of TeraSort, Search and Index jobs. The batch workload contained a single TeraSort job. Batch processing normally involves large jobs of several gigabytes or terabytes, but the communication, most of which is in the shuffle stage, is close to proportional to the workload size. Since the communication pattern is also repetitive, we can obtain representative figures using a workload of 128 MB per core, which is sufficient to maximise cluster utilization.

Since packet coalescing can increase latency, which implies more buffering in software, we present results for two different values for the maximum TCP buffer size per connection: the default value of 64 KB and an optimized setting of 1 MB. The optimized setting also enables the *TCP Window Scale* option, which allows the congestion window to grow above 64 KB. The default value of 64 KB is known to be small, so in production use the global settings must be changed and the application restarted [79].

4.2.3 EEE settings

We assume the sleep and wake timings given in Table 2.1, and evaluate several control algorithms. We begin by evaluating Power Down Threshold [48], or stall timer, without the use of packet coalescing. We use the best stall timer value, with the packet coalescing settings in Table 4.3.

Table 4.3: EEE packet coalescing settings

Label	Holding time	Trigger
nopa	No Packet coalescing	
12us10	12 µs	10 packets
120us100	120 µs	100 packets
1ms1000	1 ms	1000 packets
10ms1000	10 ms	1000 packets

Finally, we include an *ideal* case, for which sleep and wake transitions are both instantaneous and zero energy. In this case, the link is optimally controlled by simply entering LPI mode as soon as it becomes inactive, providing perfect energy proportionality without affecting runtime. This result gives a lower bound on energy consumption.

4.2.4 Summary of configurations

In summary we have the following configurations:

Number of nodes	24, 50 *or* 80
Switches	Shallow *or* deep buffers
Workload	Small jobs *or* batch job
TCP window size	Default *or* optimized
Packet coalescing	See previous subsection

4.2.5 Total runtime vs. Average runtime

On a cluster running several jobs in parallel, the performance can be measured using either the total execution time for all jobs (**total runtime**) or the average execution time per job (**average runtime**). Each benchmark contains multiple MapReduce jobs, and the total runtime is the wallclock time from the start of the first job to the end of the last job. A single job's runtime is the wallclock time from the time the job is ready to be scheduled until it finishes executing, and the average runtime is the average of these single job runtimes.

The MapReduce scheduler plays an important role in both total runtime and average runtime, and extensive research has been carried on this topic [80]. The MrPerf MapReduce simulator provides multiple scheduler implementations, but the recommended scheduler is Quincy [81], which provides fair scheduling with data locality [82].

Total runtime and average runtime both grow with cluster utilization, but the relationship between them depends on the scheduler. Figure 4.1 shows the behavior using the Quincy scheduler. Although both grow linearly, the total runtime widens in relation to the average runtime as the load increases once it is more impacted by the time completion of the last job. A detailed analysis of the effect of the scheduler is outside the scope of this work, which focusses on the performance–energy tradeoff of Energy Efficiency Ethernet.

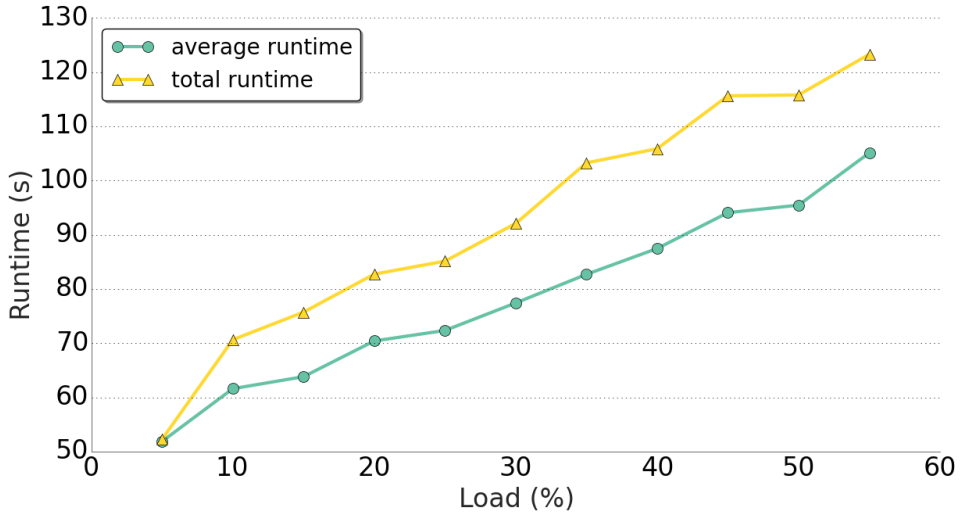


FIGURE 4.1: Average and total runtime vs. load (Quincy scheduler)

4.3 Results

This section presents the quantitative results, giving the energy savings and performance overheads for MapReduce workloads using Energy Efficient Ethernet.

4.3.1 Fixed link latency

Since EEE primarily affects execution time via its effect on latency, we begin by evaluating the effect of link latency on MapReduce performance. We added a fixed latency on each link, without using EEE, for both workloads: small tasks and batch processing. This experiment used the default TCP settings for the receive and send buffers and the scale window.

As shown in Figure 4.2, for small tasks the runtime begins to increase only when the latency per link exceeds about 100 µs. Batch processing is less sensitive to latency:

the performance starts to degrade only when the latency exceeds about 5 ms per link. The difference between the two is that batch processing has most of its communication concentrated during a single shuffle phase, whereas small tasks have the communication more distributed over time. Since small tasks experience less congestion, the baseline bandwidth is higher, and a smaller latency is sufficient to exceed the buffers.

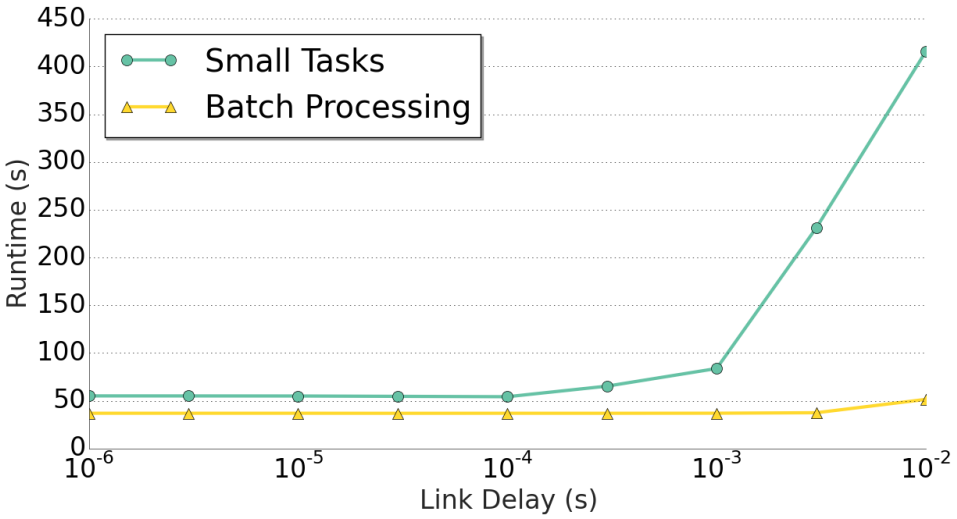


FIGURE 4.2: Runtime vs. fixed link latency per link

We conclude that, for both workloads, and even with the default TCP settings, the impact of the 1GbE wakeup latency of 16.5 µs on MapReduce performance should be negligible. Since the latency is added per link, these results already include the effect of consecutive wakeups on multiple hops.

The simplest EEE control algorithm puts the link into Low Power Idle (LPI) mode as soon as it becomes inactive [32]. A more advanced method, known as Power Down Threshold or stall timer, enters LPI mode after a defined period of inactivity, which is typically about 50 µs (see Chapter 2.1). If the stall timer is small, then the link may frequently enter and leave LPI mode, incurring a large performance penalty. On the other hand, if the stall timer is large, the links will seldom enter LPI mode, yielding poor energy savings. We would therefore expect to reproduce previous findings that the stall timer provides a trade-off between performance and energy [17].

We evaluated the effect of the stall timer setting, as shown in Figure 4.3 (runtime) and Figure 4.4 (energy consumption per port). Figure 4.4 shows the average energy consumption of the 1GbE and 10GbE ports, as well as the average of all ports of the data center (DC). These results show that, for MapReduce, the stall timer offers little advantage over the simple algorithm, since, even for our worst-case results, a stall timer of zero gives a small performance overhead of about 1% that is hard to distinguish

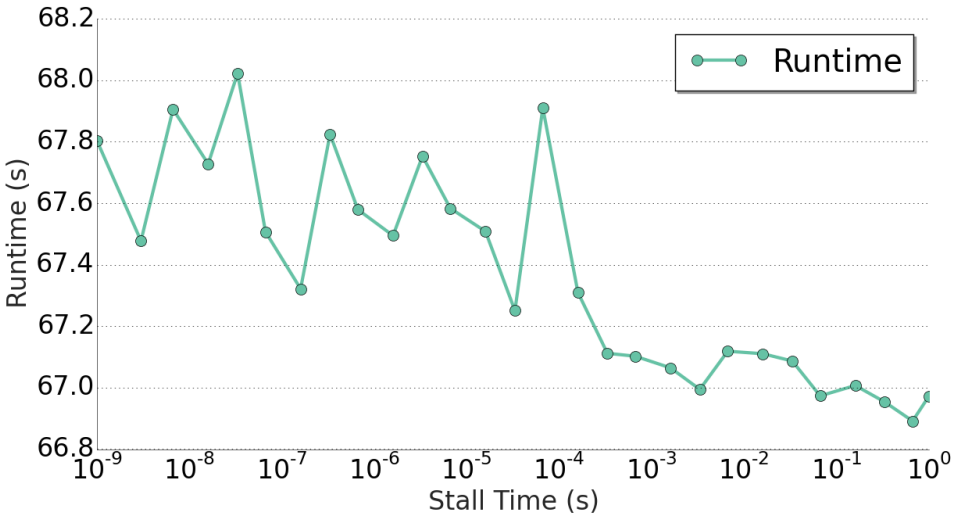


FIGURE 4.3: Runtime vs. stall time without packet coalescing (small tasks)

from scheduling and other noise. We therefore use the simple control algorithm without packet aggregation (nopa), which we refer to as *Standard EEE*.

4.3.2 Standard EEE and stall timer

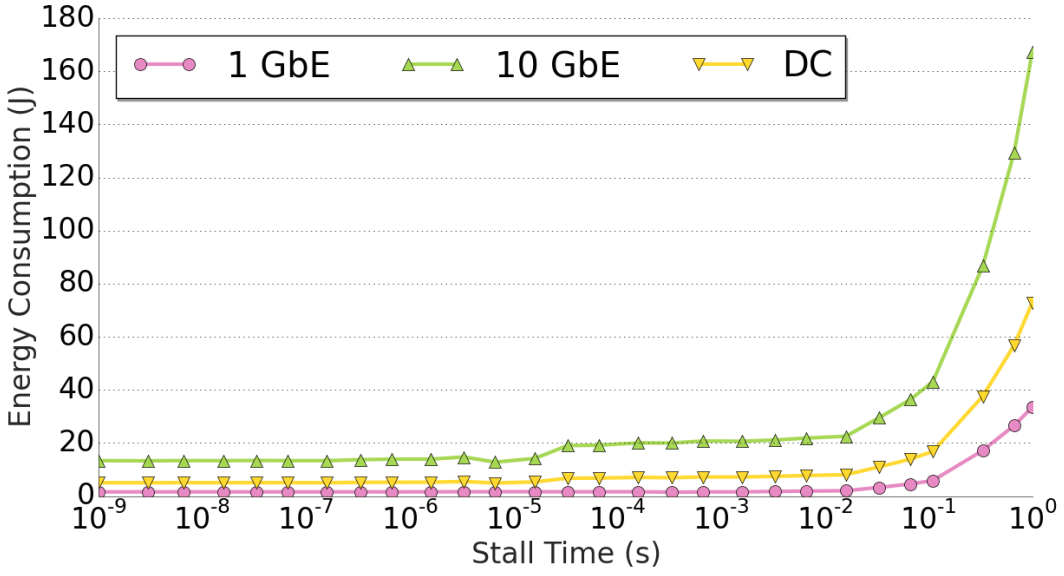


FIGURE 4.4: Average energy per port vs. stall time without packet coalescing (small tasks)

Figure 4.5 compares the energy consumption of legacy Ethernet (without EEE) and Standard EEE. It also shows the ideal case, which has instantaneous zero energy sleep

and wake transitions. In this figure and in the next subsection, energy consumption is always normalized to the current state of the art, which is Standard EEE (without packet coalescing). In Figure 4.5, standard EEE reduces the energy consumption by a factor between five and eight, depending on the workload and network over-subscription ratio. The ideal results, which are closely matched using packet coalescing, show a further factor of two improvement.

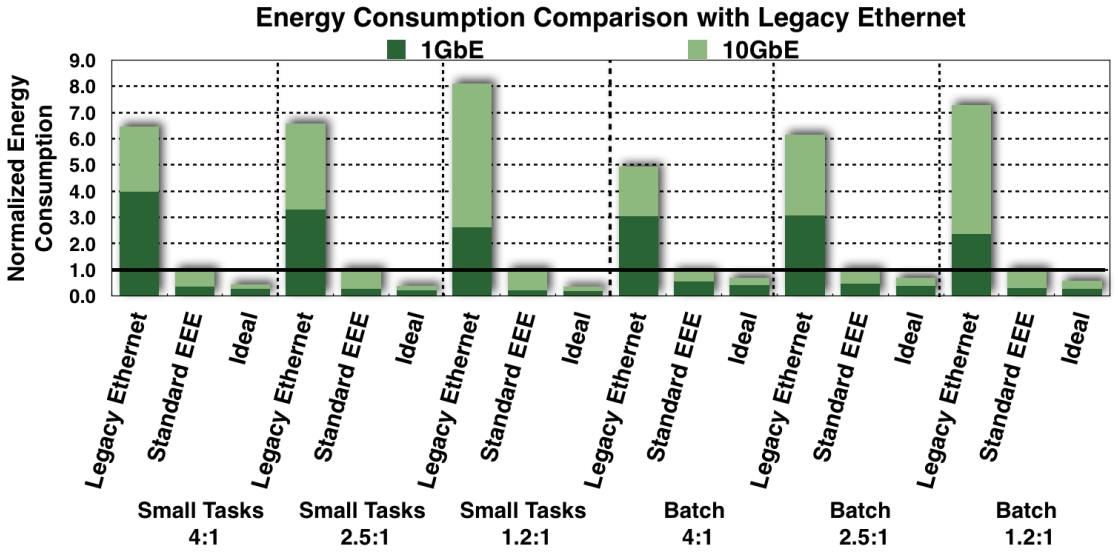


FIGURE 4.5: Average energy consumption (comparison with legacy Ethernet)

4.3.3 Optimum Energy Savings on MapReduce Cluster

This section investigates the optimum settings for EEE across the whole network. The results show that, in contrast to previous recommendations for the deployment of EEE [14, 33], packet coalescing should be enabled for the 10GbE links, but it is necessary to carefully choose the packet coalescing parameters and TCP settings. *All results in this section were normalized to the standard Energy Efficient Ethernet (without packet coalescing).*

4.3.3.1 Uniform EEE settings

The broad results in more detail are shown in Figure 4.6. Figure 4.7 displays more details as it brings the same results zoomed-in. Results are given for the five packet aggregation settings from Table 4.3, with default or optimised TCP buffers, and shallow or deep switch buffers. The main conclusion is that, with shallow buffer switches, with 64 KB per connection, the 1ms1000 and 10ms1000 settings have unacceptably large overheads.

The same results are summarized in Figure 4.8, which shows the performance and energy results averaged across all six scenarios (workloads and over-subscription ratios).

