Exploring Interconnect Energy Savings Under East-West Traffic Pattern of MapReduce Clusters

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. Energy Efficient Ethernet (EEE) is a recent standard that aims to reduce network power consumption, but current practice is to disable it in production use, since it has a poorly understood impact on real world application performance. An important application framework commonly used in modern data centers is Apache Hadoop, which implements the MapReduce programming model. This paper is the first to analyse the impact of EEE on MapReduce workloads, in terms of performance overheads and energy savings. We find that optimum energy savings are possible if the links use packet coalescing. Packet coalescing must, however, be carefully configured in order to avoid excessive performance degradation.**

***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 [1]. In 2006, U.S. data centers consumed three billion kilowatt-hours (kWh) per year, and, in 2015, although up-to-date figures are not available, power consumption is expected to have more than doubled [2]. More recently, in 2011, worldwide data centers were already responsible for 1 to 2% of total worldwide electricity consumption [3]. In the same year, U.S. data centers were anticipated to consume 100 billion kWh per year, at a cost of $7.4 billion per year [4]. A recent study even showed that the cost of energy on current data centers had exceeded the cost of the hardware [5].

A significant proportion of a data center's energy consumption is caused by the network. D. Abts et al. [6] 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% [7], divided among top of rack switches (15%), aggregation switches (10%) and core switches (5%). 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 [8], always consume full power, even when the link is idle [9].

The Energy Efficient Ethernet (EEE) standard, approved by IEEE in 2010, aims to improve 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.

This paper is, to the best of our knowledge, the first to study the impact of Energy Efficient Ethernet on MapReduce workloads. 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 external users and the data center; 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 all network links, not only those at the edges of the network, 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.

In short, our contributions 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

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

978-1-4673-6770-7/15/$31.00 ©2015 IEEE

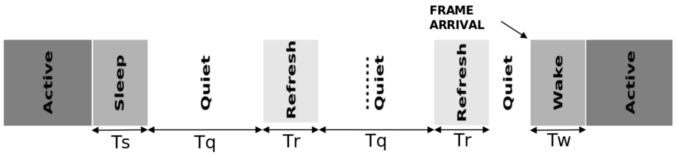


Fig. 1. Timeline of a link using Energy Efficient Ethernet

between performance and energy.

3) Dissolution of the different energy profiles of 1GbE and 10GbE links and recommended settings for each.

The rest of the paper is organized as follows: Section II provides the background about EEE, MapReduce, Hadoop, and the various TCP/IP problems encountered in modern data centers. Section III compares our approach with related work. Section IV presents the methodology, quantitative results and analysis, from which Section V distils the most important recommendations. Finally, Section VI concludes the paper.

II. Background

In this section we describe Energy Efficient Ethernet, summarize the MapReduce model, and identify problems in modern data center networks that can be made worse by Energy Efficient Ethernet.

A. Energy Efficient Ethernet

IEEE 802.3az Energy Efficiency 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. This is especially true when variable latencies cause packet loss or interfere with TCP/IP network congestion avoidance algorithms.

The EEE standard defines the low-level mechanisms for entering and leaving sleep mode, known as Low Power Idle (LPI). EEE is illustrated in Figure 1, which shows the timeline of a link that is initially active. The sleep transition into LPI mode requires time T_s. While 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 be woken from LPI mode, and doing so requires time T_w, which is approximately $4\,\mu\text{s}$ for 10GbE and $16\,\mu\text{s}$ for 1GbE, similar to the time to transmit a small number of 1,500-byte Ethernet frames.

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

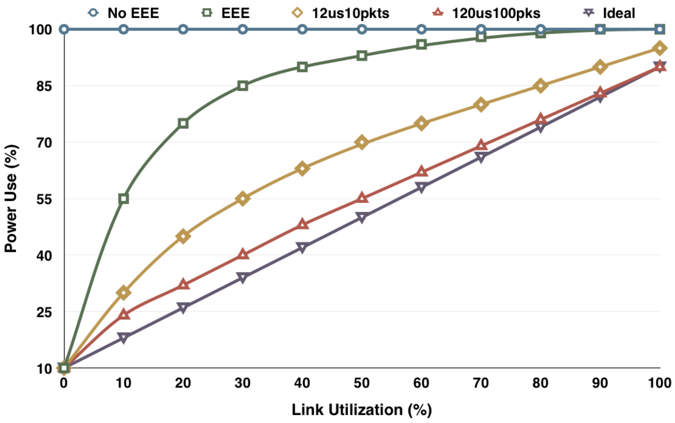


Fig. 2. Normalized link power consumption as a function of utilization, assuming Poisson arrivals (redrawn from [9])

