Energy Efficient Ethernet on MapReduce Clusters: Packet Coalescing To Improve 10GbE Links

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

***Abstract*—An important challenge of modern data centers is to reduce energy consumption, of which a substantial proportion is due to the network. Switches and NICs supporting the recent Energy Efficient Ethernet (EEE) standard are now available, but current practice is to disable EEE in production use, since its effect on real world application performance is poorly understood. This article contributes to this discussion by analysing the impact of EEE on MapReduce workloads, in terms of performance overheads and energy savings. MapReduce is the central programming model of Apache Hadoop, one of the most widely used application frameworks in modern data centers.**

We find that, while 1GbE links (edge links) achieve good energy savings using the standard Energy Efficient Ethernet implementation, optimum energy savings in the 10GbE links (aggregation and core links) are only possible if these links employ packet coalescing. Packet coalescing must, however, be carefully configured in order to avoid excessive performance degradation. With our new analysis of how the static parameters of packet coalescing perform under different cluster loads we were able to cover both idle and heavy load periods that can exist on this type of environment. Finally, we evaluate our recommendation for packet coalescing for 10GbE links using the energy–delay metric. This article is an extension of our previous work [1], which was published in Proceedings of the 40th Annual IEEE Conference on Local Computer Networks (LCN 2015).

***Keywords*—IEEE 802.3az, Green Ethernet, Energy Efficiency, Packet Coalescing, MapReduce, Hadoop**

I. Introduction

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 [2]. 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 [3]. Another study even showed that the cost of energy on current data centers had exceeded the cost of the hardware [4].

A significant proportion of a data center's energy consumption is caused by the network. D. Abts et al. [5] 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% [6], 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 [7], always consume full power, even when the link is idle [8]. 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 [9].

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 [10] and scientific computing [11]. Since EEE can incur significant performance overheads, many system vendors still advise their customers to disable it in production use [12]–[14], at least until its impact on real applications is better understood.

Our previous work was the first to study the impact of Energy Efficient Ethernet on MapReduce workloads [1]. MapReduce [15] and its open-source implementation, Apache Hadoop [16], are widely used for the processing of huge data sets on large commodity clusters. MapReduce presents a specific traffic pattern, including all-to-all communication in the shuffle phase, between mappers and reducers. It is also representative of a wider phenomenon, the move from traditional north–south data traffic, i.e. between the data center and external users; towards east–west traffic, i.e. among servers inside the same data center. In fact, more than 75% of the total traffic nowadays remains inside the data center [17].

Our work can be used to estimate the suitability of EEE for applications that follow the MapReduce programming model, in terms of both performance and energy. We find 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

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

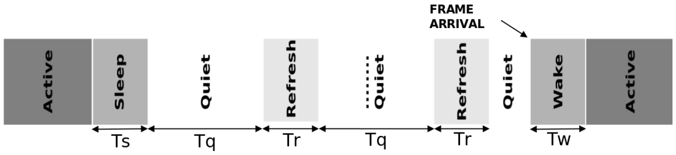


Fig. 1. Timeline of a link using Energy Efficient Ethernet

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 paper extends 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. This paper quantifies 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.

The rest of the paper is organized as follows: Section II compares our approach with related work. Section III describes the experimental methodology. Section IV presents the quantitative results and analysis, from which Section V distils the most important recommendations. Finally, Section VI concludes the paper.

II. Related Work

In this section we present the related work of Energy Efficient Ethernet, identify related problems in modern data center networks that can be made worse by it and differentiate the studies related with specific network traffic patterns using Energy Efficient Ethernet.

A. *Energy Efficient Ethernet*

IEEE 802.3az Energy Efficient Ethernet (EEE) was approved by IEEE in September 2010 [18]. Since Ethernet is the dominant technology for wire-line LANs, the power saving mechanisms of EEE are expected to bring considerable energy savings [10]. EEE has already been deployed, but many system vendors advise their customers to disable it in production use [12]–[14], 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 [12]. 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 [13]. Dell also presents a troubleshooting section related to EEE [14].

The Energy Efficient Ethernet standard defines the low-level mechanisms for entering and leaving sleep mode, known as Low Power Idle (LPI). Figure 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.

TABLE I
EEE single-frame efficiency

			1,500-byte frame		150-byte frame	
Speed	Min. T_w (µs)	Min. T_s (µs)	T_{frame} (µs)	Efficiency (%)	T_{frame} (µs)	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 [19]. The extreme case is illustrated in Table I, 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.

