Exploring Multiple Sleep Modes in On/Off based Energy Efficient HPC Networks

Karthikeyan P. Saravanan†, Paul M. Carpenter† and Alex Ramirez‡
†Barcelona Supercomputing Center (BSC) and Universitat Politècnica de Catalunya (UPC), Spain
Email: {karthikeyan.palavedu,paul.carpenter}@bsc.es
‡NVIDIA Research, USA
Email: aramirez@nvidia.com

***Abstract*—Energy efficiency is one of the key challenges in high-performance computing (HPC). The current target of 1 ExaFlop in 20 MW requires a ten-fold improvement in energy efficiency, which is only possible through significant improvements in the energy efficiency throughout the system. Interconnects are particularly inefficient, since their links are always on, consuming full power in order to provide low latency, even though the average interconnect utilization is low. To address the above, the Ethernet standards committee in-charge of 40/100/400Gb Ethernet has opted to include protocols that define low power modes, specifically Fast-Wake, alongside the older Deep-Sleep, to make interconnect links energy proportional. With these standards ratified as recently as March 2014, it is unclear how these low power modes can be used in HPC. While energy efficiency is critical, techniques with excessive performance overheads are unlikely to be adopted in HPC. To this end, this paper performs the first detailed analysis of Fast-Wake mode for link energy savings in the context of HPC. Our results show that a combination of Fast-Wake and Deep-Sleep can reduce link energy savings by up to 70% with less than 1% performance overheads. However, we show how the parameters of these low power modes must be carefully configured to obtain the right trade-offs in energy and performance. We believe that our analysis could benefit interconnect vendors looking to use these low power modes for deployment in HPC.**

I. Introduction

Energy efficiency and power consumption have become key challenges in the field of HPC. High energy consumption not only limits the feasibility of building next generation supercomputers, but could significantly increase their total cost of ownership. Exaflop supercomputers are expected to be built by the year 2018, with a power budget of about 20 MW [3].[1] Extrapolating today's machines shows that achieving this goal would require a ten-fold reduction in power consumption. Building HPC machines with such targets can only be practicable if all energy inefficiencies of the system are eliminated.

High performance interconnects consume a significant portion of system energy. Typical interconnects consume up to 12% of the total system energy at full load and even more when the application does not fully utilize the CPU and memory [5]. Interconnects, in particular, are not energy efficient because their links, which consume up to 65% of the total interconnect power, are essentially always-on, consuming energy even when idle [6, 10]. Links traditionally remain always-on, continually transmitting signals, for link alignment and synchronization [9]. Advances in energy proportionality of compute and memory elements further increases the proportion of system energy by interconnects.

Recognizing the need for energy proportional interconnects, the IEEE 802.3az Energy Efficient Ethernet (EEE) standard was ratified providing specifications to turn off links during periods of inactivity. While the standard provides mechanisms to turn on and off links, the mechanisms that govern power management are vendor specific and are an active area of research. Various studies target specific domains with different mechanisms for link on/off management [7, 9, 10]. Two key mechanisms for on/off management are frame buffering, which has been shown to work well for data center and Internet based workloads [9, 10] and Stall-Timer that has been shown to work well on HPC applications [7].

HPC applications typically require high performance interconnects to support their peak bandwidth, however, the average utilization of the network is low. This is because HPC applications are generally characterized by long periods of computation fragmented by bursts of communication [7]. Although this presents a clear opportunity for energy saving, HPC vendors typically do not use low power modes for interconnects due to their potential for large and unknown application performance overheads. Performance overheads when using on/off based links essentially stem from the added latency involved in waking up the link every time that a frame arrives while the link is off. The use of a Stall-Timer (as mentioned above) switches the link off only after it has been inactive for a defined period of time.