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%. If the load is low, the link may be woken and put to sleep just to transmit an isolated frame, consuming excessive energy and incurring a high latency penalty [19]. This is illustrated in Table I, which summarises the energy efficiency, assuming that the link wakes and sleeps to transmit a single frame.

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, as described below in Section III. Previous studies show that the energy savings depend on the traffic pattern and network load [9], [19]. Proposals include Power Down Threshold [11], or stall timer, which initiates sleep after a defined period of inactivity, typically about $50\,\mu\text{s}$. Another technique, packet coalescing, 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 [9], [19].

Packet coalescing (also known as packet aggregation) introduces a significant, and variable latency, and it is not clear which workloads can tolerate this extra latency. 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]. In [9] the authors suggest using either a timer value of $12\,\mu\text{s}$ and a trigger of 10 packets or $120\,\mu\text{s}$ and 100 packets. Their results, reproduced in Figure 2, assume that the traffic is characterised as Poisson arrivals. Another publication uses substantially different values [19], of $1\,\text{ms}$ and $10\,\text{ms}$ as timers, in both cases with 1000 packets as trigger. We show the effect for MapReduce workloads in Section IV, where we modify the parameters, including configurating different devices to use different settings.

B. 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 reexecuting failed tasks. The MapReduce framework 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]. Hadoop uses the HDFS (Hadoop Distributed File System), which uses disks attached to the same nodes used for computation.

C. Challenges of TCP in Modern Data Centers

The MapReduce programming model [15] targets commodity hardware and network equipment using the TCP/IP protocol. 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. Network equipment buffers are the main responsible for congestion of the classic TCP protocol model [25], and commodity hardware switches tend to have small (shallow) buffers. Since the classic TCP protocol takes a considerable time to react to congestion, shallow buffers will drop packets, adversely affecting network throughput [22].

III. Related Work

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].

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 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.

Regarding packet coalescing, previous authors have recommended a wide range of parameters, including holding times of 1 ms or 10 ms [19], and the much smaller 12 µs and 10 packets or 120 µs and 100 packets [9]. 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.

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 [26]. Dickov et al. presented an analysis of data compression for Infiniband network energy savings [27]. They also introduced a novel power reduction software manager for Infiniband links [28]. Both techniques of Dickov et al. are implemented in the MPI software layer, so they are only applicable to workloads written using MPI.

HPC workloads have complex dependencies and require low latency, and neither work considered packet coalescing to be useful. Another important difference with our work is that we use a detailed packet-level simulator. Both Dickov and Saravanan use high-level simulation models that abstract away fine-grain details. We found that in our context, especially with commodity switches, packet-level phenomena, such as Ethernet packet loss and the TCP/IP congestion avoidance algorithm, have a critical effect on both performance and energy. In this context, accurate quantitative results can therefore only be obtained using a packet-level simulator.

IV. Results

This section first describes the experimental methodology, and then presents the quantitative results, giving the energy savings and performance overheads for MapReduce workloads

TABLE II
SIMULATED ENVIRONMENT

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

TABLE III
SIMULATED BENCHMARKS

Benchmark	% of jobs	Input size (MB)	Shuffle size (MB)	Output size (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

using Energy Efficient Ethernet. We consider configurations with and without packet coalescing.

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 [29]. This simulator has been extended with a model of Energy Efficient Ethernet [30] and is driven by the MRPerf MapReduce simulator [31]. This methodology gives full visibility of the fine-grain details of the TCP/IP protocol in the data center environment. We did not use real hardware because the EEE control algorithm is typically implemented in NIC and switch firmware that is difficult to change.

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 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 [32]. Lower over-subscription ratios improve network performance at higher cost [33], exploring multiple points along the performance–cost trade-off.