To understand the context, Figure 2 (reproduced from Christensen et al. [8]) 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

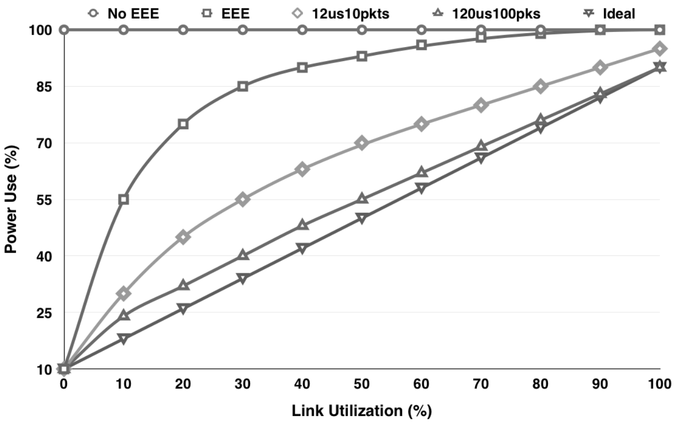


Fig. 2. Normalized link power consumption as a function of utilization, assuming Poisson arrivals (redrawn from [8])

the traffic pattern (which often deviate significantly from Poisson) and network load [8], [19]. Proposals include Power Down Threshold [11], 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 [8], [19].

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 [20]. Christensen et al. [8] 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, 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 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 [19], 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 Section IV, 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.

B. Challenges of TCP in Modern Data Centers

The MapReduce programming model targets commodity hardware and network equipment using the TCP/IP protocol [15] . More generally, recent studies show that 97% of the traffic in current data centers is carried by IP packets, being either TCP or UDP segments depending on the workload [21]. Microsoft Research published a study of 150 TB of network traces, which showed that TCP segments make up more than 99% of the internal traffic of their data center [22].

TCP was initially designed for Wide Area Networks (WANs) [23], and certain aspects, such as the minimum Retransmition Timeout (RTO) of 200 ms are better suited to WANs than to LANs. Problems that arise in a low-latency environment include (a) *TCP Incast* [23], a dramatic loss in throughput for many-to-one communication patterns, where congestion leads to packet loss, (b) *TCP Outcast* [24], where (surprisingly) the throughput to a congested node may be much lower from nearby nodes than from more distance ones, and (c) *Bufferbloat* [25], where congestion causes excessive packet buffering, leading to high latency and latency variability.

These phenomena are related to congestion, and they can be alleviated using a congestion-free network such as DCTCP [22], but such technology is not yet mainstream and is still not trivial to deploy. On the performance perspective, limiting buffer utilization may not be recommended once it can degrade performance of batch workloads such as Hadoop. TCP works well under congestion and burstiness scenarios as long as there is enough buffer to accommodate them [26].

C. MapReduce and Hadoop

In 2004 Google introduced the MapReduce programming model for reliable fault-tolerant processing of huge data sets on large commodity clusters [15]. 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 a distributed filesystem known as Apache HDFS (Hadoop Distributed File System), which uses disks attached to the same nodes used for computation. The job scheduler tries to schedule tasks to run on nodes where data is already present, resulting in high data locality [16].

The MapReduce framework first splits the input data set into independent chunks, which are processed in parallel by the map tasks. It then sorts the combined outputs from the maps, in the so-called shuffle stage, which involves all-to-all communication among nodes, and passes the sorted data to the reduce tasks. Several open-source MapReduce frameworks have been developed over the years, with by far the most popular one being Apache Hadoop [16].

D. 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 [10]. Their simulation results showed that UDP video streaming could achieve good energy savings without the

TABLE II
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	

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 [11]. 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 [27]. Dickov et al. presented an analysis of data compression for InfiniBand network energy savings [28]. They also introduced a novel power reduction software manager for InfiniBand links [29], [30]. 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 [11]. 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.

III. METHODOLOGY

This section describes the experimental methodology employed in this article.