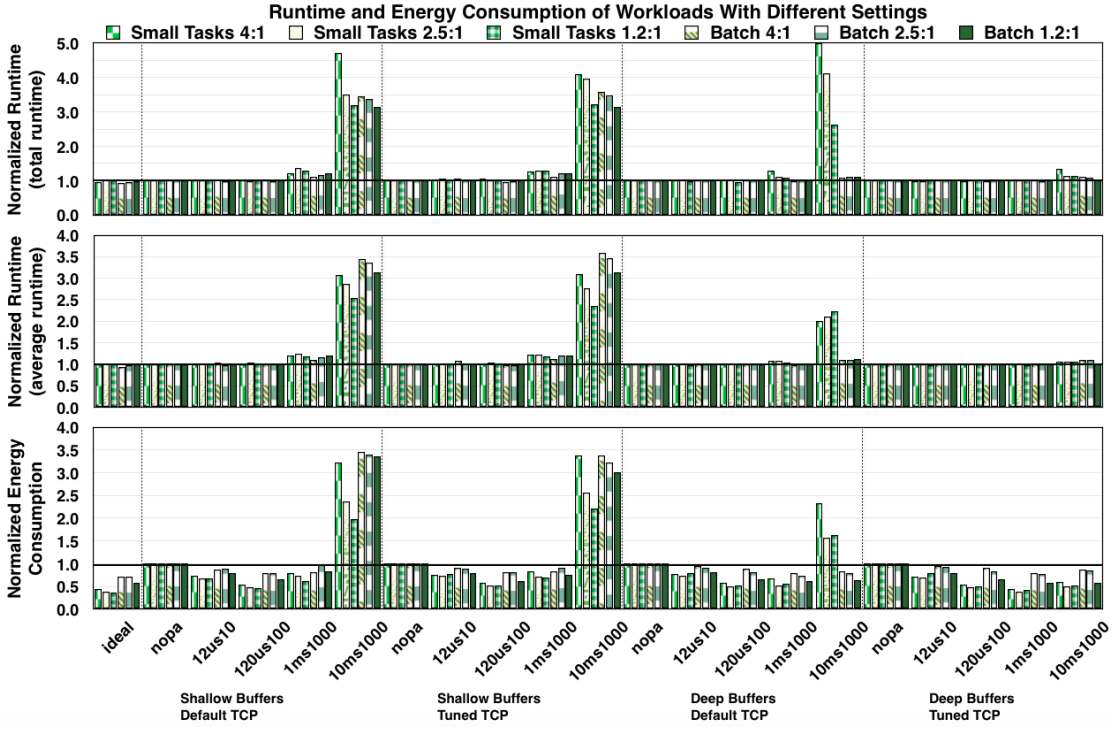


FIGURE 4.6: All runtimes and energy consumption of workloads using different settings

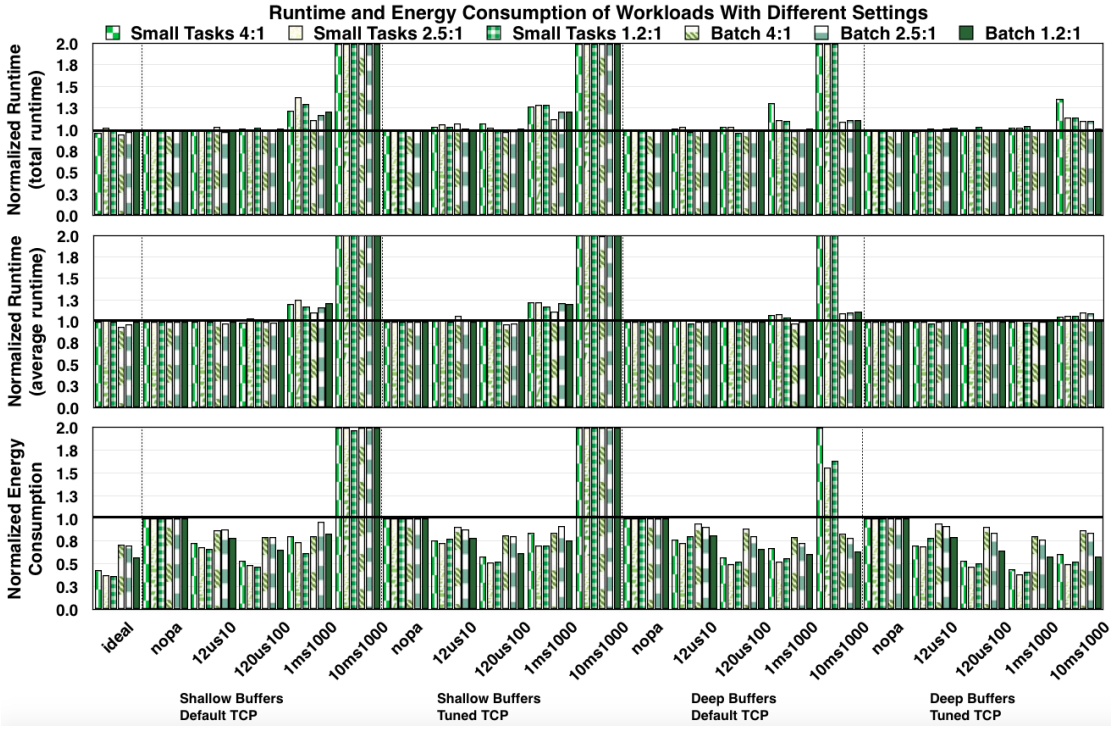


FIGURE 4.7: All runtimes and energy consumption of workloads using different settings (Zoomed-in)

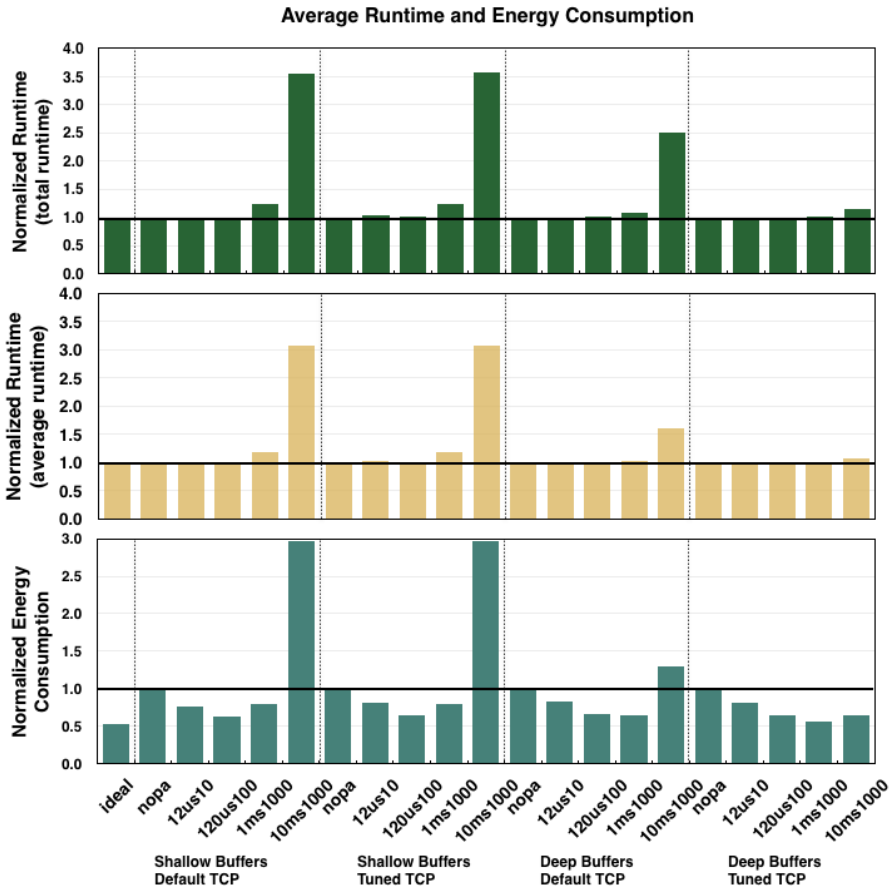


FIGURE 4.8: Average Runtime and Energy Consumption of MapReduce jobs

Regarding performance first, the 12us10 and 120us100 settings have overheads of less than 5%, for all six scenarios, with little variation among the scenarios. With deep buffer switches, of 1 MB per connection, the 1ms1000 setting is also acceptable for batch workloads, with or without tuned TCP settings. Batch workloads fully utilise the network during the shuffle phase, and the resulting congestion means that full throughput can be achieved using a relatively small TCP congestion window. This traffic is, in fact, sufficient to usually trigger the 1000-packet threshold without waiting for the timeout, giving lower additional latency and also a similar runtime for 1ms1000 and 10ms1000. In contrast, with small tasks, the shuffle phases of different jobs happen at different times, so network utilization is spread out in time and there is less network congestion. For small tasks, the additional latency introduced by aggressive packet coalescing therefore requires a tuned TCP congestion window.

The average runtime follows the same pattern as total runtime, except for deep buffer switches and default TCP using 10ms1000 setting. Surprisingly, we see that scenarios with more servers show a slightly lower performance degradation than scenarios with fewer servers. In other words, lower over-subscription ratios lead to greater performance

degradation, contrary to what would be naively expected and also to what is seen on the rest of the results. On such situation switch buffers are not the bottleneck of the system, and having more servers offers the scheduler the possibility to achieve better average runtime, even if the same is not verified on total runtime.

Turning to the energy results in Figure 4.8, it is clear that the best packet aggregation settings depend on the context. With deep buffers and tuned TCP settings, the best energy savings are obtained using 1ms1000: the energy consumption was reduced to 55% at an overhead of less than 2%. With shallow buffers, the same settings would increase the total runtime by an unacceptable 25%. The best settings for shallow buffers are 12us10 and 120us100, which increase runtime by less than 1% but reduce energy to 75% and 60% of Standard EEE, respectively. The next experiment consists in using 1ms1000 for the 10GbE NICs, and 12us10 or 120us100 for 1GbE links. We expect to save additional energy and get closer to the ideal model.

4.3.3.2 Non-uniform EEE settings

In Section 4.3.3.1, the packet aggregation settings were uniform across all switches in the network. This section investigates the benefit of *non-uniform* packet aggregation settings. Specifically, the new settings, 12us10+ and 120us100+ are the same as 12us10 and 120us1000, respectively, for the 1GbE links, but they use 1ms1000 on the 10GbE links.

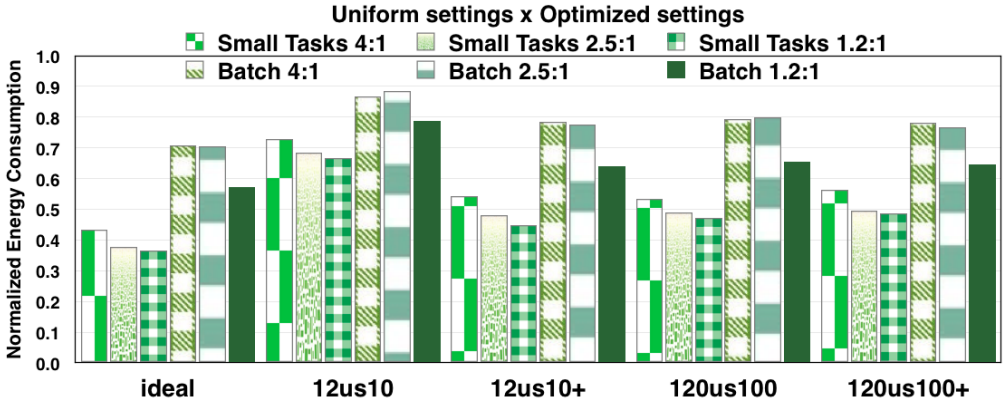


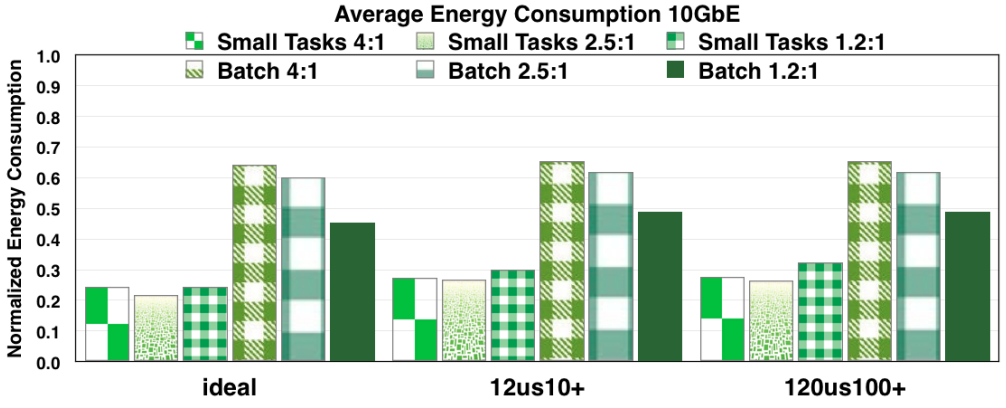
FIGURE 4.9: Energy consumption (optimized configuration) [runtime remains about the same]

As shown in Figure 4.9, using different settings of packet aggregation for different NICs improved the energy savings. The aggregation switch has better energy savings using 1ms1000, since 12us10 and 120us100 present good savings for a moderate load but not for high load. Under high load, 1ms1000 gets closer to ideal savings for 10GbE links (12us10+ and 120us100+).

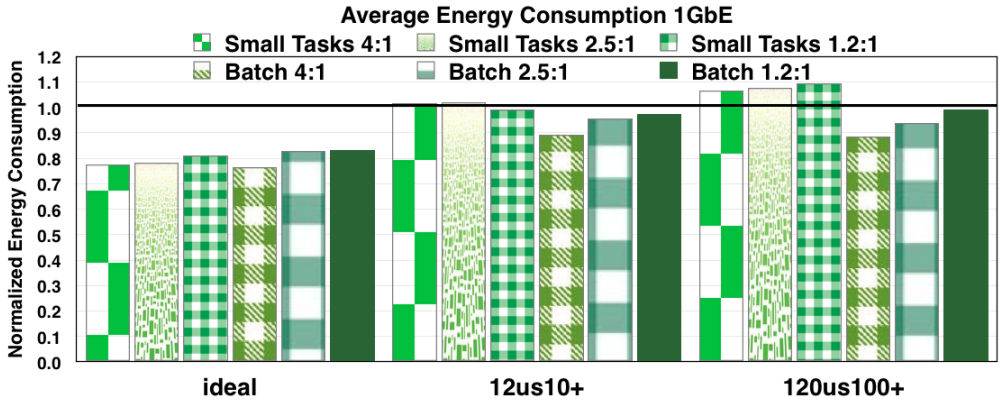
4.3.3.3 Analysis by link type

Whereas Figure 4.9 showed the average energy consumption across all the network links, Figure 4.10 shows separate energy consumption results for (a) the 10GbE links and (b) the 1GbE links. As before, values are normalized in comparison with Standard EEE.

The greatest savings are obtained for the 10GbE links, which benefit most from packet coalescing, as shown in Figure 4.10a. In contrast, Figure 4.10b shows little benefit from packet coalescing for the 1GbE links in comparison with Standard EEE. Considering the complexity and the cost for new 1GbE interfaces that would deploy packet coalescing for almost negligible benefits, the adoption of packet coalescing technique on edge interfaces is not necessary. For this reason, our last experiment (see the following results) focus on using packet coalescing technique only on 10GbE link (aggregation layer). 1 GbE links remain with Standard EEE, which means no packet aggregation on edge links.



(A) 10GbE



(B) 1GbE

FIGURE 4.10: Detailed energy consumption by NICs

4.3.3.4 Load impact on coalescing settings

Packet coalescing settings were previously evaluated for a fixed typical cluster utilization of 40%, measured by the CPU load. This section extends the evaluation to consider to what extent the conclusions depend on the CPU load. It complements previous work [14] discussed in Section 2.5, which explored the relationship between network utilization and *power* for Poisson arrivals. For workloads like MapReduce, the connection between cluster utilization and energy is more complicated, due to the non-linear relationships between cluster utilization, measured using the CPU load, and both network utilization and runtime.

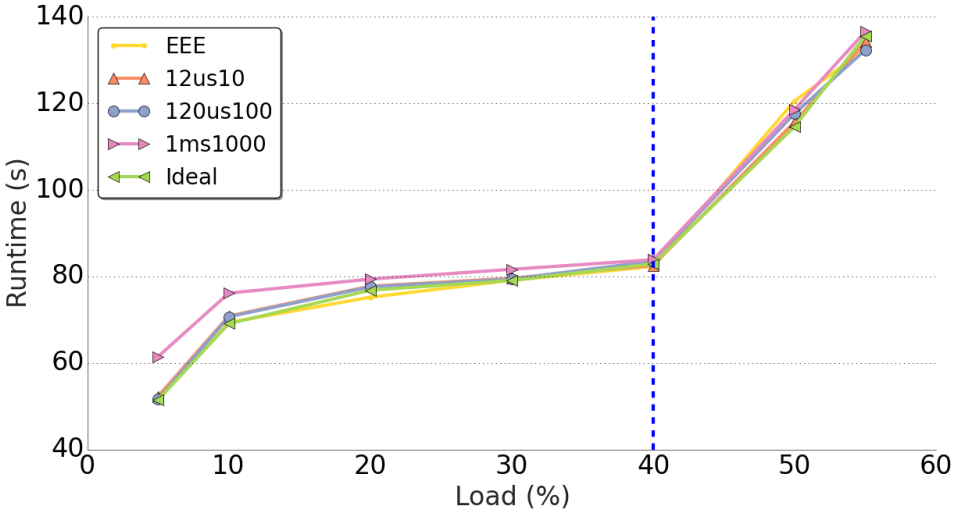


FIGURE 4.11: Total runtime as CPU load is varied (Terasort)

We concentrated on Terasort jobs on switches with shallow buffers, and varied the CPU load between 5% to 55%. Figure 4.11 shows the runtime results. The differences between ideal, standard EEE, 12us10 and 120us100 are small, whereas 1ms1000 has a large overhead below 40% utilization, rising to a performance degradation of 20% at 5% utilization. The extra delays of up to 1 ms, caused by packet coalescing, require a larger Bandwidth–Delay Product (BDP), and more in-flight packets to compensate for the extra latency. Switches with shallow buffers cannot accommodate the Incast congestion that happens on the link connecting the Top-of-Rack switch to the reduce node (see Chapter 2.5). It is therefore impossible to increase the BDP, even using optimized TCP settings, because congestion limits the number of packets in flight. Note that this problem does not happen with deep buffer switches.

Turning to the energy consumption, shown in Figure 4.12, the lowest energy consumption is always achieved using 1ms1000. As previously mentioned, however, below about 40%

utilization, the increase in runtime, visible in Figure 4.11, becomes too large, so the best option, with small difference in energy, would be 120us100.