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 [34], giving 128 KB per port for the commodity switches and 10 MB per port for the expensive switches.

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 [35]. 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 [36]. 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.

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 utilisation is about 40%. This 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% [37]. The large workload is closer to batch processing for big data applications [38], 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 performance metric is the total time needed to finish all of the jobs in the workload, which is inversely proportional to the effective throughput of the cluster. 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 utilisation. In a real system there may be a significant difference in performance, since 128 MB per core fits in DRAM caches while a realistic footprint would not, but this is not modelled in MRPerf. In addition, given that the sort is already large, both cases will have similar communication to computation ratio.

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 [39].

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, 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	Commodity *or* expensive switches
Workload	Small jobs *or* batch job
TCP window size	Default *or* optimized
Packet coalescing	See previous subsection

B. Variability

Each point in Figures 3, 4, 5, and 6 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, of about 1%, was mainly caused by dynamic scheduling and the TCP congestion control algorithm.

C. Fixed link latency

Since EEE can only affect execution time via its effect on latency, we begin by evaluating the effect of link latency on MapReduce performance. We added a constant latency on each link, without using EEE, for both workloads: small tasks and batch processing. Batch processing concentrates communication during a single shuffle phase, leading to heavy congestion, whereas small tasks have communication more distributed over time. Therefore, since the congestion is smaller, small tasks achieve greater throughput per link, requiring larger buffers to compensate for a particular delay.

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

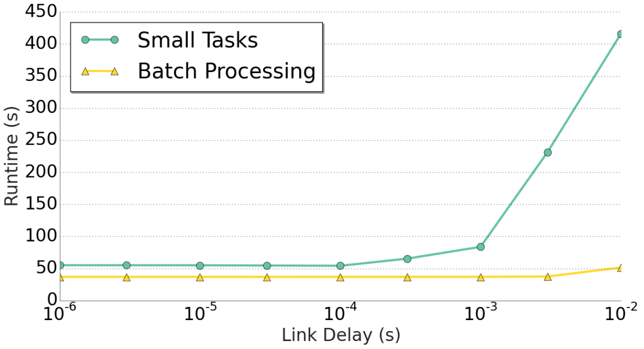


Fig. 3. Runtime time vs. fixed link latency per link

Regarding the default TCP settings (receive and send buffers and scale window), as shown in Figure 3, for small tasks the runtime begins to increase only when the latency per link exceeds about 100 µs. Batch processing is much less sensitive to latency and shows performance degradation only when the latency exceeds about 5 ms per link.

We conclude that the 1GbE wakeup latency of 16.5 µs should have a negligible impact on MapReduce performance, for both workloads, even with the default TCP settings. Since the latency is added per link, it includes the effect of consecutive wakeups on multiple hops.

D. 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

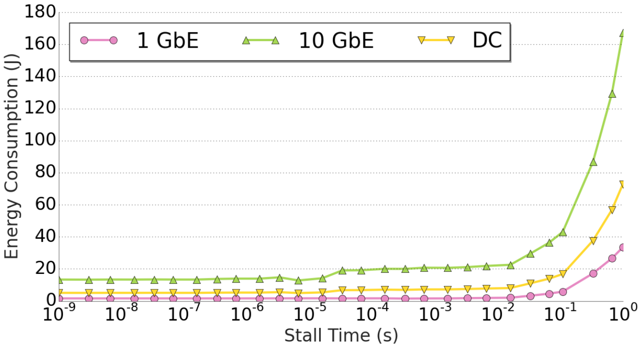


Fig. 4. Average energy per port vs. stall time without packet coalescing (small tasks)

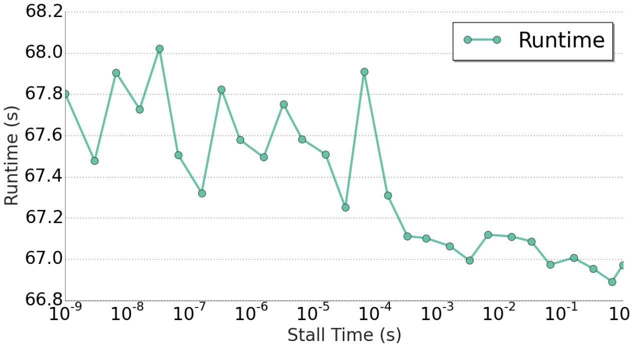


Fig. 5. Runtime vs. stall time without packet coalescing (small tasks)

stall timer, enters LPI mode after a defined period of inactivity, which is typically about 50 µs (see Section II-A). If the stall timer is too 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 too large, the links will seldom enter LPI mode, yielding poor energy savings. We therefore expect the stall timer to provide a trade-off between performance and energy.