A. 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 [31]. This simulator has been extended with a model of Energy Efficient Ethernet [32], which has been previously validated [8] and used extensively in previous work [33]. The network simulator is driven by the MRPerf MapReduce simulator [34]. 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. This methodology gives full visibility of the fine-grain details of the TCP/IP protocol in the data center environment. NS–2 and MRPerf are open source and EEE module can be obtained contacting referenced authors [32], so using the parameters described in the next section, our simulation methodology has the advantage that it can be reproduced and future work can be carried out on it.

1) Hardware configuration: The simulated hardware is shown in Table II. 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

TABLE III
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 IV
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

or 4:1. This matches Cisco's recommendation that MapReduce clusters should be deployed with an over-subscription ratio of 4:1 or lower at the access layer [35]. Lower over-subscription ratios improve network performance at higher cost [36], so we explored multiple points along this performance–cost tradeoff.

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 Section II. These problems can be alleviated using expensive switches with larger (deep) buffers. Manufacturers rarely disclose the buffer sizes in the product data sheet, so we followed the best public source we could find [37], 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 [38]. 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 [39]. 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 the table, 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 vendors to be considered. We restricted the power consumption to the NICs as considering power consumption for the servers could translate to a even wider range of values and great imprecision since different architectures and solutions could be adopted (going from low end and 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 section.

2) Workloads: Table II 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 IV-C4, 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% [40]. The large workload is closer to batch processing for big data applications [41], and we engage the whole system using a single large job, with the number of map and of reduce tasks both equal to twice the number of worker nodes.

Table III 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 [42].

3) EEE settings: We assume the sleep and wake timings given in Table I, and evaluate several control algorithms. We begin by evaluating Power Down Threshold [27], or stall timer, without the use of packet coalescing. We use the best stall timer value, with the packet coalescing settings in Table IV.

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) 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

B. Total runtime vs. Average runtime

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 [43]. The MrPerf MapReduce simulator provides multiple scheduler implementations, but the recommended scheduler is Quincy [44], which provides fair scheduling with data locality [45].

Total runtime and average runtime both grow with cluster utilization, but the relationship between them depends on the scheduler. Figure 3 shows the behavior using the Quincy scheduler. 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.

C. Variability

Each point in Figures 4, 5, 6, and 7 is the result from a single run. During our analysis and experimentation we saw small levels of variability (when we adjusted various parameters - the simulator itself is deterministic). The greatest variability among our results was about 1%, and mainly caused by dynamic scheduling and the TCP congestion control algorithm.

IV. Results

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

A. 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.

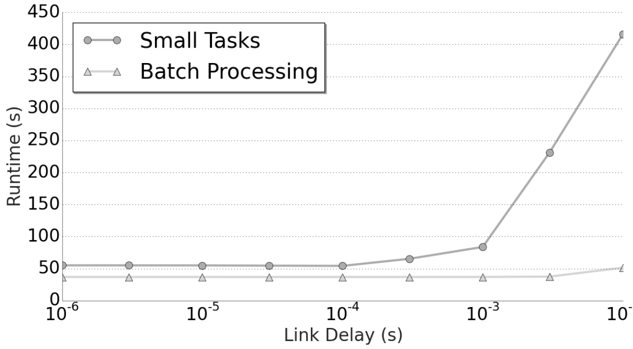


Fig. 4. Runtime vs. fixed link latency per link

As shown in Figure 4, for small tasks the runtime begins to increase only when the latency per link exceeds about $100\,\mu s$. 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.

We conclude that, for both workloads, and even with the default TCP settings, the impact of the 1GbE wakeup latency of $16.5\,\mu s$ 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.

B. Standard EEE and stall timer

The simplest EEE control algorithm puts the link into Low Power Idle (LPI) mode as soon as it becomes inactive [18]. 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 $50\,\mu s$ (see Section II-A). If the stall

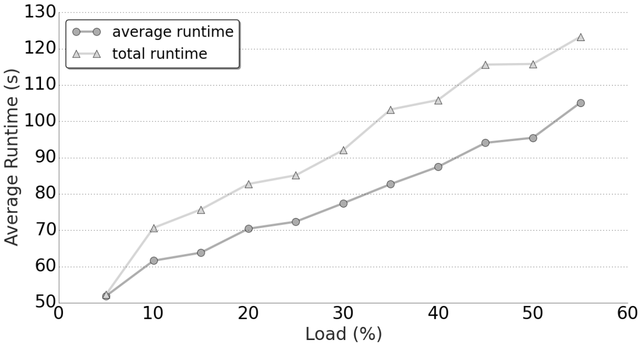


Fig. 3. Average and total runtime vs. load (Quincy scheduler)