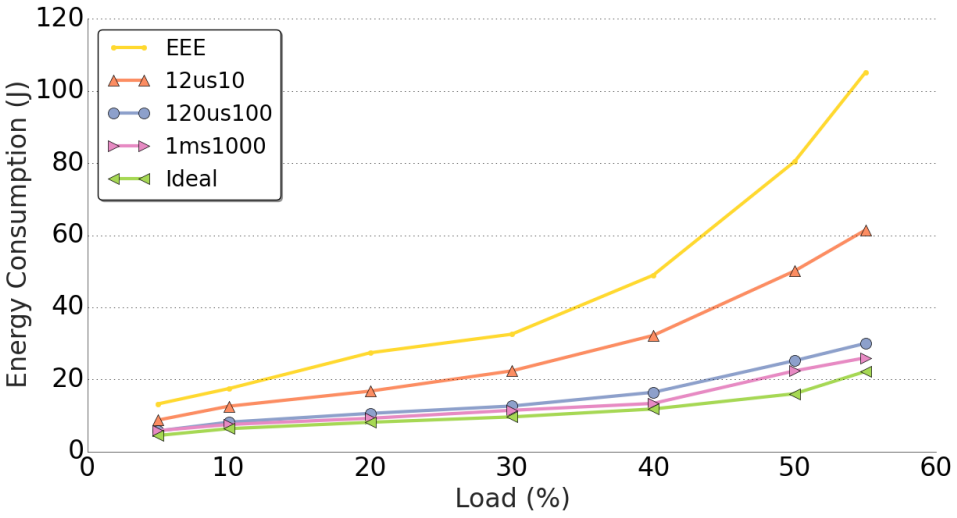


FIGURE 4.12: Energy consumption as CPU load is varied (Terasort)

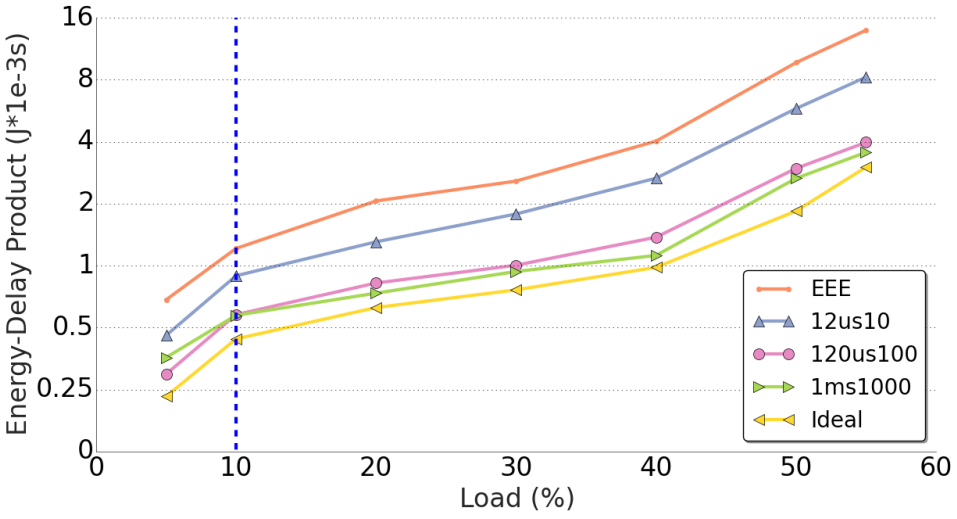


FIGURE 4.13: Energy–delay product as CPU load is varied (Terasort)

Finally, Figure 4.13 combines performance and power into a single metric, which shows the normalized energy–delay product. When cluster utilization is higher than 10%, the energy–delay metric in fact indicates a better trade-off for the 1ms1000 setting. On the other hand, when the cluster utilization is lower than 10%, the 120us100 setting has a better trade-off. Back to 1ms1000, it is important to notice that even with better results from the energy–delay metric, from 10% to 40% of cluster utilization, the better energy savings are achieved at the cost of a performance penalty previously mentioned.

This therefore has to be taken into account when choosing the proper packet coalescing setting.

4.4 Discussion and Recommendations

Recent switches that implement Energy Efficient Ethernet (EEE) already have support for energy proportionality, but until a good understanding of the impact on real application performance has been reached, these features are likely to remain switched off in practice, unnecessarily increasing the energy consumption. This chapter contributes to the necessary understanding by analysing this tradeoff in detail for MapReduce workloads, which are representative of modern applications dominated by east–west traffic.

The recommendations in this section have been divided into a) recommendations for system administrators, and b) recommendations for equipment vendors.

Recommendations for system administrators: The first finding of this chapter is that the "standard" Energy Efficient Ethernet algorithm, which turns the link off immediately when it becomes idle, and that doesn't use packet coalescing, obtains significant energy savings with negligible performance overhead. This is in contrast to previous work in the context of HPC, which found that a Power Down Threshold timer was needed to limit the performance overhead of repeated link wakeups [17]. The result is that system administrators should not be afraid to enable EEE, even when the general guidelines from the system integrator are to disable it.

Recommendations for equipment vendors: The limited configurability of existing switches mean that the remaining recommendations are currently targeted at equipment vendors. We show that the standard algorithm is, in fact, sufficient for 1GbE edge links. An ideal model, that sets EEE overheads to zero, gives an upper bound energy consumption of 20% below that of EEE. Our investigation of packet coalescing settings, however, found a maximum benefit of just 5%.

For 10GbE, however, packet coalescing delivers further energy savings, reducing the energy consumption to half or less. For this reason, equipment vendors should certainly implement packet coalescing, especially for 10GbE links. We were able to find packet coalescing settings, for all evaluated loads and workloads, that save between 35% to 75% more energy, in comparison with standard EEE, and that reach close to the ideal case (Figure 4.12).

We found, however, that the energy–performance tradeoff is strongly affected by the packet coalescing settings. Coalescing packets does not simply introduce a delay on links. It also increases the burstiness, which, especially on shallow buffers, leads to packet loss, ultimately impacting performance. This is especially true for workloads like MapReduce that have a many-to-one communication pattern. We extended our previous analysis [27] by investigating the effect of cluster utilization. When utilization is low, the best setting was 120us100, which provided good energy savings without hurting performance. When utilization is high, a more aggressive setting of 1ms1000 obtained better energy savings with little performance loss.

Some server NIC vendors implement a technique known as interrupt coalescing, which amortises the overhead of interrupting the CPU, by generating a single interrupt to process multiple received or transmitted packets [83]. Interrupt coalescing is most commonly deployed on the receiver side, in which case it has no effect on how packets are presented on the network links, and it is therefore not directly relevant to this study of Ethernet energy efficiency. When deployed on the transmitting side, interrupt coalescing may be an effective means of implementing packet coalescing on the outgoing edge links. Higher-bandwidth network links would still need packet coalescing to be implemented in the switches. We did not model interrupt coalescing in this study, but our results indicate that relatively aggressive interrupt coalescing, on both transmit and receive, would not be expected to significantly impact Hadoop performance.

We investigated the effect of over-subscription on these findings, as shown in Figure 4.9. Generally speaking, the larger over-subscription ratios use a smaller number of 10GbE links, so the energy savings in these links have lower effect on the total network energy consumption. Nevertheless, even with the largest over-subscription factor of 4:1, packet coalescing reduced the energy consumption by 20%. The benchmarks with small tasks distributed network utilization over longer periods of time, during which the utilization was lower, leading to a greater benefit from packet coalescing.

How buffering and burstiness really affect Hadoop: The results presented here are aligned with a study presented on the next chapter, which investigated the impact of using DCTCP and also AQM queues combined with ECN to reduce buffer occupancy on Data Centers. The best performance on batch workloads is achieved using deep buffers switches without Active Queue Management on the network buffers. Contrary to the naive assumption, TCP works well under congestion and burstiness scenarios as long as there is enough buffer to accommodate the bursty traffic. Limiting buffer utilization can also degrade performance of batch workloads such as Hadoop [29].

4.5 Conclusions

An important challenge of modern data centers is to reduce energy consumption, of which a substantial proportion is due to the network. The Energy Efficient Ethernet (EEE) standard, approved by IEEE in 2010, implements low-level mechanisms to improve Ethernet energy efficiency. Such standards, however, will not be adopted in practice until their effects on workload performance are well understood.

We evaluated the performance impact and energy savings, and found that the MapReduce programming model is not sensitive to the overheads of EEE, even with packet coalescing, contradicting the general guidelines from vendors to disable EEE. For 1GbE links, it is sufficient to switch links off as soon as they become idle, but optimum energy savings in the 10GbE links are only possible with packet aggregation. We therefore suggest to adopt a simple management algorithm for edge devices (1GbE), and to enable the system administrator to modify the packet coalescing parameters for core devices (10GbE). This approach improves the energy savings between 20% and 60% in comparison with standard EEE, depending on the workload and the network over-subscription ratio. At last, our findings widen on this study demonstrated how cluster utilization plays an important role when choosing the settings for packet coalescing.

Chapter 5

Controlling Delay Mechanisms on MapReduce Clusters

This chapter presents the first analysis of mechanisms that can be employed in Hadoop clusters to control packet delay.

5.1 Summary

For Big Data workloads, the highest performance is achieved using expensive network equipment with large buffers, which are better able to accommodate congestion and network traffic bursts. Large buffer switches, however, suffer from the bufferbloat phenomenon, in which TCP (greedily) makes full use of the available buffers, even when maximum performance can be achieved using much less buffering. Bufferbloat has been found to cause excessive packet delays within data centers [26]. Nevertheless, it is reasonable to expect that bufferbloat would have little direct effect on Hadoop, since its communication is dominated by long network flows.

Throughput-sensitive big data applications are, however, often executed in the same data center as other workloads that directly interact with external users, and these workloads *are* sensitive to network latency. In this case, the network is shared between both classes of application, so it should therefore provide not only the maximum possible throughput, but also the lowest possible latency.

This chapter presents the first analysis of mechanisms to control packet delay in a Hadoop cluster. We study Active Queue Management (AQM) and Explicit Congestion Notifications (ECN), both of which are supported in modern network switches. We perform a quantitative evaluation of the tradeoff between throughput and latency, for

four different approaches for controlling buffer occupancy and latency: Random Early Detection (RED) and CoDel queues, both standalone and also combined with ECN and Data Center TCP (DCTCP). This is done in the context of MapReduce and Apache Hadoop [35], which present a specific traffic pattern in the shuffle phase.

Our work provides recommendations to administrators of Hadoop clusters. We show experimental results at the network level, in terms of network throughput and packet latency. More importantly, we also show the impact on Hadoop job execution time. Previous analysis suggested that CoDel and other techniques that reduce buffer occupancy would also translate to better performance during Hadoop's all-to-all communication phase [54]. We find that TCP already functions well, so Hadoop execution time is not improved by such techniques. Moreover, in some cases a poorly-chosen AQM configuration increases the execution time by an unacceptable 20%. We do, however, identify good AQM configurations that are able to maintain Hadoop execution time gains from larger buffer to within 5%, while reducing packet latency caused by bufferbloat by 85%.

In short, our contributions are threefold:

1. A study of mechanisms that can be employed on Hadoop clusters to control packet delay.
2. A quantitative evaluation of the tradeoff between throughput and latency for RED and CoDel queues, both standalone and also combined with ECN and DCTCP.
3. Recommendations to cluster administrators to improve latency without degrading throughput.

The rest of the chapter is organized as follows: Section II presents the specific methodology for this chapter while Section III presents the quantitative results and analysis, from which Section IV distills the most important recommendations. Finally, Section V concludes the chapter.

5.2 Methodology

This section describes the experimental methodology specific from this chapter.

Table 5.1: 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. 300 KB per port
	Expensive switches	2000 packets - max. 3 MB per port

5.2.1 Simulation Environment and Workload Characterization

The simulated hardware is shown in Table 5.1. We simulate a 4-rack cluster with 40 nodes per rack, each node having the throughput of a two-core Xeon at 2.5 GHz and a single 1GbE link to the top-of-rack (ToR) switch.

We provide results for both shallow and deep buffer switches. Hadoop clusters often use inexpensive commodity switches, which have small (shallow) buffers. Small buffers can cause excessive packet loss over bursty communication, and network equipment vendors are already promoting deeper buffered equipments for Hadoop clusters as seen in Chapter 2.5. Tasks such as small queries are typically completed in a latency of a few milliseconds and therefore, they tend to be severely impacted by packet drops. In some cases it can translate into a performance loss that reaches a full second due to several retransmission timeouts [24]. Small buffering can also lead to unfair bandwidth allocation between different flows [24]. In our experiments we found that for long-lived TCP flows with bursty communication, fully avoiding packet loss by using larger buffers can improve TCP throughput performance by about 10%, which means the time spent on communication during the shuffle phase of Hadoop can decrease compared to results obtained by using shallow buffer switches. Yet, as seen in recent work [21], restricting the buffer utilization of network equipment can also translate into a performance degradation of 20% for batch workloads such as Hadoop. Related work comparing Active Queue Management tends to use packets instead of buffer size so we selected two different values for queue size: 200 packets or 2000 packets. For packets using the maximum payload size of 1500 bytes the maximum capacity per port is 300 KBytes or 3 MBytes respectively. Additionally, for the average payload size of 500 bytes the average capacity per port is 100 KBytes or 1 MByte respectively.

Table 5.1 also shows the configuration of the simulated workloads. We reserve one node for Hadoop housekeeping, to serve as namenode and jobtracker, with the remaining nodes used as worker nodes for processing map and reduce tasks. On our previous work presented in Chapter 4 [27] [28] we simulated a smaller cluster with different type of workloads to understand the nuances found on a MapReduce cluster. We found small variability, most caused by the Hadoop scheduler. As for this work we expanded the size of our cluster, to limit the noise introduced by different scheduling decisions, we used MRPerf (see Chapter 3.1) to generate a single Terasort job configured to sort 6.4 GBytes (random elements) with 100 mappers. Using more nodes on this experiment would lead to numbers where the shuffle phase would represent a much wider proportion of the total runtime, as seen in similar experiment [84]. Terasort is a popular batch benchmark commonly used to measure MapReduce performance on a Hadoop cluster. In order to make it representative we characterized the shuffle phase as shown in Figure 5.1, to be consistent with a study of traces obtained at Facebook, which shows that most of the jobs were small [77] and the shuffle phase which represents in average 33% of the jobs execution time can be slightly more than 50% of the total runtime for nearly one quarter of the jobs which have reduce phase [84]. Shuffle is considered the MapReduce phase that mostly stresses the network because its all-to-all communication between mappers and reducers.

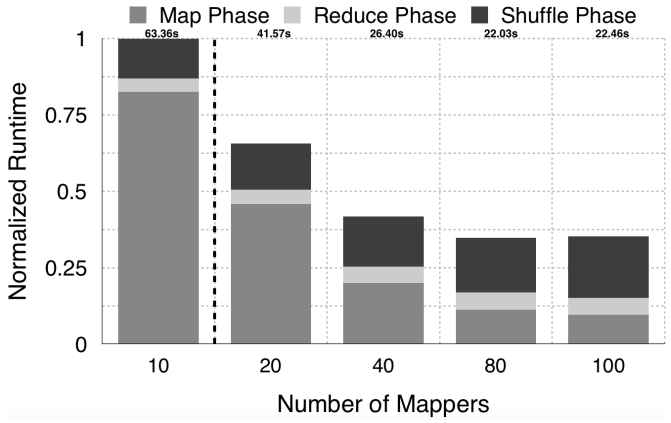


FIGURE 5.1: Shuffle Characterization

With the purpose of having a scenario where the network is characterized as the bottleneck of the system, we run the experiment as mentioned before. Figure 5.1 was obtained using shallow buffer switches, running the same Terasort job and changing the number of map inputs and the chunk size value (the total size of the Terasort task) while fixing the number of reducers to the recommended, which is the number of workers in order to use the full system [85]. In a real cluster is recommended to use the factor 0.95 and leave a few nodes free in case of node failures. Since MRPerf does not simulate failing nodes we modeled the Terasort task divided by servers, each of them handling a proportion of the output from the map nodes. As our cluster has capacity of 2 mapslots and 2

reduce slots per node it gives us relatively low utilization of not more than 40% which also matches with production tracers [77]. The communication, most of which is in the shuffle stage, is close to proportional to the workload size, but increases as the network becomes the bottleneck. It leads to a situation where the servers would spend more time with data transfer and therefore, since the data size that each reducer receives is constant, increasing the time spent during the shuffle phase also increases the reduce phase (final computation phase of Hadoop). In other words, the benefits of speeding up the initial computation phase (map phase) are not translated into a linear decrease of the total computation time once the linear increment on communication also affects the time completion of the final computation phase (reduce phase). Since the communication pattern is also repetitive, we can obtain representative figures using only one task as network and cluster utilization were proportionally designed based on available tracers.

We use three performance metrics: the *runtime* which is the total time needed to finish the Terasort workload, found to be inversely proportional to the effective throughput of the cluster; the *average throughput* per node and the *average end-to-end latency* per packet.

For throughput and runtime we used the same baseline for the whole set of results which is the DropTail queue for shallow buffers. This way we were able to compare improvements on runtime when using deep buffer equipments as promoted by new vendors.