We evaluated the effect of the stall timer setting, as shown in Figure 4 (energy consumption per port) and Figure 5 (runtime). Figure 4 also shows the average energy consumption of all ports of the Data Center (DC). These results show that 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 that is hard to distinguish from scheduling and other noise. We therefore refer to the simple control algorithm without packet aggregation (nopa) as *Standard EEE*.

Figure 6 compares the energy consumption of legacy Ethernet with Standard EEE. In this figure and in the next subsection, energy consumption is relative to Standard EEE, which is the current state of the art. We see that the energy consumption is reduced by a factor of between five and eight, depending on the scenario (workload and network over-subscription ratio). Figure 6 also shows the energy results for the ideal case, which has instantaneous zero energy sleep and wake transitions. In the following section we find that close to ideal energy savings are possible using packet aggregation.

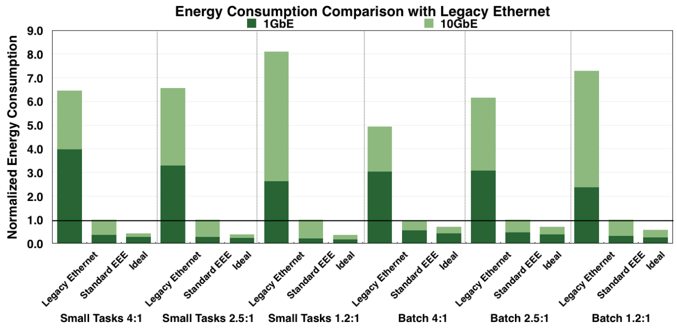


Fig. 6. Average energy consumption (comparison with legacy Ethernet)

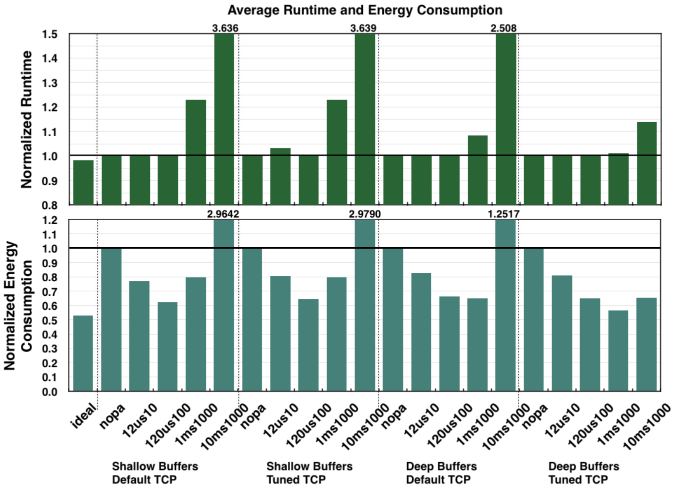


Fig. 7. Average Runtime and Energy Consumption of MapReduce jobs

E. Optimum Energy Savings on MapReduce Cluster

In contrast to previous recommendations for the deployment of EEE [9], [19], we find optimum energy savings from EEE across the whole network, specially for 10GbE, but only when packet coalescing is enabled. As showed by our results, packet coalescing parameters and TCP settings must, however, be carefully chosen. *The results were normalized to the standard Energy Efficient Ethernet (without packet coalescing).*

1) Uniform EEE settings: Figure 7 resumes the averages across the six scenarios (workload and over-subscription ratio), with zoomed vertical axes that show the most relevant results for the five packet aggregation settings from Table IV, default or optimised TCP buffers, and shallow or deep switch buffers. More detailed results which include the normalized runtime and energy results for each one of the six scenarios (workload and over-subscription ratio) are presented in Figure 8.