Energy Efficient Ethernet (EEE) protocol provides specifications for energy savings in 100Mb, 1Gb and 10Gb links. With the success of EEE, current and recent efforts for the standardization of 40Gb, 100Gb and 400Gb backplanes and optical Ethernet, have opted to include and have included energy savings mechanisms from EEE. While incorporating EEE for 40Gb, 100Gb and 400Gb links, the standard introduced an additional sleep state known as **Fast-Wake**. In Fast-Wake mode the link does not turn off all its components, but only a few to trade-off higher energy consumption for a faster wake-up time.

These protocols, specifically IEEE 802.3bj and 802.3bm, providing standards for 40Gb and 100Gb backplanes and optical Ethernet, respectively, were ratified as recently as March 2015, and IEEE 802.3bs for 400Gb is expected to be ratified in 2017. As with the original EEE protocol, products based on the recent standards, which include Fast-Wake, may be deployed for HPC within a year, but it is likely that Fast-

[1] Recent estimates suggest that 2018 and 20 MW may both be too optimistic.

Wake will be disabled by default. With about 40% of Top500 HPC machines using Ethernet based interconnects every year, that could potentially have these protocols in their switches; it is imperative to understand the need for Fast-Wake, its performance impact and possible configuration parameters in the context of HPC applications.

In this paper, we describe a comprehensive analysis of Energy Efficient Ethernet with Fast-Wake using traces from fifteen HPC applications of various domains obtained from production supercomputers. Our analysis answers the following important questions regarding the use of Fast-Wake, specifically:

1) How useful is Fast-Wake or an intermediate sleep state for HPC networks, or would a single Deep-Sleep mode be sufficient?
2) How long should the link remain in the intermediate Fast-Wake state before entering Deep-Sleep?
3) What is the reduction in performance overhead and the energy savings on using Fast-Wake, compared with EEE without Fast-Wake?

We believe that the answers to these questions would benefit interconnect vendors designing interconnects with EEE and Fast-Wake targeting HPC.

The main contributions of this paper are,

1) A detailed analysis of the Fast-Wake mode over HPC applications to understand its energy saving potential. To this end, we first present a separate analysis of Deep-Sleep and Fast-Wake, following which we present energy and performance results on the use of both low power modes combined. We show that using Fast-Wake and Deep-Sleep together outperforms the older Deep-Sleep mode in terms of energy and performance.
2) We extend our analysis to study Fast-Wake and Deep-Sleep over upper level links, we present an analysis of designing Fast-Wake with different wake-up timings and energy levels in its low power state.

The rest of this paper is organized as follows. Section II discusses background and prior work. In Section III, we present our experimental methodology, our analysis of the aforementioned energy savings mechanisms, Fast-Wake and Deep-Sleep, and our recommendations for using these mechanisms in HPC. Finally, in Section IV, we conclude our work.

II. Background and Related Work

In this section, we describe the background specific to Energy Efficient Ethernet and Fast-Wake, following which, we describe prior art and related work.

A. Background

This paper focuses on analyzing an intermediate sleep mode, specifically Fast-Wake for Energy Efficient Ethernet (EEE), in the context of HPC. To this end we first discuss the background and technical details of EEE: its power savings mechanisms, Deep-Sleep and the newly introduced Fast-Wake. Here, we also briefly discuss previous work on the use of Stall-Timer for HPC applications, and frame buffering, used in data centers and on the Internet.

Energy Efficient Ethernet (EEE): Already in the year 2010, the Internet and specifically data centers accounted for 1.1% to 1.5% of global energy consumption, with this percentage having doubled since 2005 [4]. The growing energy cost of Internet infrastructure and data centers have pushed for energy proportionality across the computing spectrum. Addressing the above, in 2010, the IEEE 802.3az Energy Efficient Ethernet Task Force published its standard for Ethernet energy efficiency [2, 9]. The goal of the task force was to reduce the significant contribution of network devices to the national power budget, especially since large sections of the Internet and data center infrastructure are built using Ethernet [9]. After considering various proposals, including adaptively changing the link rate, the task force adopted the proposal known as Low Power Idle (LPI) [2, 9].