For latency we considered two baselines. The naive assumption would be assuming congestion is something rare that happens only when there are peaks during the communication. On networks, congestion happens all the time, which means it is the steady state of the network. The goal of any transport protocol as TCP is to maximize the usage of the network. TCP, or any other congestion protocol, will be probing the network, trying to find how many packets the network can carry until it loses the packet and then back-off. On other words, TCP is always pushing the network into congestion and then backing-off. Using deeper buffer equipments will automatically increase the average delay per packet in order to obtain some gain on throughput and bursty tolerance. It is a trade-off between adding extra latency to the network while having more tolerance to bursty communication and also obtaining a higher throughput. For this reason we considered the DropTail queue of each set of simulations as the baseline for latency. When comparing the delay on shallow buffers, the smaller DropTail queue is the considered baseline. The same happens with the deep buffer set of results, which means the larger DropTail queue is the baseline for latency on bufferbloat scenarios.

Although the techniques utilized on this work could be used to obtain some improved figures regarding the tail latency, their focus is on reducing the queue length and the

average latency of the network [86]. Reducing queue utilization and average latency is the first step towards the employment of new applications to run in parallel while sharing the same network resources on Hadoop clusters. We consider that a proper study focused on reducing the network tail latency is an important requirement that should be carried on, specially when scaling up the cluster size. We also point it as future work to be derived from this thesis as seen on Chapter 8.

5.3 Results

This section presents the quantitative results, giving the runtime, throughput and latency for Hadoop using control delay mechanisms.

5.3.1 Random Early Detection (RED)

The default implementation of Random Early Detection queue needed to be adapted for the high-speed networks found on data centers. RED is typically implemented using the average queue length for decisions of marking or drop of packets. We tried to use the average queue length on our experimentation but our results were too similar to the DropTail queue. Therefore, to allow simplification we do not include them. Instead, we used the instant queue length on all our simulations with RED queue. We don't claim novel on this as previous works already demonstrated that instant queue length fits better for high speed networks as within the data center, we just confirmed and emphasize what was already demonstrated on previous evaluation [40].

RED offers an auto configuration setting that needs only one parameter: the target delay [44]. Such feature takes in consideration the speed of the network interface card and based on the "packet time constant", which is the maximum number of average sized packets that can be transmitted per second. If the network interface is fast enough and the target delay is also big enough, the thresholds will not be useful and the queue will behave as a DropTail queue. We couldn't find any references on literature so we performed a sweep using different values as found in Table 5.2. Results are found in Figure 5.2.

We also considered different values found on previous publications and also proposed new values. Our considered values for RED are found in Table 5.3. Results for fixed settings are found in Figure 5.3.

It is easy to verify that RED auto configuration feature shows a clear tradeoff between latency and throughput. It seems to simplify the process of tuning the queue as we can

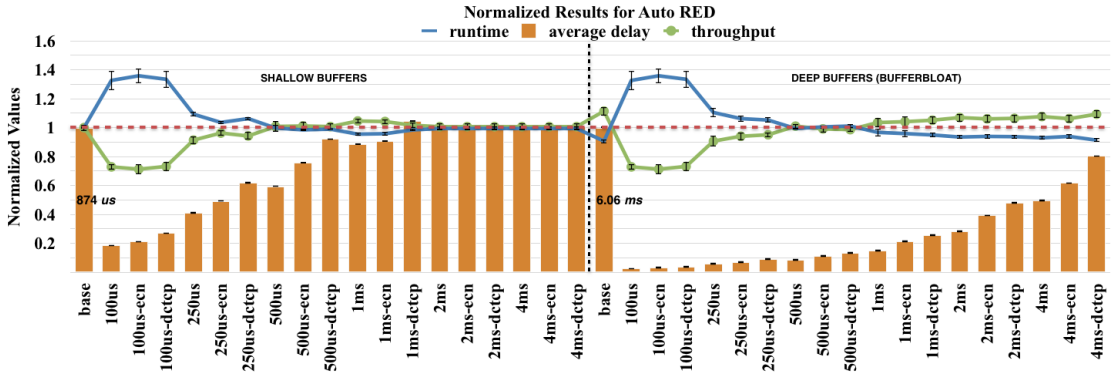


FIGURE 5.2: Normalized Results for Auto RED Queue

TABLE 5.2: Auto Random Early Detection Settings

Target delay	1 GbE thresholds	10 GbE thresholds
100 µs	12.5 -37.5	125 - 375
250 µs	31.25 - 93.75	312.5 - 937
500 µs	62.5 - 187.5	625 - 1875
1 ms	125 - 375	NO AQM
2 ms	250 - 750	NO AQM
4 ms	500 - 1500	NO AQM

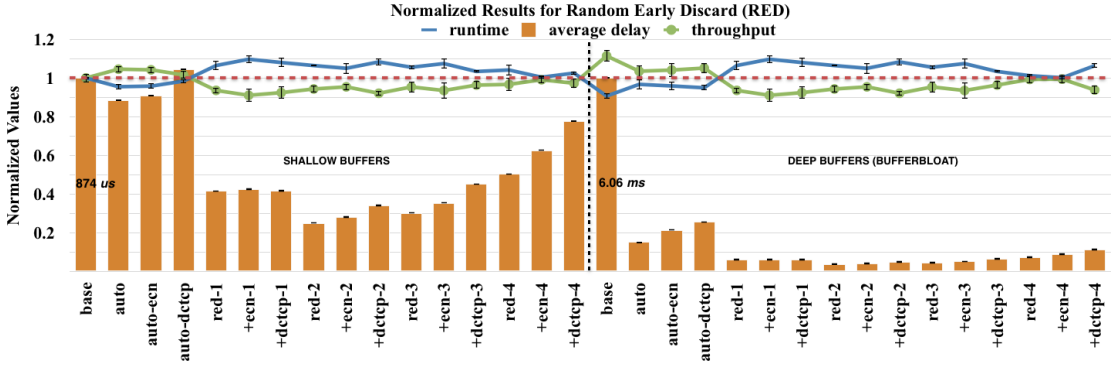


FIGURE 5.3: Normalized Results for Random Early Detection Queue

TABLE 5.3: Random Early Detection Settings

	Min Th.	Max Th.	Reference
Config 1	70	70	[40]
Config 2	25	51	[87]
Config 3	25	75	Proposal 1
Config 4	50	150	Proposal 2

clearly see that small values as 100 µs will drop the performance by unacceptable 30%. As the values increase, we can see the average delay also increasing. To compare with the other values proposed on literature we decided to use 1 ms as it was able to reduce the latency by 85% on bufferbloat scenario while still keeping a improvement of 5% on execution time.

When analyzing the fixed settings we can see that the first three configurations (config 1, config 2 and config 3) didn't perform well for RED, increasing the runtime in 10%. Configuration 4 was able to maintain the same execution time and still reduce latency by up to 90%.

5.3.2 Controlled Delay (CoDel)

For Controlled Delay queue we used the available implementation without any modification. CoDel is considered a parameterless queue, but the network administrator still has to provide two values, target and interval as described on Related Work (see Chapter 2.5). For our evaluation we considered values found on previous publications and we also proposed new values as well to tolerate a bigger delay on bufferbloat scenarios. The used values on our simulations can be found in Table 5.4. Still, we couldn't find many references specifically related to tune CoDel for data center networks. We can list the values found on Bufferbloat project [47] which recommends the use of 500 µs for target and 20 ms for interval. Another short paper [87] tested different range of values for CoDel recommending 300 µs for target and the much smaller 750 µs for interval. We also proposed 400 µs and 1 ms; and 800 µs and 1.5 ms for target and interval respectively as found in Table 5.4. Results can be found in Figure 5.4.

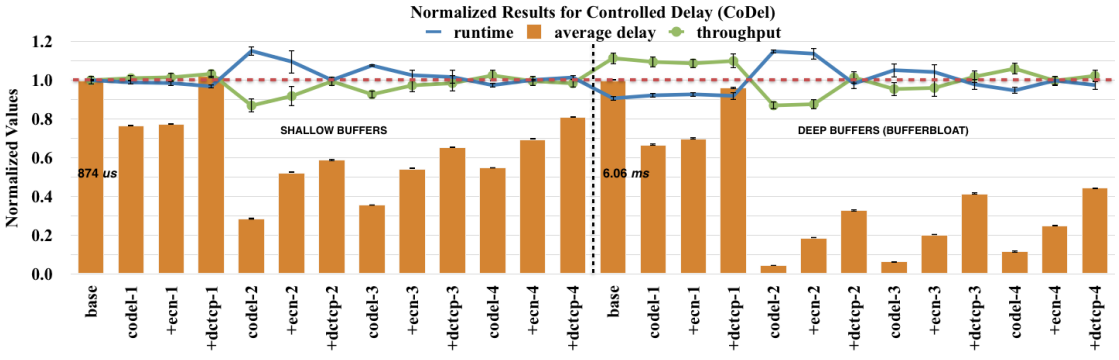


FIGURE 5.4: Normalized Results for Controlled Delay Queue

Starting by the results with shallow buffers, CoDel was able to reduce the average delay per packet by half using our second proposal (config 4) with virtually no loss on throughput. When using the settings recommended by the Bufferbloat project (config 1) the

TABLE 5.4: Controlled Delay Settings

	Target delay	Interval	Reference
Config 1	500 µs	20 ms	[47]
Config 2	300 µs	0.75 µs	[87]
Config 3	400 µs	1 ms	Proposal 1
Config 4	800 µs	1.5 ms	Proposal 2

delay dropped about 25% when combined with classical TCP. For all configurations, the smaller delay was achieved with standalone CoDel, followed by CoDel + ECN and at last, DCTCP showed the higher delay. On bufferbloat scenarios, we see that configurations that tolerate a higher delay also offer similar higher throughput. The settings recommended by Bufferbloat project were able to reduce the latency by 35% and still keep about 10% improvement on execution time. As the baseline for such scenarios has an average delay per packet of about 6 ms, for latency sensitive applications the configurations using CoDel stand alone were able to reduce latency to less than 700 µs with config 4 and even less than 400 µs using config 2. As a downside, specially configurations 2 and 3 were too aggressive reducing the congestion window and throughput was severely impacted. On the next subsection we compare the best values observed on CoDel and on RED.

5.3.3 CoDel x RED

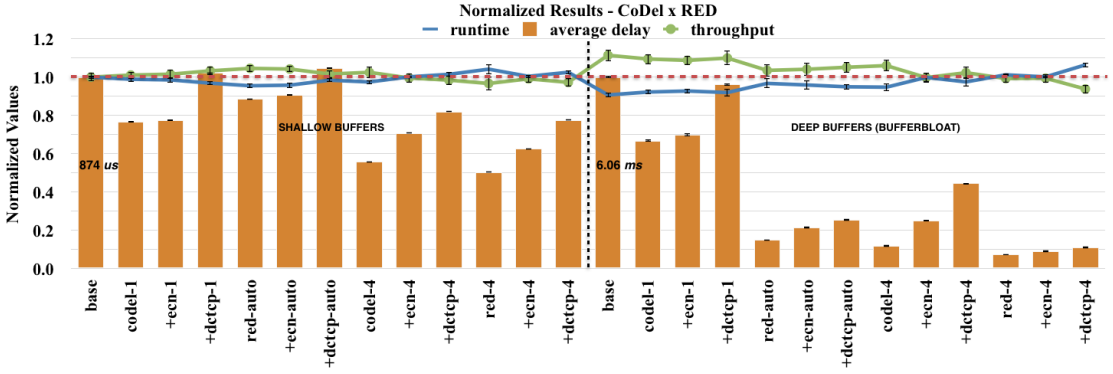


FIGURE 5.5: Normalized Results for CoDel x RED Queues

Figure 5.5 compares the best settings of each queue side-by-side. When comparing CoDel and RED side-by-side we selected the two best values of each. The first pair are the settings that offer the best throughput but do not have a considerable cut on latency. The last pair are the settings which offer the best reduction on delay while still

maintain some gain on throughput. We can see that even though CoDel and RED use values that are more or less at the same scale (800 µs vs. 1 ms), RED tend to achieve smaller latency than CoDel. On the other hand, CoDel was able to achieve the best runtime. One explanation for such difference is that RED is using the instant queue length, which turns out to be more responsive than the dynamic calculation performed by CoDel. Also, CoDel feature of marking packets on dequeuing, when combined with DCTCP, seems to be too conservative on reducing the congestion window and therefore does not reduce the latency as much as standalone CoDel or CoDel combined with ECN. By choosing other settings as 2 ms or 4 ms from RED auto configuration we would be able to match CoDel's performance but latency would be increased as well.

5.4 Discussion and Recommendations

As mentioned in the introduction, data center networks are starting to employ a new generation of switches with larger buffers, in order to accommodate bursty communications and deliver better application performance. A major portion of applications currently use TCP as the transmission protocol, so they will suffer from large packet latency, due to TCP's tendency to fully utilize the available buffering. Recent switches that implement AQM already support ECN, but until a good understanding of the impact of congestion control on real application performance has been reached, these features are likely to remain switched off, unnecessarily forgoing a feature that could significantly reduce latency.

This chapter contributes to the necessary understanding by analyzing the tradeoff in detail for MapReduce workloads, which are representative of modern big data applications in modern data centers.

When congestion happens at an AQM queue, if combined with ECN, it will tell the sender proactively. The naive assumption would be to think that packet loss is always destructive, but an equally naive assumption is to believe that something even more destructive would be the reduction in the size of the congestion window. As seen in the results section, CoDel combined with DCTCP can be too conservative on reducing the congestion window. Therefore the delay is not reduced as much as it was on standalone CoDel or CoDel combined with ECN while the throughput remained about the same.

Surprisingly, with standalone RED or RED combined with ECN, we were able to considerably reduce the latency on bufferbloat scenarios by 85%, and still maintain performance gains from larger buffers within 5%. Using instantaneous queue length instead of the average queue length was extremely necessary to obtaining these results. It also

open discussions whether CoDel should have faster converging mode, once its interval, even smaller than 1 ms, seems to deliver much higher latency on similar bases as RED, specially when it is combined with DCTCP.

Finally we finish with recommendations for system administrators. We observed that DCTCP could not achieve the best cuts on latency. That can be explained because DCTCP had not been yet analyzed on scenarios with deep buffers. In deep buffer scenarios, the baseline for latency is considerably higher and the original recommendation as a minimum 65 of packets, which was presented in DCTCP's evaluation, does not suit for such scenarios. Also, as we mention previously, it may not perform well with just any AQM, as it was the case with CoDel. Therefore we recommend careful consideration before deploying DCTCP on bufferbloat scenarios.

In summary, our observations and recommendations are threefold:

1. Contrary to what could be expected, both RED and CoDel queues may perform better standalone than combined with ECN or DCTCP.
2. For high-speed networks, using instant queue length on RED turned it more responsive than CoDel and its dynamic evaluation.
3. DCTCP delivered the highest latency per set of configuration on both RED and CoDel, but it was not translated on considerable gains on throughput to justify its usage.

5.5 Conclusions

A new challenge for modern data centers is to reduce latency caused by large buffers in the network equipment. Specific efforts to reduce latency are related to the requirements of particular workloads. Such actions, however, should not be considered trivial, because the choice of congestion control has a significant effect on throughput and performance, and should not be adopted in practice until the effects on workload performance are well understood.

This chapter presented our new work investigating the effects of latency control mechanisms on a Hadoop cluster. We demonstrated how throughput and burst tolerance play important roles for such big data workloads. We evaluated the performance impact on execution time, throughput and latency, when using RED or CoDel, both combined with and without ECN and at last DCTCP as the transport protocol, and found that the MapReduce programming model is not sensitive to the latency but it is sensitive to

even small reductions in the network throughput. We demonstrated that in some cases with poorly-chosen AQM configuration the execution time increases by an unacceptable 20%. We also identified good AQM configurations that were able to maintain Hadoop execution time gains from larger buffer to within 5%, while reducing packet latency caused by bufferbloat by 85%.

Therefore we suggest that cluster administrators carefully consider whether to adopt such techniques, depending on the cluster design and its utilization. For a heterogeneous cluster also running latency-sensitive workloads concurrently, it is important to reduce latency and buffer occupancy caused by larger buffers. In contrast, clusters used only for batch big data processing can neglect the latency, and benefit from the lowest execution time.

Chapter 6

High Throughput and Low Latency on Hadoop Clusters

This chapter presents a solution for the difficulty in configuration of Active Queue Management in Hadoop clusters.

6.1 Summary

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 [40], 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 [88].

As presented in the previous chapter, in certain settings however, recent studies have found that attempts to reduce latency often cause a degradation in throughput or performance [21, 29]. 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 [21, 29]. This chapter investigates why, and it proposes a solution.

Figure 6.1 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 chapter. This figure was generated using the methodology of Section 6.3.

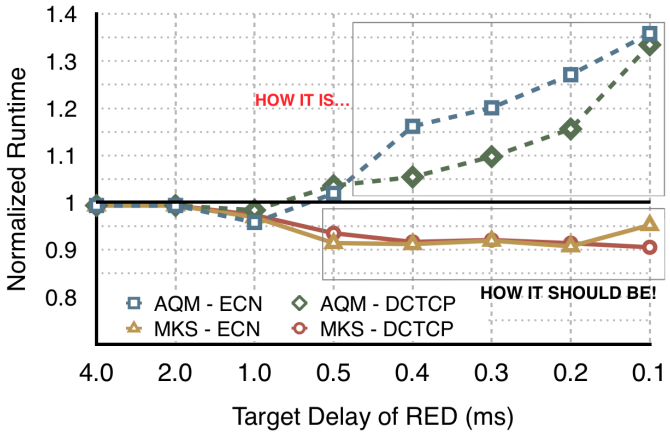


FIGURE 6.1: Hadoop job execution time affected by Active Queue Management

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 in some cases the shuffle time can even be responsible for more than 70% of the total runtime in the worst cases [84]. 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 also has evolved from a batch oriented workload to a more responsive and interactive 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 interactive solutions [21, 29]. For that, the network latency on current Hadoop clusters has to be decreased.