Regarding performance first, all results with the 12us10 and 120us100 settings increase runtime by less than 5% (Figure 7), with little variation among the six scenarios (Figure 8). Clearly, using deep buffer switches can help mitigate the impact of burstiness. But avoiding congestion does not eliminate the problem completely. Since delay is introduced on the network, it is necessary to tune TCP settings to allow more packets in flight and overcome the introduced delay.

In contrast, the 1ms1000 and 10ms1000 increase the runtime significantly, especially with shallow buffer (commodity) switches, where even the best case is three times slower than the baseline. The exception is the batch workloads on deep buffer switches, which do not need tuned TCP settings, independently of the subscription ratio. Congestion is the limiting factor during the shuffle phase when the cluster is fully utilized so the TCP congestion window does not grow enough to require tuned receive and send TCP buffers. In addition, the traffic load is sufficient to usually trigger the

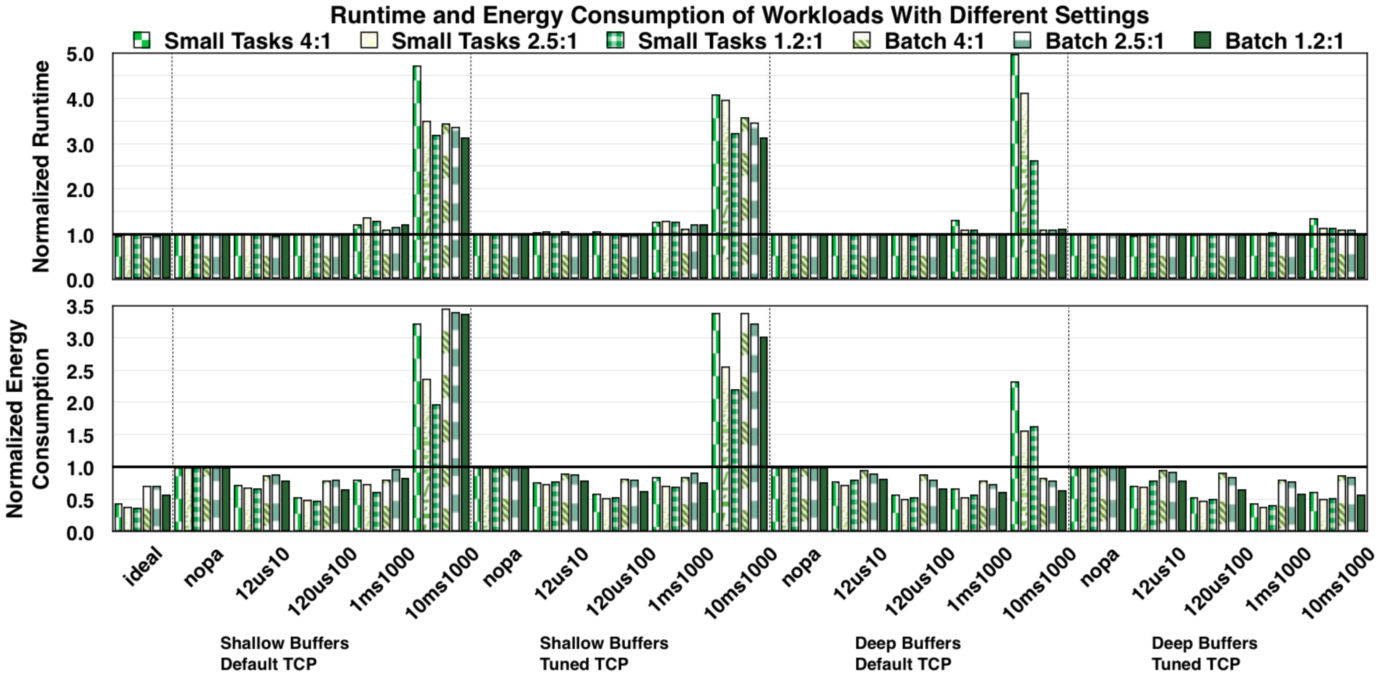


Fig. 8. All runtimes and energy consumption of workloads using different settings