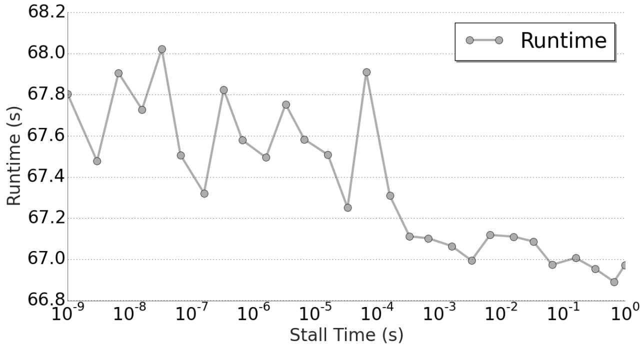


Fig. 5. Runtime vs. stall time without packet coalescing (small tasks)

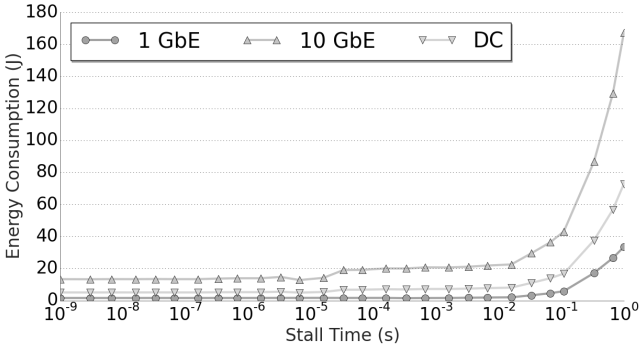


Fig. 6. Average energy per port vs. stall time without packet coalescing (small tasks)

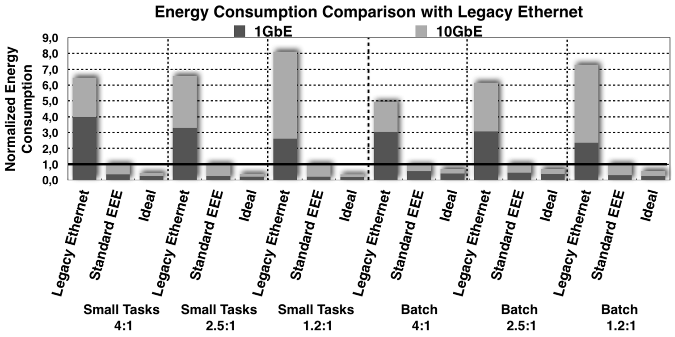


Fig. 7. Average energy consumption (comparison with legacy Ethernet)

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 [11].

We evaluated the effect of the stall timer setting, as shown in Figure 5 (runtime) and Figure 6 (energy consumption per port). Figure 6 shows the average energy consumption of the 1GbE and 10GbE ports, as well as 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 from scheduling and other noise. We therefore use the simple control algorithm without packet aggregation (nopa), which we refer to as *Standard EEE*.

Figure 7 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 7, 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.

C. 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 [8], [19], 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).*

1) Uniform EEE settings: The broad results in more detail are shown in Figure 8. Results are given for the five packet aggregation settings from Table IV, 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 9, which shows the performance and energy results averaged across all six scenarios (workloads and over-subscription ratios).

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 9, 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