This work provides recommendations to network equipment manufacturers and network administrators on how to reduce network latency on Hadoop clusters without degrading performance. We expect 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. Aligned with the study presented on the previous chapter, we show the impact on Hadoop job execution time.

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 this chapter is organized as follows. Section II describes the problem and its solution. Section III describes our infrastructure and methodology and Section IV presents the evaluation and results. Based on these results, Section V distills the most important recommendations. Section VI compares our approach with related work. Finally, Section VII concludes the chapter.

6.2 The Problem and Motivation

Network transport protocols, such as TCP, traditionally signal congestion to the sender by dropping packets. This mechanism is simple, but it reduces throughput due to potential timeouts 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 (as explained below in Section 6.2.1). 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 [40]. 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)[42]. RED is an Active Queue Management (AQM) scheme 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 authors believed that this approach would be able to mimic the behavior of a marking scheme on ECN-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 support ECN and can be marked to indicate congestion, but in the same situation the non-ECT-capable packets would be dropped.

6.2.1 A deeper look at TCP packet marking

The main role of the network switch buffers is to absorb burstiness in packet arrivals, which 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 [53].

TCP connections will greedily use the available buffering on their network path. Therefore persistently full deep buffers can cause a problem known as Bufferbloat [26]. 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.

Latency increasing on Data Centers has become a major problem and with that, DCTCP gained much more attention. A recent study [88] 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 dropped. They even pointed to a possible problem with the opening of new connections. Since SYN packets are not ECT-capable, congestion could cause an excessive dropping of SYN packets, which would prevent new connections from being established.

Recent studies focusing on Hadoop clusters present experiments with ECN and DCTCP in an attempt to improve the network latency without degrading throughput or performance [21, 29]. In the latter study, presented in the previous chapter of this thesis, we were able to provide useful configurations, but fine-tuning the AQM queues was considered to be non-trivial.

On this study, 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 6.2 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.

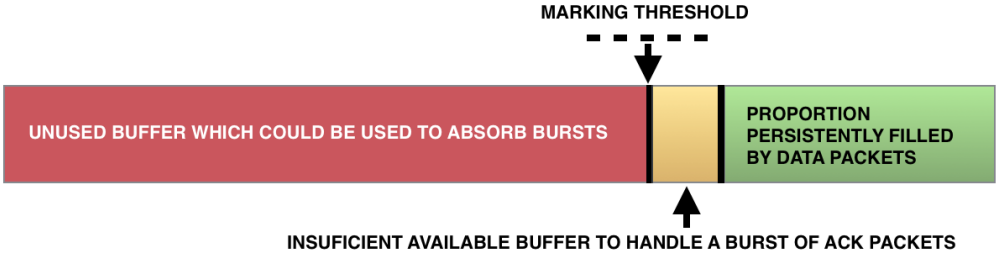


FIGURE 6.2: Typical snapshot of a network switch queue in a Hadoop cluster

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 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, and we investigate why this happens. Our simulations show that using ECN as congestion control actually works well with long lived TCP flows.

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. The major problem with trying to use proactive congestion control as ECN or DCTCP when data packets are marked by AQM queues is that packets that cannot be marked will be dropped. Dropping a class of packets while a distinguished set of packets will benefit from an only-marking-no-dropping approach to signalize congestion can degrade TCP performance, specifically on a framework with an all-to-all communication pattern as Hadoop.

This problem was not detected in the original evaluation of DCTCP, nor has it appeared in more recent experiments because the experimental testbed considered simple topologies with many senders and one receiver. Although such an approach recreates a network bottleneck, which allows to benchmark congestion for both TCP and DCTCP, such an evaluation does not involve data packets sharing the same egress port as ACKs. Another situation we verified is the reason why, on recent studies and evaluations, DCTCP outperformed TCP extended with ECN. Partition/Aggregation type of workloads have the

network traffic typically being consistently comprised of short query messages. For short messages, for which the TCP congestion window never grows significantly, a severe cut of the congestion window on the first indication of congestion can significantly decrease throughput.

Currently, it seems that TCP extended with ECN has already been discarded as an option for data centers. In its most recent evaluation [88], the authors considered ECT-capable flows as only DCTCP flows, considering that TCP and DCTCP traffic cannot share the same infrastructure. They did not evaluate TCP extended with ECN, because, as above, it seems to have been already discarded as a solution for data center networks. On the other hand, in this work, we present results using TCP extended with ECN, which outperforms DCTCP in a specific situation. We will refer to TCP extended with ECN as TCP–ECN, which also carries ECT-capable 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 [88] 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 on data packets 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.

6.2.2 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 in Table 6.1. As seen in Table 6.2, current AQM implementations only check for ECT(0) or 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 cluster throughput.

Table 6.1: ECN codepoints on TCP header

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

Table 6.2: 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

Our second proposal is to finally implement a true simple marking scheme on switches, independently of the buffer density per port. For shallow buffer devices 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, specially on commodity switches which offer shallow buffer density per port.

In the next section we present the experimental methodology, then followed by the results with the new proposals. We demonstrate 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 even improved, specially for commodity switches as long as any important packet which is not ECT-capable is allowed to be kept on the resilient buffer that remains available when using tight marking thresholds.

6.3 Simulation Environment and Workload Characterization

TABLE 6.3: Simulated Environment (based on Chapter 5)

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. 300 KB per port
	Expensive switches	2000 packets - max. 3 MB per port

The topology selected for this work was the leaf–spine architecture [67], based on a two-tier Clos architecture, were every lower-tier switch (leaf layer) is connected to each of the top-tier switches (spine layer) in a full-mesh topology as seen in Figure 3.1 and described in Chapter 3. Additionally, most part of the methodology described here is also replicated from last chapter.

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 [44]. It is also implemented by network equipment vendors including Cisco [45] and Juniper Networks [46]. RED uses configurable thresholds to decide when to mark packets if combined with ECN, and drops packets based on a probability that grows with the queue occupancy.

TABLE 6.4: 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*

First threshold is when the queue starts to mark or early dropping packets, both based on probability. Second threshold defines when a packet will be either always marked or

always dropped, with probability reaching 100%. Table 6.4 shows the used settings for the autoRED configuration. Since the max capacity of deep buffers was limited on 2000 packets, some values will not produce effect for 10 GbE. 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 never early dropping packets.

Controlled Delay is recommended by the Bufferbloat initiative [26, 47]. 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 6.5 shows the configured settings for CoDel queue.

Table 6.5: Controlled Delay (CoDel) Settings

Aggressiveness level	Target delay	Interval	Reference
1	500 µs	20 ms	[47]
2	800 µs	1.5 ms	[29]
3	400 µs	1 ms	[29]
4	300 µs	0.75 ms	[87]

We modified both AQM queues 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 both AQM queues to correctly mimic a true simple marking scheme. We could identify the problem related the the extra ACKs which are neither ECT-capable nor 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. 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, either if they are echoing a congestion experience with an ECE-bit or not.

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. Therefore the improvements on TCP throughput measured here can be translated to a production environment. The real gain on the MapReduce runtime will depend on how much is the extent of each workload regarding the proportion of computation and communication.

6.4 Results

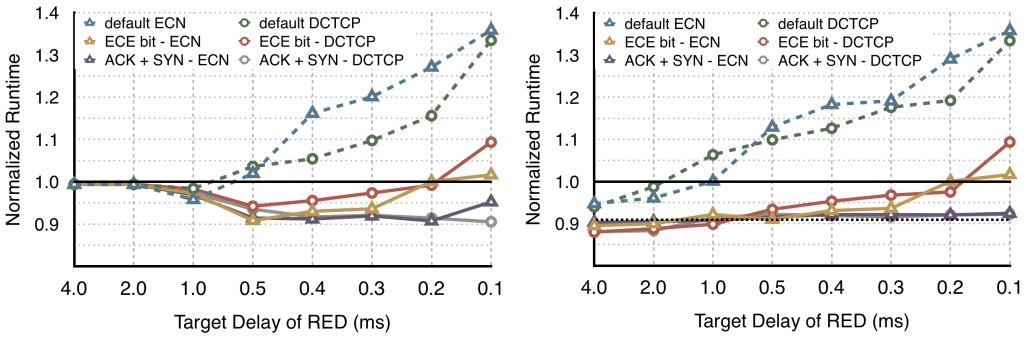
This section presents the quantitative results in terms of the runtime, throughput and latency for Hadoop using the methodology described in the previous section. Results are shown for both RED and CoDel, in order to demonstrate that the approach advocated in this chapter is independent of the precise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

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.

6.4.1 Random Early Detection (RED)

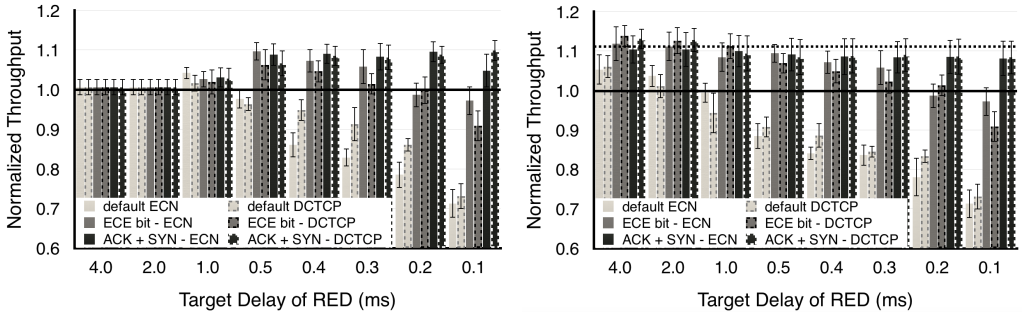
We start by presenting the effect of configuring the target delay of RED and how its different thresholds (see the previous section) affect Hadoop runtime for switches with shallow buffers. Figure 6.3a shows the runtime for shallow buffers and that for shallow buffers the best runtime is achieved either 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 6.4a 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% compared to the baseline, as confirmed on Figure 6.5a.



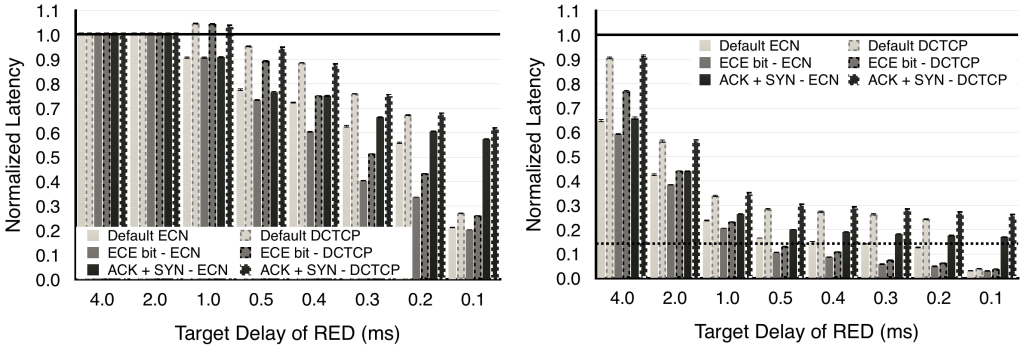
(A) Shallow Buffers (B) Deep Buffers

FIGURE 6.3: Hadoop Runtime - RED



(A) Shallow Buffers (B) Deep Buffers

FIGURE 6.4: Cluster Throughput - RED



(A) Shallow Buffers (B) Deep Buffers

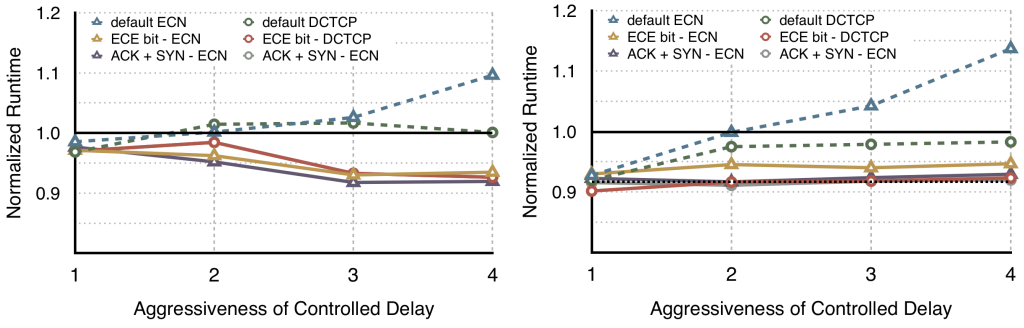
FIGURE 6.5: Network Latency - RED

For deep buffers, we start with Figure 6.4b. 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 6.5b, 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 6.3b 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.

6.4.2 Controlled Delay (CoDel)

Using a modified CoDel enabled us to verify that regarding how logic of the queue to decide when marking and dropping packets also influence on 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.

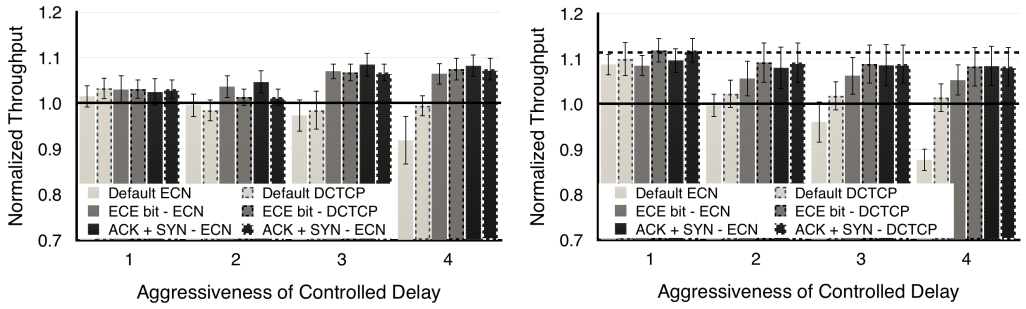
We start again by presenting the effect of delay control applied to CoDel as regarding the aggressiveness of the settings described on previous section. Figure 6.6a shows that ACK+SYN for ECN achieved he fastest Hadoop runtime, about 8% faster than the DropTail baseline. Cluster throughput was also stable as verified in Figure 6.7a, even used the most aggressive settings it remained above 5% improvements for both proposed solutions and all the flows. Analysing Figure 6.8a shows CoDel reduced its latency by half of what RED achieved, which translate on a network latency reduced by 40%.



(A) Shallow Buffers (B) Deep Buffers

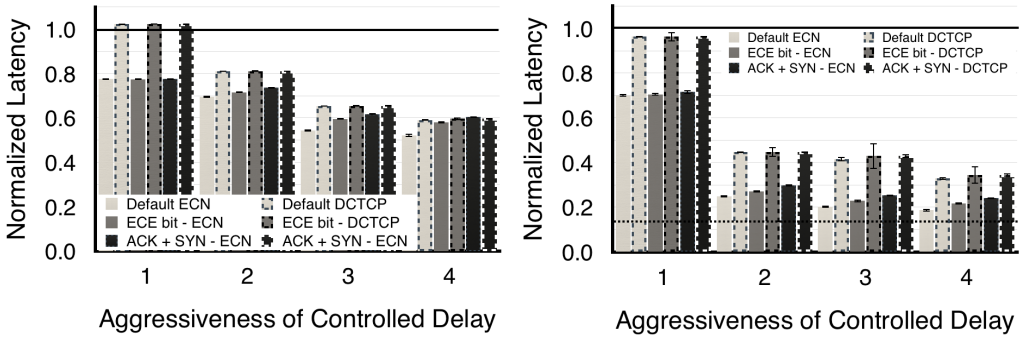
FIGURE 6.6: Hadoop Runtime - CoDel

As seen in Figure 6.6b, 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 6.7b. At last, the small reduction in throughput was compensated for by a reduction in network latency of about 80%,



(A) Shallow Buffers (B) Deep Buffers

FIGURE 6.7: Cluster Throughput - CoDel



(A) Shallow Buffers (B) Deep Buffers

FIGURE 6.8: Network Latency - CoDel

almost reaching the same network latency found on shallow buffer switches (pointed by the dashed line) in Figure 6.8b. Specially for CoDel, ECN solutions had much lower network latency than DCTCP solutions, which matches with results related by previous work (see related work).

6.4.3 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 miss-configured AQMs. Next section presents discussions and recommendations regarding our results.

6.5 Discussion and Recommendations

The results presented in this chapter show that Hadoop can tremendously benefit from a true simple marking scheme. These results are not exclusive to Hadoop, but are expected to be reproduced on 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 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 on a way that egress queues would never be shared between data and ACK packets can make the problem disappear. The cluster topology and workload organization plays an important part on reproducing the problem described here. Hence why previous work failed on reproducing or identifying it. On the other hand, MapReduce workloads were designed on a way where a big portion of the cluster, it not the whole of it, will be engaged on a fully distributed computation. To do so, a many-to-many communication between servers is required and the network will be consistently congested on such phase.