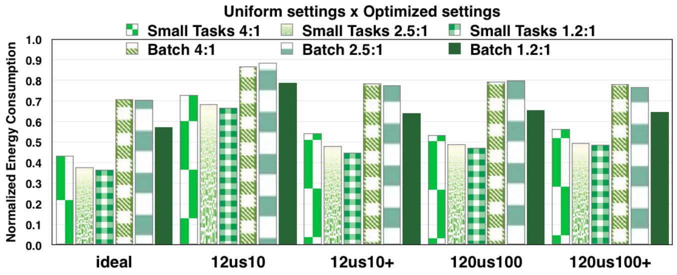
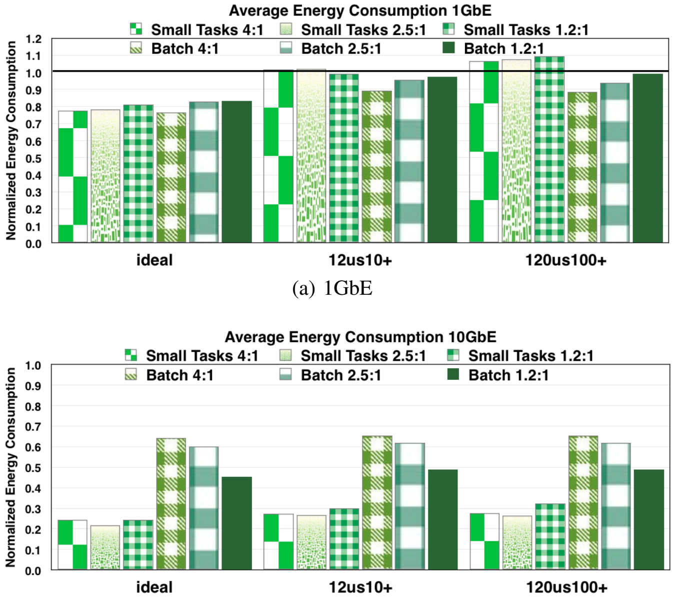


Fig. 9. Energy consumption (optimized configuration) [runtime remains about the same]

1000-packet threshold without waiting for the 10 ms timeout, so the runtime of these workloads is similar to 1ms1000. For small tasks, in contrast, even if the CPU utilization is high the data transfer between mappers and reducers will happen independently for each task what does not achieve the minimum load to trigger link re-activation.

Turning to the energy results, in Figure 7, we see that packet aggregation with 12us10 and 120us100 settings reduce energy consumption to 75% and 60% of Standard EEE, respectively. This is a good result because, as seen above, the runtime increases by less than 1%. In comparison, although 1ms1000 saves the most energy using deep buffers and tuned TCP settings, it increases the runtime by an unacceptable 25% in scenarios with shallow buffers. The final setting of 10ms1000 has even higher overheads.

2) Non-uniform EEE settings: As we can see from Figure 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. The next experiment consists in using 1ms1000 for the 10GbE NICs, and 12us10 or 120us100 for the rest of the data center. We expect to save additional energy and get closer to the ideal model.



(a) 1GbE

(b) 10GbE

Fig. 10. Detailed energy consumption by NICs

3) Analysis by link type: Finally, Figure 10 shows separate energy consumption results for (a) the 1GbE links and (b) the 10GbE links. As before, values are normalized in comparison with Standard EEE, and 12us10+ and 120us100+ are optimized settings, using 1ms1000 for the 10GbE links. The greatest savings are obtained for the 10GbE links, which benefit from packet coalescing, as shown in Figure 10b. In contrast, Figure 10a shows little benefit for the 1GbE links, from either 12us10+ or 120us100+, in comparison with Standard EEE.

V. Discussion and Recommendations

Our initial results showed that, so long as packet coalescing is disabled, the performance of MapReduce is not significantly affected by EEE. We found, however, that the energy consumption without packet aggregation was about twice the best energy consumption with packet aggregation.

After the experiment with latency per link (see Subsection IV-C), we believed that packet aggregation should not hurt the performance of MapReduce workloads considerably since these workloads are not latency sensitive. In contrast, our previous results show that it does affect performance. Coalescing packets not only introduces delay on links but it adds burstiness that is not desirable on Local Area Networks. Packet coalescing affects performance through its effect on burstiness on commodity switches, which leads to packet loss. Burstiness behaviour on LANs can hurt the performance of a workload such as MapReduce because we have many-to-one communication pattern and buffers are limited on commodity hardware. Our results demonstrate this.