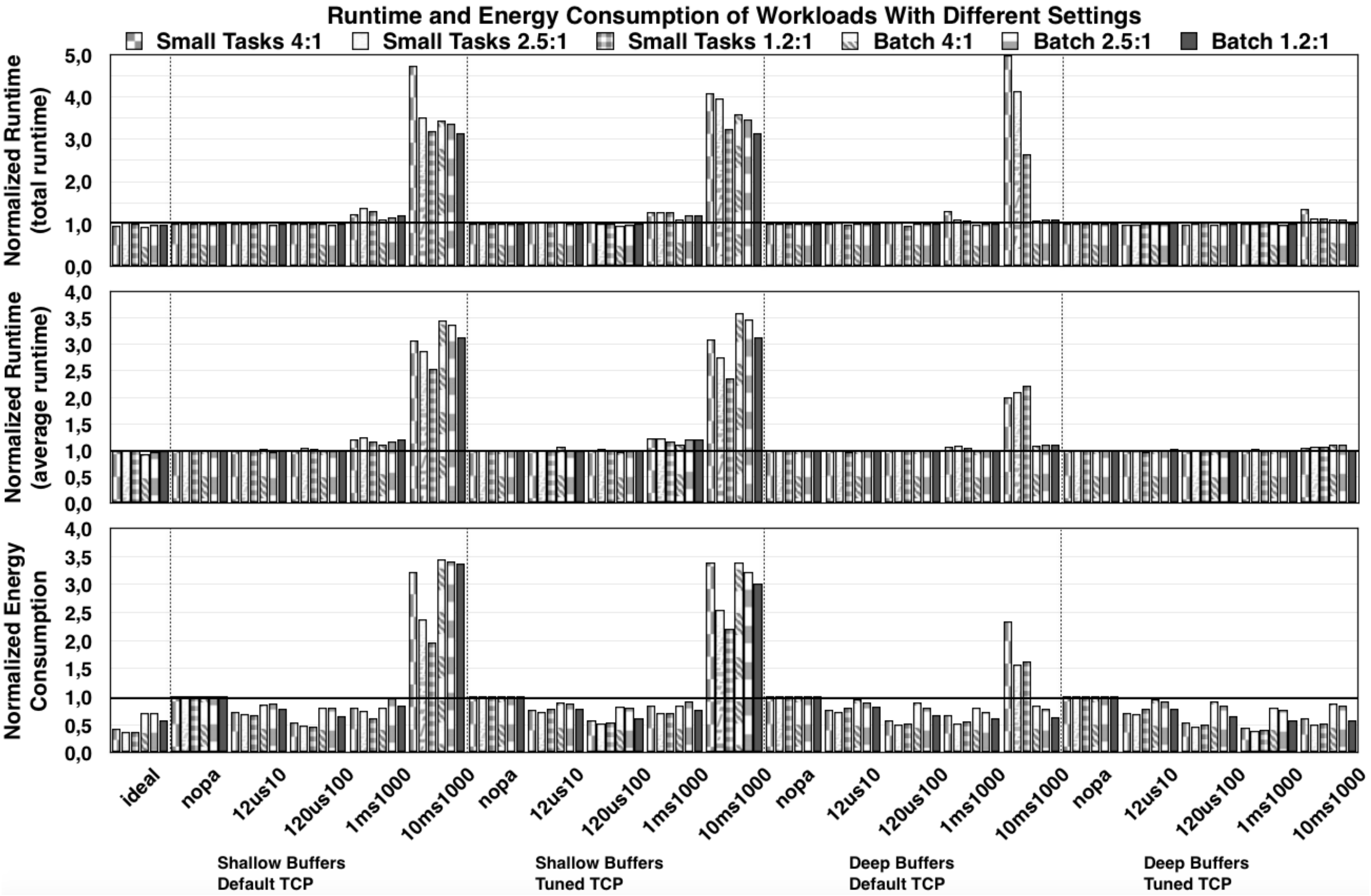


Fig. 8. All runtimes and energy consumption of workloads using different settings