DCTCP has a strong premises which is to maintain latency low while delivering really high throughput, even higher than classical TCP itself. It was demonstrated on previous evaluation and these results still hold for the circumstances on how it was evaluated. A workload composed by short TCP flows will benefit from a more gentle cut on the congestion window. TCP takes one round trip to double its congestion window. 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 per Second), where it is basically composed by short-lived TCP flows such as queries messages (flows typically shorter than 1 MB). Since TCP-ECN cuts the congestion window by half, it can degrade throughput on such type of workloads.

For long lived flows, as more than 1 MB, on DCTCP evaluation itself, there was no virtual difference on performance when it was compared against TCP-ECN. We verified that for Hadoop, where the network will be constantly congested, TCP-ECN will benefit from a more aggressive cut on the congestion window, offering a lower latency comparing to 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 signalize congestion as soon as possible. As the buffer density per port is limited, signal the congestion before drops happen will be translated on TCP throughput gains. For deep buffers switches, using settings with moderated aggressiveness allow more data to be persistent on egress queues while they can still proactively signalize congestion to the senders, which finally translates to an improved TCP throughput. We now distill our most important recommendations.

Recommendations for equipment vendors: The limited configurability of existing switches egress queues which do not allow the network administrator to disable drops mean that if AQMs are combined with ECN flows, the best results on terms of throughput and network latency for the shuffle phase of a Hadoop infrastructure cannot be fully achieved. An operational mode where egress queues only mark ECT-capable packets using a simple marking threshold while avoid drops from the non-ECT-capable traffic will improve the performance of workloads as Hadoop and it could be achieved with a firmware update. It should be offered to commodity switches as well, even if the switch offers a buffer density per port lower than 1 MB, as our results demonstrate how it can tune performance of legacy 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 the egress queue limit and congestion control is fully performed by TCP endpoints.

Recommendations for network administrators: Until a true simple marking scheme 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 very high buffer density per port. Although this solution will perform equally great on terms of throughput and network latency as the results obtained on this work, it is unlikely to lead to a low cost solution.

As a solution for switches which do not offer the option to forward non-ECT-capable traffic, the network latency can be reduced to optimum level at the cost of performance loss on throughput. The performance of the standalone queue was out from the scope of this work as it was already covered by previous work, related on the following section. Therefore, for commodity or legacy switches which offer AQM and ECN features, as an immediate low latency solution, we recommend the utilization of the standalone AQM queue, completely disabling ECN from TCP flows and from the queue configuration. The next section covers the related work, which enrich our recommendations presented here.

6.6 Related Work

The original DCTCP paper [40] 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 [55] 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 [88] pointed to the same problem that happens on a saturated egress queue when trying to open new connections.

A recent and promising method to perform congestion control on TCP endpoints obtained better results than DCTCP for data center networks, by implementing a new

congestion control technique based on the logistic growth function [89]. To perform their evaluation, the authors used a simple marking scheme that starts to mark packets using a tight threshold, which was set to only one packet. The authors, following the same approach as in previous work, claim that RED could mimic it, which is why they considered that the method would not require any changes in hardware.

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. [21] 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 on 65 packets as recommended in DCTCP's original paper [40]. 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 [29], 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 [90]. 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 scheme

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.

6.7 Conclusions

In this chapter, 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 chapter are not exclusive but can also 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 chapter can help reduce Hadoop runtime and allow low-latency services to run concurrently on the same infrastructure.

Chapter 7

Energy Savings and Lower Latency Networks

This chapter analyses the impact of Packet Coalescing in the context of network latency. Then, combining Packet Coalescing with Active Queue Management, we investigate how to design and configure interconnects to provide the maximum energy savings without degrading cluster throughput performance or network latency.

7.1 Summary

An important challenge of modern data centres is to minimise energy consumption, a significant proportion of which is due to the network. Network energy savings are possible using Energy Efficient Ethernet (EEE) IEEE 802.3az, which is already supported by a large number of NICs and switches.

In the Chapter 4 we presented the first study from the impact of Energy Efficient Ethernet on MapReduce workloads [27]. Overall, we found that although substantial energy savings are available, the packet coalescing settings must be carefully configured to avoid a substantial loss in performance [27].

Meanwhile, network switches are steadily increasing their per-port buffer capacities. New SDRAM-based products are being launched with per-port buffer densities of up to ten times larger [25]. Large buffers increase throughput, but they can exacerbate the Bufferbloat problem [26], with network latencies reaching tens of milliseconds for certain classes of workloads.

We presented in Chapter 5 the first quantitatively evaluation from the control of network latency in Hadoop clusters, which was done using Active Queue Management (AQM) with Explicit Congestion Notification (ECN), and had minimal impact on Hadoop batch performance [29].

This chapter connects these two distinct efforts, by introducing a cluster design approach for reducing the interconnect energy consumption while also reducing network latency. As previously demonstrated, the Packet Coalescing settings must be carefully configured in order to avoid a substantial loss in performance [27]. Even so, the impact of the extra network latency incurred by packet coalescing, which increased the Bandwidth–Delay Product, had to be compensated with more buffering and TCP packets in-flight. The increase in latency was tolerable because of Hadoop's original batch-oriented design. In contrast, it is surprisingly difficult to effectively combine packet coalescing on the 10 GbE links with controlled latency, as implemented using ECN with AQM.

Our guidelines are especially targeted for workloads with long east–west flows inside the data centre, such as Apache Hadoop. Our findings are simple to implement and straightforward to understand. By considering our settings and configurations, vendors and cluster administrators can reduce interconnect energy consumption without adversely affecting network latency. We also wish to open discussion and promote research towards new solutions. We present experimental results in terms of interconnect energy consumption, cluster throughput and network latency. Finally, we show the impact on Hadoop job execution time.

In short, our main contributions are:

1. We analyse the impact of different buffer densities and Packet Coalescing settings on Hadoop network latency.
2. We align the Packet Coalescing technique with Active Queue Management to reduce network latency and identify how to extract the best from the combined techniques.
3. We evaluate the proposed solution in terms of interconnect energy consumption, 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 this chapter is organized as follows. Section II describes the motivation. Section III describes the infrastructure and methodology and Section IV presents the

evaluation and results. Based on these results, Section V distills the most important recommendations. Finally, Section VI concludes the chapter.

7.2 Motivation

This section summarizes the motivation for linking recent work towards better Hadoop interconnect energy consumption with attempts to reduce Hadoop cluster network latency.

7.2.1 Packet Coalescing

In Chapter 4 we showed that the MapReduce programming model is not sensitive to the overheads of Energy Efficient Ethernet, even when combining the Packet Coalescing technique, contradicting the general guidelines from vendors to disable EEE. The extra latency introduced by Packet Coalescing can be compensated with more buffering and TCP packets in-flight. On the perspective that Hadoop was designed to be thoughput-sensitive as a batch-oriented workload, such solution provides near ideal interconnect energy savings and virtually no loss on throughput and performance.

That chapter also demonstrated that for 1GbE links, EEE already does a great job by providing close to ideal energy proportionality, i.e. the sleep and wake operations introduce negligible energy and performance overheads. On the other hand, 10GbE links, which are responsible for nearly half of the interconnect energy consumption, provide good energy savings only if the operator enables Packet Coalescing. Moderate Packet Coalescing settings provide low energy savings, whereas aggressive Packet Coalescing settings, which deliver greater energy savings, increase the Bandwidth-Delay Product (BDP), so maximizing throughput requires a greater number of packets in flight and therefore more buffering.

7.2.2 Buffer density and Hadoop Network Latency

The steady increase in network switch per-port buffer capacities is related to the wish to minimize packet drops, which typically arise in two distinct situations. Firstly, in upper-layer devices, at the aggregation and core layers, when bursty traffic on multiple incoming links is redirected to the same outgoing port, the switch will have to queue the packets before transmitting them. Secondly, in the access layer; i.e. in the Top-of-Rack (ToR) switches, incoming traffic intended for server nodes may arrive on a link that has

higher bandwidth than the link to the server; e.g. packets arrive on an incoming 10GbE link, but they must be retransmitted on an outgoing 1GbE link.

Employing high density buffer per port on network equipments can lead to excessive network latency. Figure 7.1 illustrates how network latency is impacted by the buffer density per port. These results were generated using the methodology described on the next section. In particular, the shallow-buffer switches have 0.1 MB of buffering per port, whereas the deep-buffer switches have 1 MB per port.

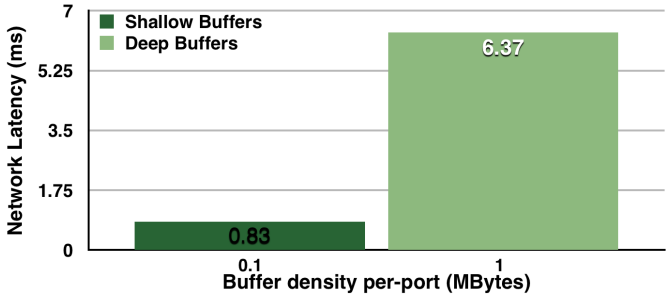


FIGURE 7.1: Buffer density impact on network latency on Hadoop

This problem is known as Bufferbloat [26], which is caused by TCP connections greedily using the available buffering on their network path. To reduce buffer utilization, Active Queue Management (AQM), which is already implemented in network switches, relies on two mechanisms to signal congestion: drop and mark. 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 timeouts and the need to re-transmit packets. Recent extensions, such as Explicit Congestion Notification (ECN) avoid these overheads by indicating imminent congestion by "marking" the packets (using specific bits in the IP header). Such congestion control based on proactive signaling was conceived with the premise that it was better to identify congestion before it is necessary to drop packets and wait for the sender to react [40].

Batch-oriented workloads, which have high throughput requirements can also benefit from a more proactive use of ECN combined with already available AQMs. As demonstrated in Chapter 5, network latency can be significantly reduced with minimum impact on cluster throughput. The proactive reaction to congestion translates to much lower buffering requirements needed to maximize throughput. With less buffering requirements, ECN with AQM can also reduce the number of packets needed in-flight. Combined with Packet Coalescing, however, as discussed in this chapter, both techniques must be carefully adjusted in order to realise the potential gains from both of them.

7.3 Methodology

7.3.1 Simulation Environment and Workload Characterization

Table 7.1: Simulated Environment

Category	Parameter	Value
Simulated hardware		
System	Number nodes	80
	Number racks	2
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:* 1
	Each spine switch	— *10GbE:* 1
Buffers	Shallow buffer per-port	200 packets - max. 300 KB per port
	Deep buffer per-port	2000 packets - max. 3 MB per port
Link power	10GbE	2.5 W
RED settings	Min. and Max. Thresholds	125 - 375
TCP buffer	Max. packet per connection	Unlimited

Table 7.1 shows the configuration, simulated enviroment and also the simulated workload. 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 4:1. 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 4.9 GB (random elements) with 79 mappers. Terasort is a popular batch benchmark commonly used to measure MapReduce performance on a Hadoop cluster. In order to be representative we simulated a batch workload where approximately 50% of time is due network communication, transferring data (shuffle phase) and the other 50% is spent on computation (map and reduce phases). It is consistent with a study of traces obtained at Facebook, which shows that most of the jobs were small and the shuffle phase represented in average 33% of the jobs execution time, which can even be more than 70% of the total runtime in the worst cases [84].

We assume the sleep and wake timings given in Table 7.2. The *ideal* case uses Energy Efficient Ethernet, but the sleep and wake transitions were considered to be both instantaneous and zero energy, providing an "ideal" point of comparison. In this case, the link is optimally controlled by simply entering low power mode as soon as it becomes inactive, providing perfect energy proportionality without affecting runtime. This result gives a lower bound on energy consumption. Finally, we also considered different values for packet coalescing. The right configuration is critical for maximum energy savings

and low performance overhead [34]. Christensen et al. [14] suggest using either a timer of 12 µs and a trigger of 10 packets or 120 µs and 100 packets. Another publication uses substantially different values [33], of 1 ms and 10 ms as timers, in both cases with 1,000 packets as trigger. As seen in Chapter 4, a timer of 10 ms can dramatically degrade the performance of MapReduce jobs. In that context where latency was not considered, 1 ms delivered close to ideal energy savings for 10GbE NICs. Additionally, we also consider an intermediate timer of 500 µs with a trigger of 500 packets, as seen in Table 7.3.

Table 7.2: EEE wake and sleep operations

Speed	Min. T_w (µs)	Min. T_s (µs)
1000Base-T	16.5	182
10GBase-T	4.48	2.88
Ideal	0	0

Table 7.3: Ethernet Specs

Label	Packet Coalescing settings	
	Holding time	Trigger
legacy eth	*No Energy Efficient Ethernet*	
ideal	*No overhead from sleep and wake operations*	
eee	*Energy Efficient Ethernet - no Packet Coalescing*	
12us10	12 µs	10 packets
120us100	120 µs	100 packets
500us500	500 µs	500 packets
1ms1000	1 ms	1000 packets

We considered RED as the selected AQM to mark packets with ECN feature configured on TCP senders. Implementations of Random Early Detection are found on Linux, Solaris, and FreeBSD [44]. It is also implemented by network equipment vendors including Cisco [45] and Juniper Networks [46]. RED uses configurable thresholds to decide when to mark packets if combined with ECN, and drops packets based on a probability that grows with the queue occupancy. First threshold is when the queue starts to mark or early dropping packets, both based on probability. Second threshold defines when a packet will be either always marked or always dropped, with probability reaching 100%.

Finally, the four performance metrics considered are: the *interconnect energy consumption* which is the energy consumed by 10GbE links, 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.

7.4 Results

This section applies the methodology described in the previous section, and presents the quantitative results in terms of interconnect energy consumption, the Hadoop runtime, cluster throughput and network latency.

7.4.1 Buffer density and Packet Coalescing on Hadoop

We start the analysis of our results by discussing Figure 7.2. The dashed area shows the extra latency introduced by Packet Coalescing, which for shallow buffers translates into an additional latency of 12% while for deep buffers the extra latency is about 6%. The baseline for the latency is the same presented in Figure 7.1 and discussed in Section 7.2. Since the latency found on deep buffers is much higher, the extra latency incurred by Packet Coalescing accounts for a lower (relative) impact on the normalized numbers.

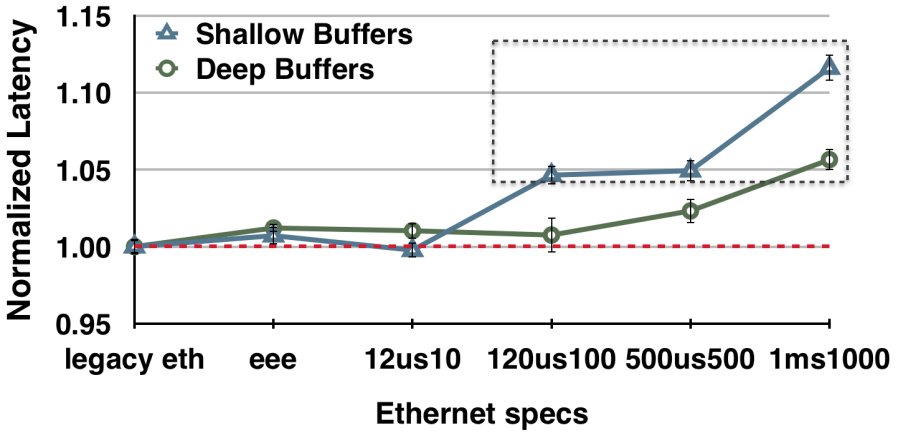


FIGURE 7.2: Packet coalescing impact on network latency considering different buffer sizes

Figure 7.3 presents the execution time and throughput results normalized to the shallow buffer baseline. We verify that more buffer density translates to higher throughput which translates to a faster runtime. We also verify that Packet Coalescing increases variability, but overall, the extra latency can be compensated by more buffering and packets in flight. Therefore, the gains obtained from deep buffers where maintained, even with the more aggressive setting for Packet Coalescing.

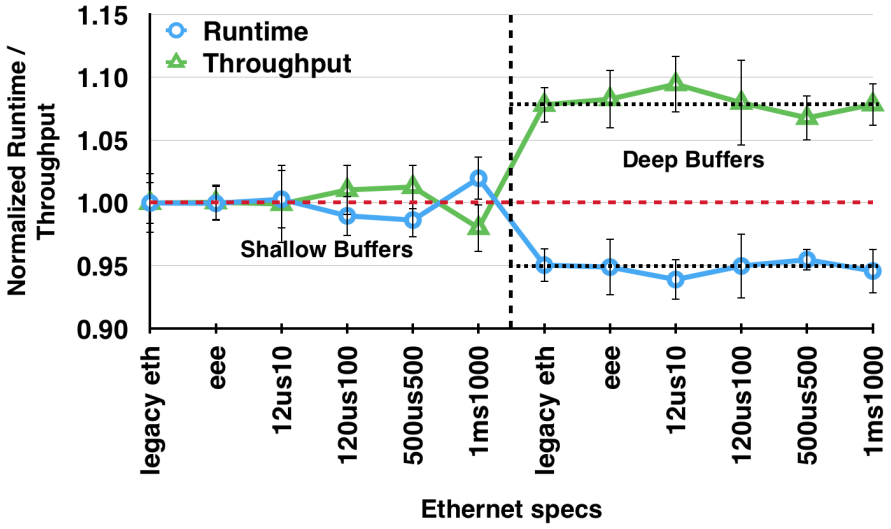


FIGURE 7.3: Packet coalescing impact on runtime and throughput considering different buffer sizes

Finally, we analyse Figure 7.4. The values are normalized to the ideal energy consumption, which means zero energy for sleep and wake operations. On our benchmark we verify that the 10GbE links consumed more than five times the energy consumption per NIC. We zoomed-in the bars to obtain a clearer comparison for the other settings. Energy Efficient Ethernet is able to significantly reduce the energy consumption but it is still almost 80% from ideal. It was therefore possible to obtain considerable gains with Packet Coalescing. The best gains were obtained using 1ms1000 with deep buffers, reaching near only 5% more energy than the ideal model, while 120us100 was near 10% from that and 500us500 in between these two settings.