We showed that standard Energy Efficient Ethernet (without packet coalescing) currently saves only half of what is potentially available using an ideal model, which excludes the overhead to sleep and wake the links. Because packet coalescing affects performance, the choice of settings to coalesce packets must seek for a balance between energy proportionality and performance degradation.

An ideal model without EEE overheads indicates that for 1GbE links would be possible to save at most about 20% more energy in comparison with standard EEE. In comparison with default EEE, when we applied packet aggregation on 1GbE links we were not able to accomplish more than 5% better energy savings, and in some cases the benefits were negligible.

For 10GbE links the energy savings are limited by the traffic load, specially for batch processing. Increasing network over-subscription consumed proportionally more energy since the network becomes the bottleneck of the system. Independently of the load and workload, we verify that is not only feasible as we actually were able to get close to the ideal, saving between 35% to 75% more energy in comparison with the default EEE.

As verified in Figure 9, the average energy consumption per NIC demonstrates that in the worst case (Batch 4:1), coalescing packets overpass standard EEE by 20% in average. Larger over-subscription ratios decrease the weight 10GbE links have in the average calculation. On the contrary, packet coalescing can surpass standard EEE saving between 50% and 60% more energy in average if the cluster is running small tasks (depending on the network over-subscription ratio).

As a further conclusion, by our results we indicate how to mitigate or eliminate the effect of burstiness on MapReduce clusters. It is necessary to use a transport protocol that reacts quickly to congestion. That is not the case for TCP, since it does not implement Explicit Congestion Notifications (ECNs) nor does it have a suitable Retransmission Timeout (RTO) for LANs. On the contrary, DCTCP [22] or other transport protocols that react fast to congestion may help to alleviate the problem since such protocols reduce buffer utilization while keeping the maximum throughput possible. Finally, we recommend using packet aggregation in a MapReduce cluster as long as it is carefully tailored for different equipment of different layers (edge or core). Our example shows it is better to use 12us10 for edge devices with 1GbE links and 1ms1000 for core devices with 10GbE links since they have higher traffic load and therefore there is more potential for benefits comparing with the standard Energy Efficient Ethernet.

VI. Conclusions

In this paper, we presented a novel analysis of Energy Efficient Ethernet for MapReduce clusters. We evaluated the performance impact and energy savings, and found, contrary to recommendations from manufacturers, that the MapReduce programming model is not sensitive to the overheads of EEE, even with packet coalescing. We even demonstrate that a simple control algorithm that switches idle links off immediately is sufficient, but that optimum energy savings in the 10GbE links are only possible with packet aggregation. We suggested tailored settings for use in edge devices (1GbE) and core devices (10GbE), improving the energy savings between 20% and 60% in comparison with standard EEE, depending on the workload and the network over-subscription ratio. As part of future work, we plan to extend our study to use a protocol that reacts faster to congestion, reducing the buffer utilization of network equipments.

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] 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

[2] P. Mahadevan, S. Banerjee, P. Sharma, A. Shah, and P. Ranganathan, "On energy efficiency for enterprise and data center networks," *Communications Magazine, IEEE*, vol. 49, no. 8, pp. 94–100, August 2011.

[3] J. Koomey, “Growth in data center electricity use 2005 to 2010,” *A report by Analytical Press, completed at the request of The New York Times*, 2011.

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

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

[6] 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

[7] 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.

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

[9] 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.

[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] I. Cisco Systems, “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] T. A. S. Foundation, “Apache hadoop project,” http://hadoop.apache.org, accessed: 2015-01-29.

[17] I. Cisco Systems, “Cisco global cloud index: Forecast and methodology, 20132018,” Tech. Rep., 2014.

[18] 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.

[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 (M N)*. 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] 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

[27] 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.

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

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

[30] 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

[31] 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

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

[33] 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.

[34] “Uscs: Packet buffers,” http://people.ucsc.edu/∼warner/buffer.html, accessed: 2015-03-18.

[35] 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.

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

[37] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, “The case for evaluating MapReduce performance using workload suites,” in *2011 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, July 2011, pp. 390–399.

[38] 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

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