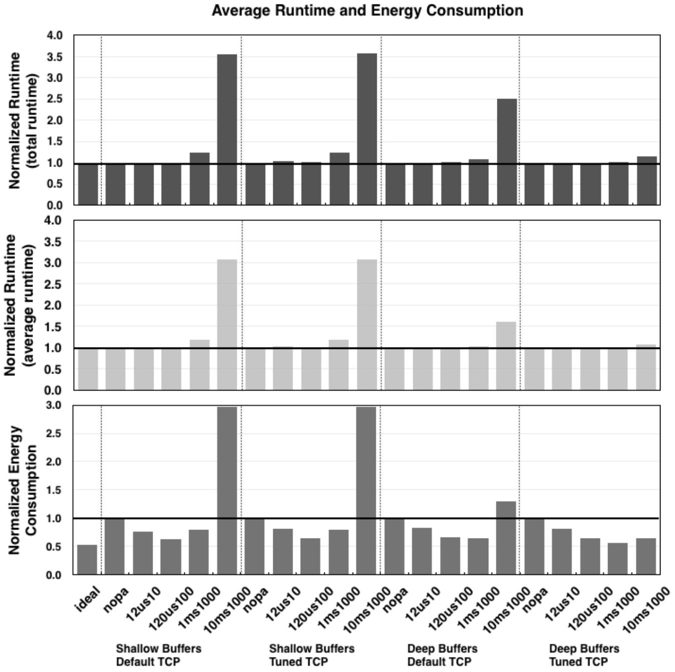


Fig. 9. Average Runtime and Energy Consumption of MapReduce jobs

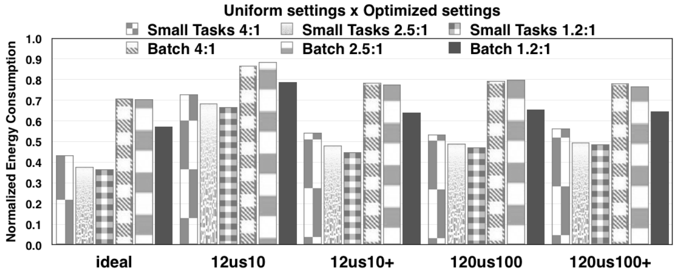
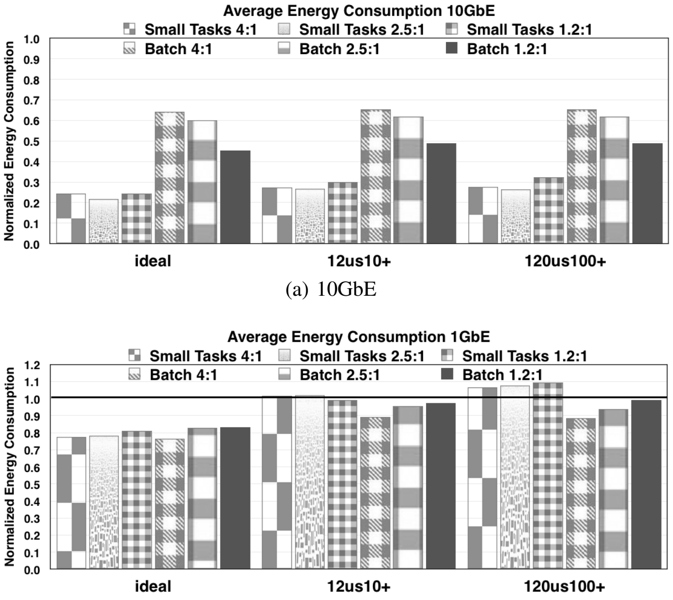


Fig. 10. Energy consumption (optimized configuration) [runtime remains about the same]

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.

2) Non-uniform EEE settings: In Section IV-C1, 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.



(a) 10GbE

(b) 1GbE

Fig. 11. Detailed energy consumption by NICs

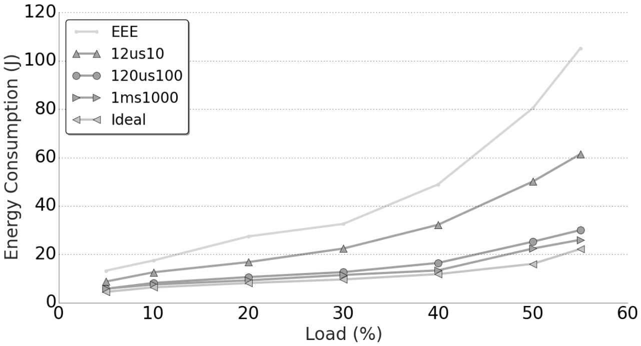


Fig. 13. Energy consumption as CPU load is varied (Terasort)

As shown in Figure 10, 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.

3) Analysis by link type: Whereas Figure 10 showed the average energy consumption across all the network links, Figure 11 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 11a. In contrast, Figure 11b 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.

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 [8] discussed in Section II, 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 nonlinear relationships between cluster utilization, measured using the CPU load, and both network utilization and runtime.