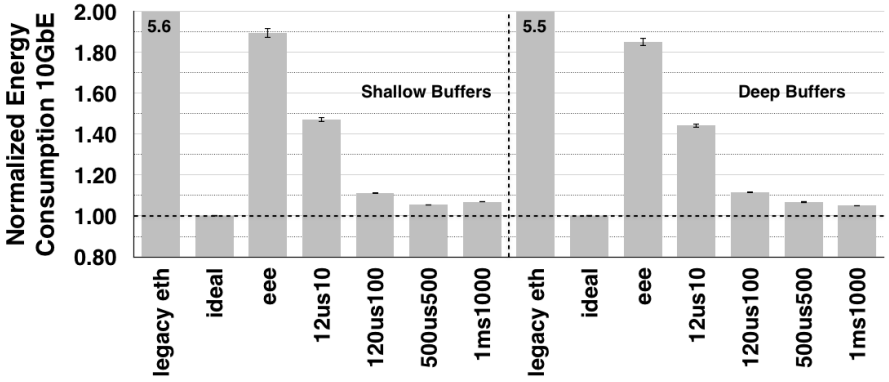


FIGURE 7.4: Packet Coalescing impact on energy consumption of 10GbE considering different buffer sizes

We move on with the next set of experiments, which consist of: enabling ECN on the TCP end-points, enabling ECN's marking feature on each RED egress buffer, and using the configuration described in Section 7.3. We expect to not only reduce network

latency but also maintain cluster throughput and specially maximize the energy savings for the 10GbE links, obtained with Packet Coalescing. Figure 7.5 shows the impact of congestion control on network latency for deep buffers, which will be the baseline for the results in the next set of results.

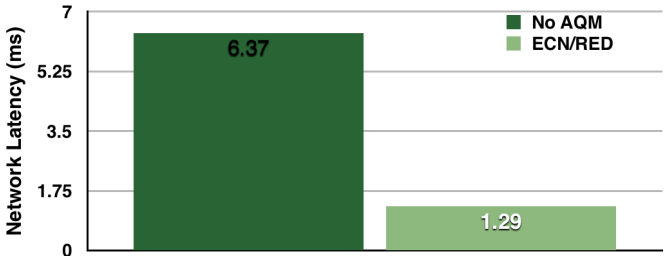


FIGURE 7.5: Congestion control impact on network latency on deep buffers

7.4.2 Combining Packet Coalescing with ECN/AQM/RED

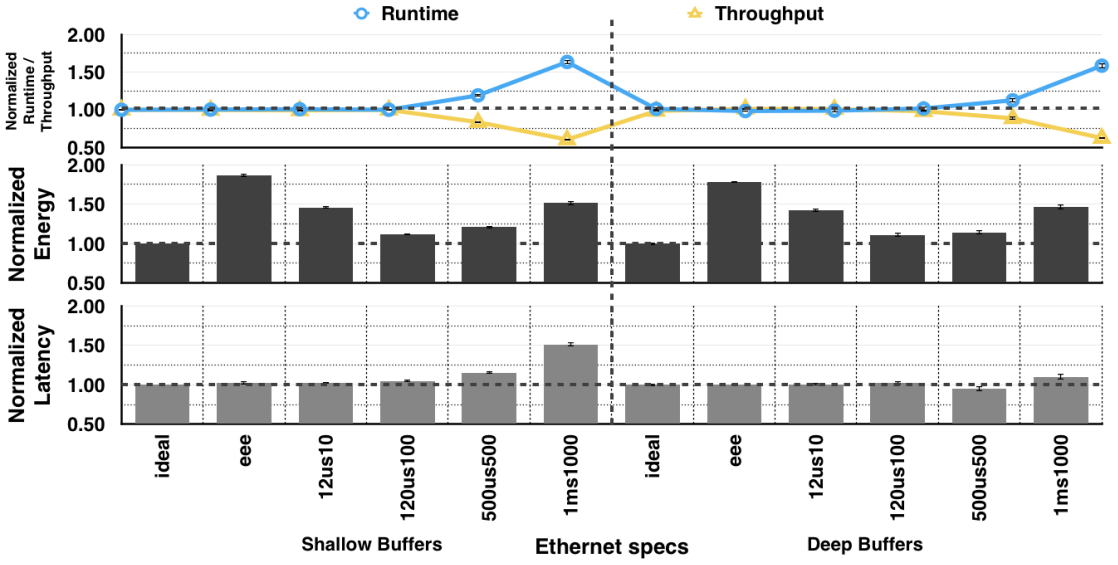


FIGURE 7.6: Runtime, Latency, Throughput and Energy values for Packet Coalescing combined with RED and ECN

Figure 7.6 brings the detailed results considering our four metrics described on Section 7.3. Starting by Hadoop performance, we can see that the two more agressive settings decreased cluster throughput, which also impacted on a larger execution time. While 500us500 impacted on approximately 25% performance degradation, the more aggressive Packet Coalescing 1ms1000 inflicted a loss on performance that was higher than 50%. Proactive congestion control limits the buffering required to keep throughput performance compared to the baseline. The larger latency inflicted by larger coalescing times increases the Bandwidth-Delay Product (BDP), so maximizing throughput

requires a greater number of packets in flight and therefore more buffering, which is restrained by the employed proactive congestion control.

Analyzing the energy consumption, we also see that the two more aggressive Packet Coalescing settings no longer provide the best energy savings. The increase on execution time was responsible for losing all the greatness on energy savings we verified when Packet Coalescing is used stand-alone.

Finally, we verified an overall reduction on network latency as expected. Considering the Packet Coalescing settings, the network latency suffered an increment of almost 50% for shallow buffers when using the more aggressive Packet Coalescing settings. For deep buffers the extra latency was responsible for a smaller increment of 10%. Still, when combining our metrics together, we can no longer verify any benefit of utilizing 1ms1000 or even 500us500.

7.4.3 Summary of Results

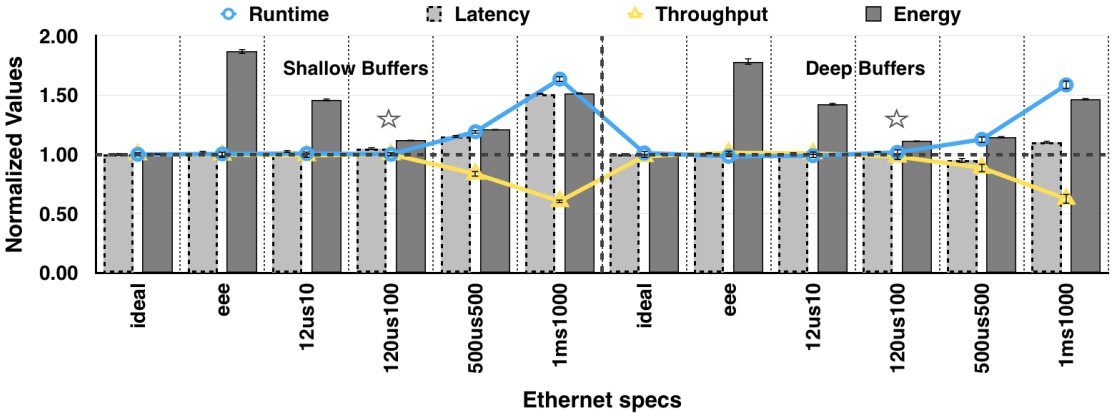


FIGURE 7.7: Runtime, Latency, Throughput and Energy values for Packet Coalescing combined with RED and ECN

Figure 7.7 brings the big picture with all the four metrics combined. We included a star to highlight the best combination which includes latency compared to the baseline, energy near 10% the ideal model and finally no verified loss on performance and cluster throughput. Considering 120us100 packets, we demonstrate it is possible to achieve a much lower network latency while still maintaining the interconnect energy savings obtained by utilizing Packet Coalescing.

7.5 Discussion and Recommendations

The results presented in this chapter show that Hadoop clusters can significantly benefit from packet coalescing combined with proactive congestion control mechanisms. The results presented here are not exclusive to Hadoop, but are expected to be reproduced on other types of workload that present the following three characteristics:

- East–west traffic patterns and long-lived TCP flows with bursty communication.
- TCP flows configured to use ECN, either as TCP–ECN or DCTCP, and switches configured to mark packets.
- NICs and switches that implement Energy Efficient Ethernet with the option to coalesce packets.

We now distill our most important recommendations.

Recommendations for equipment vendors: Due to the potential energy savings, equipment vendors should consider implementing Packet Coalescing in their NICs and switches. It is important, however, to offer some reconfigurability, since depending on the workload, more aggressive settings may be desired while for other classes of workloads, less aggressive settings may already provide good energy savings. As a pratical example, some vendors implement coalescing on the receiver side, which is typically named Interrupt Coalescing. Interrupt Coalescing is a related feature that is typically implemented with the possibility for configuration by the network administrator. Our simulations with packet coalescing for EEE explicitly implements it on both sender and receiver sides, which can be useful at the servers layer (NIC's can aggregate more packets from different flows before transmitting it), and also on switches layer where many flows from different ingress ports are being redirected to the same egress (output) port. Another difference to Interrupt Coalescing is that it is typically recommended to use lower values as extra latency could decrease the performance of latency-sensitive workloads [83]. Currently available Energy Efficient Ethernet NICs still don't have option to configure the packet coalescing settings. We believe that as the Interrupt Coalescing, Packet Coalescing will also be implemented as a configurable feature by network manufactures in the near future.

Recommendations for network administrators: Energy Efficient Ethernet NICs do not currently offer the possibility to adjust the configuration of the Packet Coalescing settings. We argue that EEE NICs should in future offer such flexibility. If this does

finally happen, we recommend this work as a guideline to obtain maximum energy savings without degrading Hadoop performance or network latency. For batch workloads where latency is not a concern, we recommend the more aggressive settings which have its extra latency compensated with more buffering and packets in-flight. If reducing network latency is the major concern, we recommend the utilization of some congestion control mechanism as ECN or DCTCP without discarding the utilization of Packet Coalescing. We demonstrated it is feasible and possible to combine both techniques with no loss on the four metrics considered on this work.

7.6 Conclusions

This chapter has presented a novel analysis of the impact of Energy Efficient Ethernet (EEE) and Packet Coalescing on network latency for Hadoop Clusters. Combining Packet Coalescing with ECN plus AQM, which is already found on network switches, delivers network latencies comparable to that found for ideal on/off links, without EEE's sleep and wake overheads. We were also able to reduce the energy consumption from 10GbE links by 70%, compared to default EEE, which does not use Packet Coalescing.

In summary, we suggest that equipment vendors implement Packet Coalescing and also provide the ability for operators to modify the Packet Coalescing configuration settings. In turn, we suggest that network administrators use the recommendations in this chapter together with knowledge of their application's network latency requirements. Doing so will provide the best possible energy savings without compromising performance or latency requirements.

Chapter 8

Conclusion and Future Work

Big Data workloads have emerged in the recent past and have undergone a constant evolution to fit application requirements which aim to satisfy users needs from a universe of currently available online services. As we previously showed a major portion, corresponding to more than 3/4 of total global traffic, resides inside the Data Center. This traffic has all the sources and destinations being the servers themselves, within the Data Center infrastructure and the Big Data workloads are considered responsible for such generated traffic.

The Hadoop ecosystem, which has been adopted by the biggest Internet companies these days is one of the most famous Big Data frameworks, responsible for processing huge data sets on distributed clusters. Such development has been supported by the same companies that require its utilization and with such enormous support the platform has evolved from a batch-oriented processing workload to a more responsive platform providing real time and online processing services.

As the platform evolves, investments and improvements in infrastructure are made necessary. First, it is desired to keep up with the generated traffic while also reducing its energy footprint; and second to improve responsiveness from the infrastructure itself, allowing designers and users to benefit from new classes of applications.

The network is commonly described as the bottleneck present of Hadoop infrastructure. Reasons relate to the fact that Data Centers face a distinct class of problems regarding the Local Area Network traffic. For example, most Data Center applications are written using TCP as the transport protocol. TCP was originally designed for Wide Area Networks. Therefore some problems that would never happen on a WAN arise on LANs, mainly related to the difference of latency and to the distributed nature of Data Center networks. While WANs will have their traffic latency measured in hundreds

of *milliseconds*, LANs will operate on a much lower magnitude, typically hundreds of *microseconds*.

Before applying Energy Efficient Ethernet to reduce interconnect energy consumption, we took into consideration the phenomenona described in Chapter 2. We did the same when we applied AQM and congestion control (ECN) to improve performance and reduce network latency. Yet, the task of dealing with each of these major challenges was rewarded with other several minor problems that we had to solve, which after being tackled will help to improve current generation of interconnects for clusters running Big Data frameworks. At this point, we believe the problems solved in this thesis can help to improve not only Hadoop clusters but also any Big Data workload that fits the problem and characterization we described throughout the development of this thesis.

Specifically mentioning the energy problem, we concluded that for Hadoop clusters, state-of-the-art Energy Efficient Ethernet can be drastically improved if it starts to support Packet Coalescing. Although EEE can already be used and contrary to what we expected it can provide virtually no loss in performance, the Packet Coalescing technique should also be implemented, offering some option for reconfigurability to the network administrators, as depending on the workload and cluster configuration, it can incur on extra latency. Yet, for aggregation links, it could translate into an energy consumption reduced by almost 80%.

Considering the performance problem, we achieved the understanding as to why AQMs are not easy to configure on such type of workloads and even in some cases it can decrease throughput performance by unacceptable 20%. We proposed two solutions to modify how switch egress queues handle ECN marked packets. The greatest benefit would be for shallow buffer switches which can offer great benefit to cluster throughput with its performance increased by 10% while also reducing latency by 40%. On deep buffer switches, our proposed solution can reduce network latency by 85% while maintaining virtually the same performance for cluster throughput. At last, we verified that our solution offer robustness configuration, while the state-of-the-art requires a big effort to achieve similar, yet inferior performance.

We conclude this chapter presenting what in our vision can be carried out on future work to derive from our thesis.

8.1 Future work

Regarding **cluster design and topology**, during the development of our thesis we followed the guidelines provided by the two major companies that offer custom solutions

and deployments of Hadoop clusters. Cloudera and Hortonworks are today the two biggest players in number of deployments so we believe our virtual clusters were carefully designed taking in account their considerations so we could obtain more realistic figures for the results presented in this thesis. Mid-sized Haddop clusters are recommended to be organized on a Leaf-Spine topology for its focus on east-west traffic among servers of the same cluster. Yet, the Leaf-Spine topology is indeed organized in a two-tier clos architecture. It can "scale out" to a fairly large numbers of servers by adding more switches although it also brings a concern to the amount of cables and the cost of the network equipment required, once each leaf switch must be connected to each spine switch [69]. We believe the results seen on this work to remain similar when deployed on a cluster organized on a traditional hierarchical fat-tree with three layer network, which is more suitable for north-south traffic but similarly to the Leaf-Spine topology, is also categorized as a clos architecture.

Regarding **hardware specifications and configurations**, Hadoop is still targeting commodity hardware and Hortonworks recommends to not spend more than 20% of the available budget on network equipment including NICs from the servers [75]. Therefore, the starting point for any Hadoop cluster is the configuration of 1GbE on servers and Top of Rack switches, and 10GbE on the aggregation level. This was a solid configuration in 2014 when we started this work and it is still a starting point for small clusters today [91] [75]. Yet, the industry is starting to push faster NICs to the servers and aggregation switches. As a higher performance solution, 10GbE NICs are being considered for servers while 40GbE NICs are being considered for the aggregation level.

We believe that, as a future work to be carried out from this thesis, one could follow the industry trend of pushing for higher performance network equipments. Considering that, it would be of a great value to reproduce our experiments using these faster NICs. Applying 10/40GbE NICs instead of 1/10GbE would require new analysis on Energy Efficient Ethernet as the energy profile might change. Also, since there is a big improvement on the network capabilities which leads to expected transmitting packets on a faster pace, it could lead to different buffer utilization profiles and therefore, some new findings might be obtained from Packet Coalescing and AQMs. Finally, we believe that if the industry follows our suggestions to implement Packet Coalescing in EEE NICs and also implements a true marking scheme on network switches, an evaluation on real hardware should be performed to confirm our findings from this work.

Regarding **performance improvements**, Luiz Andre Barroso, distinguished Google fellow researcher recently shared his view from two of the three most important problems in the Data Center to achieve big amount of data processed in little amount of time ("big data, little time"): Energy Proportionality and Tail latency [92]. Our work can be used

to achieve energy proportionality for Hadoop cluster networks and we consider to have given the first step towards the direction of reducing the tail latency as we successfully reduced the average network latency found on these clusters from a few *milliseconds* to hundreds of *microseconds*. The techniques analysed on this work focused on reducing the average network latency [86], therefore we believe that reducing tail latency will require a different methodology to implement techniques not covered throughout the development of this thesis.

Regarding **other frameworks, workloads and benchmarks**, we believe the results from this work can be successfully reproduced in workloads that follow similar characterization as we presented as discussions on the previous chapters. As a framework that could benefit from this work, Apache Spark executes in-memory computations to eliminate disk spills found on MapReduce framework which therefore, reduces the data processing time and makes this framework more suitable for real time data analytics [93]. As a matter of fact, for non batch-oriented benchmarks, Spark can be from ten to a hundred times faster than MapReduce. Deploying Energy Efficient Ethernet on these clusters could likely help with achieving better energy proportionality on network link although packet coalescing may not be suitable to be deployed on a Spark infrastructure, as it could increase the network latency of real time applications. On the other hand, with the purpose of reducing network latency, the proactive congestion control techniques presented on this thesis could nicely fit this real time latency-sensitive framework.

Another benchmark to be studied on future work, Graph500 is a benchmark that is typically run in HPC clusters to rank these computing systems and their capacity to run Big Data applications [94]. HPC applications are typically written using Message Passing Interface (MPI) library, which under the hood can be implemented using either Stream Control Transmission Protocol (SCTP) or TCP as the transport protocol [95]. Both SCTP and TCP support ECN [96], so proactive congestion control could also improve latency on these clusters when running Big Data applications. On the other hand, these clusters also run heterogenous class of applications with low latency restringments and Energy Efficient Ethernet was found to impact application performance, even without packet coalescing due to collective dependent aggregation messages from MPI parallel programming paradigm [17]. These suggestions conclude our thoughts for future work. We conclude this thesis with the publication list related to the contributions obtained from this work.

8.2 List of Publications

- **[LCN15] Renan Fischer e Silva** and Paul M. Carpenter, *Exploring Interconnect Energy Savings Under East-West Traffic Pattern of MapReduce Clusters.* on the 40th IEEE Conference on Local Computer Networks 2015 [27].
  🏆Nominated for The Best Paper Award (top 3 best papers).
- **[LCN16] Renan Fischer e Silva** and Paul M. Carpenter, *Controlling Network Latency in Mixed Hadoop Clusters: Do We Need Active Queue Management?* on the 41st IEEE Conference on Local Computer Networks 2016 [29].
- **[CLUSTER17] Renan Fischer e Silva** and Paul M. Carpenter, *High Throughput and Low Latency on Hadoop Clusters using Explicit Congestion Notification: The Untold Truth* on the 19th IEEE International Conference on Cluster Computing 2017 [30].
- **[LCN17] Renan Fischer e Silva** and Paul M. Carpenter, *Interconnect Energy Savings and Lower Latency Networks in Hadoop Clusters: The Missing Link* on the 42nd IEEE Conference on Local Computer Networks 2017 [31].
- **[Transactions on Networking] Renan Fischer e Silva** and Paul M. Carpenter, *Energy Efficient Ethernet on MapReduce Clusters: Packet Coalescing To Improve 10GbE Links* on the IEEE/ACM Transactions on Networking, October 2017 [28].
- **[- 2018] Renan Fischer e Silva** and Paul M. Carpenter, *TCP Proactive Congestion Control Revamped: the Marking Threshold*, under review.

Bibliography

[1] "5 Web Technology Predictions for 2017." https://www.sencha.com/blog/5-web-technology-predictions-for-2017/. Accessed: 2018-02-27.

[2] Cisco Systems, Inc, "Cisco Global Cloud Index: Forecast and Methodology, 2013–2018," tech. rep., 2014.

[3] Cisco Systems, Inc, "Cisco global cloud index: Forecast and methodology, 2015–2020," tech. rep., 2016.

[4] S. Vaillancourt, "Scalable data center designs for canadian small & medium size business," tech. rep., 2014.

[5] "The history of hadoop - by marko bonaci." https://medium.com/@markobonaci/the-history-of-hadoop-68984a11704/. Accessed: 2018-02-27.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, (New York, NY, USA), pp. 29–43, ACM, 2003.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation*, OSDI'04, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2004.

[8] W. Si, J. Taheri, and A. Zomaya, "A distributed energy saving approach for ethernet switches in data centers," in *Proceedings of the 2012 37th Conference on Local Computer Networks (LCN 2012)*, LCN '12, (Washington, DC, USA), pp. 505–512, IEEE Computer Society, 2012.

[9] R. Brown *et al.*, "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.

[10] C. L. Belady, "In the data center, power and cooling costs more than the it equipment it supports." https://www.electronics-cooling.com/, 2007. Accessed: 2018-02-27.

[11] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, (New York, NY, USA), pp. 338–347, ACM, 2010.

[12] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[13] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy aware network operations," in *INFOCOM Workshops 2009*, pp. 1–6, IEEE, April 2009.

[14] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. Maestro, "IEEE 802.3az: the road to energy efficient ethernet," *Communications Magazine, IEEE*, vol. 48, pp. 50–56, November 2010.

[15] "Broadcom at interop: Energy efficient ethernet is good for the planet." https://www.broadcom.com/blog/broadcom-at-interop-energy-efficient-ethernet-is-good-for-the-p. Accessed: 2018-02-27.

[16] A. De La Oliva, T. R. V. Hernández, J. C. Guerri, J. A. Hernández, and P. Reviriego, "Performance analysis of energy efficient ethernet on video streaming servers," *Computer Networks*, vol. 57, no. 3, pp. 599–608, 2013.

[17] K. Saravanan, P. Carpenter, and A. Ramirez, "Power/performance evaluation of energy efficient ethernet (eee) for high performance computing," in *2013 International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 205–214, IEEE, April 2013.

[18] I. Cisco Systems, "IEEE 802.3az energy efficient ethernet: Build greener networks," tech. rep., 2011.

[19] Yamaha, "Disabling energy efficient ethernet (eee)." http://www.yamahaproaudio.com/global/en/training_support/selftraining/dante_guide/chapter2/02_eee/. Accessed: 2018-02-27.

[20] Dell, "Resolving issues with energy efficient ethernet (eee) or green ethernet." http://www.dell.com/support/Article/us/en/19/421774/EN. Accessed: 2018-02-27.

[21] 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), pp. 1–14, USENIX Association, 2015.

[22] G. Mone, "Beyond hadoop," *Commun. ACM*, vol. 56, pp. 22–24, Jan. 2013.

[23] "MapR Takes Road Less Traveled to Big Data." https://davidmenninger.ventanaresearch.com/mapr-takes-road-less-traveled-to-big-data-1. Accessed: 2018-02-27.

[24] A. Bechtolsheim, L. Dale, H. Holbrook, and A. Li, "Why Big Data Needs Big Buffer Switches. Arista White Paper," tech. rep., 2011.

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

[26] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, pp. 40:40–40:54, Nov. 2011.

[27] R. F. e Silva and P. M. Carpenter, "Exploring interconnect energy savings under east-west traffic pattern of mapreduce clusters," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pp. 10–18, Oct 2015.

[28] 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. 25, pp. 2731–2742, Oct 2017.

[29] 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)*, pp. 415–423, Nov 2016.

[30] R. F. e. Silva and 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)*, pp. 349–353, Sept 2017.

[31] R. F. E. Silva and P. M. Carpenter, "Interconnect energy savings and lower latency networks in hadoop clusters: The missing link," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 514–517, Oct 2017.

[32] T. I. of Electrical and I. Electronics Engineers, "IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 3: Csma/cd access method and physical layer specifications amendment 5: Media access control parameters, physical layers, and management parameters for energy-efficient ethernet," *IEEE Std 802.3az-2010 (Amendment to IEEE Std 802.3-2008)*, pp. 1–302, Oct 2010.

[33] P. Reviriego, J. Maestro, D. Larrabeiti, and D. Larrabeiti, "Burst transmission for energy-efficient ethernet," *Internet Computing, IEEE*, vol. 14, pp. 50–57, July 2010.

[34] S. Herreriá-Alonso, M. Rodríguez-Pérez, M. Fernández-Veiga, and C. López-Garcıá, "Optimal configuration of energy-efficient ethernet," *Computer Networks*, vol. 56, no. 10, pp. 2456 – 2467, 2012. Green communication networks.

[35] The Apache Software Foundation, "Apache Hadoop Project." http://hadoop.apache.org. Accessed: 2018-02-27.

[36] U. o. S. C. Information Sciences Institute, "Transmission control protocol," 1981.

[37] M. Ghobadi and Y. Ganjali, *TCP Adaptation Framework in Data Centers*. University of Toronto, 2014.

[38] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP incast throughput collapse in datacenter networks," in *Proceedings of the 1st Workshop on Research on Enterprise Networking*, WREN '09, (New York, NY, USA), pp. 73–82, ACM, 2009.

[39] P. Rygielski, S. Kounev, and S. Zschaler, "Model-based throughput prediction in data center networks," in *2013 International Workshop on Measurements and Networking Proceedings (M&N)*, pp. 167–172, IEEE, Oct 2013.

[40] 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*, SIGCOMM '10, (New York, NY, USA), pp. 63–74, ACM, 2010.

[41] P. Prakash, A. Dixit, Y. C. Hu, and R. Kompella, "The TCP outcast problem: Exposing unfairness in data center networks," in *Proceedings of the 9th Conference on Networked Systems Design and Implementation*, NSDI'12, (Berkeley, CA, USA), pp. 30–30, USENIX Association, 2012.

[42] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug 1993.

[43] K. Nichols and V. Jacobson, "Controlling queue delay," *Queue*, vol. 10, pp. 20:20–20:34, May 2012.

[44] "References on RED (Random Early Detection) Queue Management." http://www.icir.org/floyd/red.html. Accessed: 2018-02-27.

[45] "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: 2018-02-27.

[46] "RED Congestion Control." https://www.juniper.net/techpubs/en_US/junos12.2/topics/concept/random-early-detection-congestion-control-overview.html. Accessed: 2018-02-27.

[47] "Technical introduction to bufferbloat." https://www.bufferbloat.net/projects/bloat/wiki/. Accessed: 2018-02-27.

[48] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, "A performance perspective on energy efficient hpc links," in *Proceedings of the 28th International Conference on Supercomputing*, ICS '14, (New York, NY, USA), pp. 313–322, ACM, 2014.

[49] B. Dickov, M. Pericas, P. Carpenter, N. Navarro, and E. Ayguade, "Analyzing performance improvements and energy savings in infiniband architecture using network compression," in *2014 26th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 73–80, IEEE, Oct 2014.

[50] B. Dickov, M. Pericas, P. Carpenter, N. Navarro, and E. Ayguade, "Software-managed power reduction in infiniband links," in *2014 43rd International Conference on Parallel Processing (ICPP)*, pp. 311–320, IEEE, Sept 2014.

[51] B. Dickov, P. Carpenter, M. Pericas, and E. Ayguade, "Self-tuned software-managed energy reduction in infiniband links," in *ICPADS 2015*, December 2015.

[52] "Apache Myriad: Deploy Apache YARN Applications Using Apache Mesos." http://myriad.incubator.apache.org/. Accessed: 2018-02-27.

[53] Cisco Systems, Inc, "Big Data in the Enterprise - Network Design Considerations White Paper," tech. rep., 2011.

[54] K. Nichols and V. Jacobson, "Controlled delay active queue management," 2016.

[55] 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*, CoNEXT '12, (New York, NY, USA), pp. 25–36, ACM, 2012.

[56] N. Kuhn, E. Lochin, and O. Mehani, "Revisiting Old Friends: Is CoDel Really Achieving What RED Cannot?," in *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS '14, (New York, NY, USA), pp. 3–8, ACM, 2014.

[57] Y. Chen, R. Griffit, D. Zats, and R. H. Katz, "Understanding TCP Incast and Its Implications for Big Data Workloads," No. UCB/EECS-2012-40, Apr 2012.

[58] "Network Simulator NS-2." http://www.isi.edu/nsnam/ns. Accessed: 2018-02-27.

[59] P. Reviriego, K. Christensen, A. Sánchez-Macián, and J. Maestro, "Using coordinated transmission with energy efficient ethernet," in *NETWORKING 2011* (J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, eds.), vol. 6640 of *Lecture Notes in Computer Science*, pp. 160–171, Springer Berlin Heidelberg, 2011.

[60] S. Herrería-Alonso, M. Rodriguez-Perez, M. Fernández-Veiga, and C. Lopez-Garcia, "How efficient is energy-efficient ethernet?," in *2011 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 1–7, IEEE, 2011.

[61] 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*, LSAP '09, (New York, NY, USA), pp. 19–26, ACM, 2009.

[62] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, "Hsim: a MapReduce simulator in enabling cloud computing," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 300–308, 2013.

[63] "Controlled Delay (CoDel) Active Queue Management NS-2 code." http://pollere.net/Codel.html. Accessed: 2018-02-27.

[64] "Data Center TCP NS-2 code." http://simula.stanford.edu/~alizade/Site/DCTCP.html. Accessed: 2018-02-27.

[65] P. Reviriego, K. Christensen, J. Rabanillo, and J. Maestro, "An initial evaluation of energy efficient ethernet," *Communications Letters, IEEE*, vol. 15, pp. 578–580, May 2011.

[66] D. Dove, "A scalable base-t approach." http://www.ieee802.org/3/NGBASET/public/sep12/dove_01b_0912.pdf. Accessed: 2018-02-27.

[67] Cisco, "Cisco data center spine-and-leaf architecture: Design overview," tech. rep., 2016.

[68] Cisco, "Cisco's massively scalable data center: Network fabric for warehouse scale computer," tech. rep.

[69] W. Nelson, "Introduction to spine-leaf networking designs," tech. rep., 2017.

[70] A. Vahdat, M. Al-Fares, and A. Loukissas, "Scalable commodity data center network architecture," July 9 2013. US Patent 8,483,096.

[71] F. Testa and L. Pavesi, *Optical Switching in Next Generation Data Centers*. Springer, 2018.

[72] "Fat-tree design — clusterdesign.org." http://clusterdesign.org/fat-trees/. Accessed: 2018-02-27.

[73] E. Networks, "Extreme networks: Big data a solutions guide," tech. rep., 2014.

[74] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), pp. 266–277, ACM, 2011.

[75] Hortonworks, "Cluster planning guide." http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.4/bk_cluster-planning/content/hardware-for-slave.1.html. Accessed: 2018-02-27.

[76] "Uscs: Packet buffers." http://people.ucsc.edu/~warner/buffer.html. Accessed: 2018-02-27.

[77] 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)*, pp. 390–399, IEEE, July 2011.

[78] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads," *Proc. VLDB Endow.*, vol. 5, pp. 1802–1813, Aug. 2012.

[79] P. S. Center, "Pittsburgh supercomputing center: Enabling high performance data transfers." http://www.psc.edu/index.php/networking/641-tcp-tune#options. Accessed: 2018-02-27.

[80] N. Tiwari, S. Sarkar, U. Bellur, and M. Indrawan, "Classification framework of mapreduce scheduling algorithms," *ACM Comput. Surv.*, vol. 47, pp. 49:1–49:38, Apr. 2015.

[81] G. Wang, A. Butt, H. Monti, and K. Gupta, "Towards synthesizing realistic workload traces for studying the hadoop ecosystem," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2011 IEEE 19th International Symposium on*, pp. 400–408, July 2011.

[82] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 261–276, ACM, 2009.

[83] "Ibm knowledge center: Interrupt coalescing." http://www.ibm.com/support/knowledgecenter/ssw_aix_61/com.ibm.aix.performance/interrupt_coal.htm. Accessed: 2018-02-27.

[84] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), pp. 98–109, ACM, 2011.

[85] T. White, *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[86] S. Liu, H. Xu, and Z. Cai, "Low latency datacenter networking: A short survey," *CoRR*, vol. abs/1312.3455, 2013.

[87] S. N. Ismail, H. A. Pirzada, and I. A. Qazi, "On the effectiveness of codel in data centers," tech. rep.

[88] 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), pp. 145–157, USENIX Association, 2015.

[89] P. Teymoori, D. Hayes, M. Welzl, and S. Gjessing, "Even lower latency, even better fairness: Logistic growth congestion control in datacenters," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 10–18, Nov 2016.

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

[91] "How-to: Select the right hardware for your new hadoop cluster." https://blog.cloudera.com/blog/2013/08/how-to-select-the-right-hardware-for-your-new-hadoop-cluster/. Accessed: 2018-02-27.

[92] L. A. Barroso, "Landheld computing," in *International Solid-State Circuits Conference 2014*, ISCCC '14, 2014.

[93] "Apache spark - lightning-fast cluster computing." https://spark.apache.org/. Accessed: 2018-02-27.

[94] P. Fuentes, J. L. Bosque, R. Beivide, M. Valero, and C. Minkenberg, "Characterizing the communication demands of the graph500 benchmark on a commodity cluster," in *2014 IEEE/ACM International Symposium on Big Data Computing*, pp. 83–89, Dec 2014.

[95] "Mpi-sctp." http://www.cs.ubc.ca/labs/dsg/mpi-sctp/. Accessed: 2018-02-27.

[96] "Why is sctp needed given tcp and udp are widely available?." http://www.isoc.org/briefings/017/index.shtml. Accessed: 2018-02-27.