We concentrated on Terasort jobs on switches with shallow buffers, and varied the CPU load between 5% to 55%. Figure 12 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

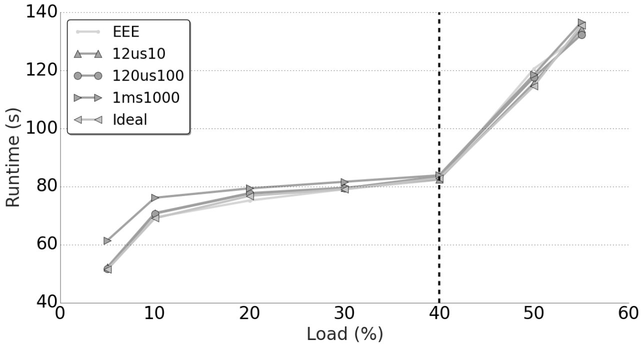


Fig. 12. Total runtime as CPU load is varied (Terasort)

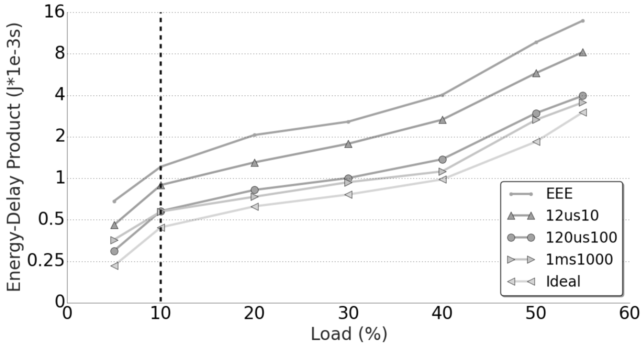


Fig. 14. Energy–delay product as CPU load is varied (Terasort)

that happens on the link connecting the Top-of-Rack switch to the reduce node (see Section II). 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 13, the lowest energy consumption is always achieved using 1ms1000. As previously mentioned, however, below about 40% utilization, the increase in runtime, visible in Figure 12, becomes too large, so the best option, with small difference in energy, would be 120us100.

Finally, Figure 14 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.

V. Discussion and Recommendations

As mentioned in the introduction, a typical data center network accounts for 12% or more of the total system energy consumption. 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 paper 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.

a) Recommendations for system administrators: The first finding of this paper 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 [11]. 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.

b) 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 13).

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 [1] 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 [46]. 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 EEE 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, whould not be expected to significantly impact Hadoop performance.

We investigated the effect of over-subscription on these findings, as shown in Figure 10. 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.

c) How buffering and burstiness really affect Hadoop: The results presented here are aligned with a new study investigating 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 [26].

VI. 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 adopted in practice until their effects on workload performance are well understood.

This paper extended our previous work investigating Energy Efficient Ethernet for MapReduce workloads. Our last findings widen our previous study by demonstrating how cluster utilization plays an important role when choosing the settings for packet coalescing. 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. As future work, we plan to extend our study to use a protocol that reacts faster to congestion, reducing the occupancy of network buffers and improving the ability to absorb burstiness introduced by packet coalescing.

VII. Acknowledgment

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

References

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

[2] 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)*, ser. LCN '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 505–512. [Online]. Available: http://dx.doi.org/10.1109/LCN.2012.6423667

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

[4] C. L. Belady, "In the data center, power and cooling costs more than the it equipment it supports," http://www.electronics-cooling.com, 2007.

[5] 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*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 338–347. [Online]. Available: http://doi.acm.org/10.1145/1815961.1816004

[6] 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.

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

[8] 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, no. 11, pp. 50–56, November 2010.

[9] "Broadcom at interop: Energy efficient ethernet is good for the planet," http://www.broadcom.com/blog/network-infrastructure/broadcom-at-interop-energy-efficient-ethernet-is-good-for-the-planet/, accessed: 2016-01-16.

[10] 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.

[11] 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)*. IEEE, April 2013, pp. 205–214.

[12] Cisco Systems, Inc, "IEEE 802.3az energy efficient ethernet: Build greener networks," Tech. Rep., 2011.

[13] Yamaha, "Disabling energy efficient ethernet (eee)," http://www.yamahaproaudio.com/global/en/training_support/selftraining/dante_guide/chapter2/02_eee/, accessed: 2015-02-04.

[14] Dell, "Resolving issues with energy efficient ethernet (eee) or green ethernet," http://www.dell.com/support/Article/us/en/19/421774/EN, accessed: 2015-02-04.

[15] 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*, ser. OSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251254.1251264

[16] The Apache Software Foundation, "Apache Hadoop Project," http://hadoop.apache.org, accessed: 2015-01-29.

[17] Cisco Systems, Inc, "Cisco global cloud index: Forecast and methodology, 20132018," Tech. Rep., 2014.

[18] The Institute of Electrical and Electronics Engineers, Inc, "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.

[19] P. Reviriego, J. Maestro, D. Larrabeiti, and D. Larrabeiti, "Burst transmission for energy-efficient ethernet," *Internet Computing, IEEE*, vol. 14, no. 4, pp. 50–57, July 2010.

[20] S. Herrerá-Alonso, M. Rodríguez-Pérez, M. Fernández-Veiga, and C. López-Garcá, "Optimal configuration of energy-efficient ethernet," *Computer Networks*, vol. 56, no. 10, pp. 2456 – 2467, 2012, green communication networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128612000990

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

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

[23] 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*, ser. WREN '09. New York, NY, USA: ACM, 2009, pp. 73–82. [Online]. Available: http://doi.acm.org/10.1145/1592681.1592693

[24] 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*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 30–30. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228339

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

[26] R. F. E. Silva and P. M. Carpenter, “Controlling network latency in mixed hadoop clusters: Do we need active queue management?” in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, Nov 2016, pp. 415–423.

[27] 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*, ser. ICS ’14. New York, NY, USA: ACM, 2014, pp. 313–322. [Online]. Available: http://doi.acm.org/10.1145/2597652.2597671

[28] 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)*. IEEE, Oct 2014, pp. 73–80.

[29] ——, “Software-managed power reduction in infiniband links,” in *2014 43rd International Conference on Parallel Processing (ICPP)*. IEEE, Sept 2014, pp. 311–320.

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

[31] “Network simulator ns-2,” http://www.isi.edu/nsnam/ns, accessed: 2015-01-29.

[32] P. Reviriego, K. Christensen, A. Snchez-Macin, and J. Maestro, “Using coordinated transmission with energy efficient ethernet,” in *NETWORKING 2011*, ser. Lecture Notes in Computer Science, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds. Springer Berlin Heidelberg, 2011, vol. 6640, pp. 160–171. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20757-0_13

[33] 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)*. IEEE, 2011, pp. 1–7.

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

[35] Cisco Systems, Inc, “Big data in the enterprise - network design considerations white paper,” Tech. Rep., 2011.

[36] Hortonworks, “Cluster planning guide,” http://docs.hortonworks.com/HDPDocuments/HDP1/HDP-1.3.7/bk_cluster-planning-guide/content/typical-hadoop-cluster-hardware.html, accessed: 2015-03-16.

[37] “USCS: Packet Buffers,” http://people.ucsc.edu/~warner/buffer.html, accessed: 2015-03-18.

[38] P. Reviriego, K. Christensen, J. Rabanillo, and J. Maestro, “An initial evaluation of energy efficient ethernet,” *Communications Letters, IEEE*, vol. 15, no. 5, pp. 578–580, May 2011.

[39] D. Dove, “A scalable base-t approach,” http://www.ieee802.org/3/NGBASET/public/sep12/dove_01b_0912.pdf, accessed: 2015-03-16.

[40] 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*. IEEE, July 2011, pp. 390–399.

[41] 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, no. 12, pp. 1802–1813, Aug. 2012. [Online]. Available: http://dx.doi.org/10.14778/2367502.2367519

[42] Pittsburgh Supercomputing Center, “Pittsburgh supercomputing center: Enabling high performance data transfers,” http://www.psc.edu/index.php/networking/641-tcp-tune#options, accessed: 2015-03-16.

[43] N. Tiwari, S. Sarkar, U. Bellur, and M. Indrawan, “Classification framework of mapreduce scheduling algorithms,” *ACM Comput. Surv.*, vol. 47, no. 3, pp. 49:1–49:38, Apr. 2015. [Online]. Available: http://doi.acm.org/10.1145/2693315

[44] 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*, July 2011, pp. 400–408.

[45] 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*. ACM, 2009, pp. 261–276.

[46] “Ibm knowledge center: Interrupt coalescing,” http://www.ibm.com/support/knowledgecenter/ssw_aix_61/com.ibm.aix.performance/interrupt_coal.htm, accessed: 2016-12-21.

Renan Fischer e Silva is a Ph.D researcher at the Barcelona Supercomputing Center (BSC). He received both his B.Sc. and M.Sc. in computer science from the Federal University of Paraná, Brazil, and is currently on the path of his Ph.D. in computer architecture from the Universitat Politècnica de Catalunya (UPC), started in 2014. Prior to starting his Ph.D., he worked in industry for 7 years. His research interests are data center infrastructure, local area networks and big data workloads.

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