Low Power Idle, or **Deep-Sleep** as it is now known, proposes modifications to existing Ethernet standards that allow the link to switch between "sleep" and "wake" modes on demand, to save energy. In Deep-Sleep mode, the link is still periodically refreshed and awaits frames, hence is not completely off. Arrival of a frame triggers the signaling of a wake-up transition to turn on the link. Frame transmission starts when the transceiver and receiver PHYs are both active. At the subsequent hop, arriving frames are buffered while a subsequent link is signaled for wake-up. Deep-sleep (or LPI) was considered straightforward to implement, since it freezes the state of the transceiver when the link enters sleep and it restores the state when it wakes [2]. The IEEE 802.3az standard was published for 10Gb Ethernet with its wake-up and sleep timings specified by the draft to be $4.16\,\mu s$ and $2.88\,\mu s$ respectively, with about 90% energy savings at low power mode compared to when the link is active [1]. Switches that support EEE, targeting data centers, were commercially available within a year from the date of standardization.

While EEE provides specifications for link on and off, the mechanisms behind when to do so are left to the vendor. An early evaluation by P. Reviriego, et al., showed how energy savings with EEE decrease quickly with an increase in link utilization [10]. According to their results, when average link utilization is 20%, its respective energy consumption was greater than 70%. The high energy consumption was attributed to workloads having a large number of small messages that frequently turned the links on and off; so much so that the time spent in switching between on and off states was significantly greater than the actual time spent in transmitting frames. A proposed solution to this problem, known as Frame Buffering or Packet coalescing [10], attempts to buffer frames (without waking up the link) up to a certain frame limit or a time-out period. With appropriate frame limits and time-out, the technique was shown to bring links closer to energy proportionality [9, 10]. This technique could however, only work on workloads that are not latency sensitive as buffering frames significantly increases the transmission latency.

Energy Efficient Ethernet in HPC: HPC applications, as previously mentioned, are characterized by long compute periods (where the network is idle) fragmented by short bursts of communication. These short bursts however, require peak bandwidth and are generally latency sensitive. Buffering frames for $100\,\mu s$, as recommended for Internet and data-center workloads, can cause high and unacceptable performance

overheads in HPC. For this reason, work by Saravanan, et al., proposes the use of a **Stall-Timer** that, contrary to buffering frames, leaves the link on (while idle), after every frame transmission [7].[2] The Stall-Timer ensures that subsequent frames do not incur any wake-up delays, and it elapses after a time-out period of inactivity. The authors show that performance overheads can be brought down to below 1%, using Stall-Timers, while still having, on average, up to 70% link energy saving. Stall-Timers do not have to be large in order to have the link remain on throughout a communication phase, as they are reset on every communication. This ensures that the link remains on during the small gaps in frame arrival during a typical communication phase. A short Stall-Timer expires soon after the network goes idle, at the start of a compute phase, leading to large energy savings.

Energy Efficient Ethernet (EEE) with Fast-Wake: With the 10Gb variant of Energy Efficient Ethernet (EEE) ratified in 2010, subsequent standardization efforts, which focused on building specifications for 40Gb, 100Gb and 400Gb backplane and optical Ethernet, adopted the energy savings mechanisms from EEE. In the process of integrating the aforementioned Deep-Sleep, an additional sleep state, Fast-Wake, was first introduced by the IEEE 802.3bj task force in charge of the 100Gb Backplane and copper cable standardization effort. The motivation for a Fast-Wake mode came from the relatively high wake-up time of the Deep-Sleep mode, whose effect on performance is more pronounced at higher link speeds. The possible increase in latency due to Deep-Sleep is considered to be too high, so it is expected to be disabled for 100Gb links. The long wake-up delay in the original EEE standard comes from signaling components, specifically the PMA and PMD components in the PHY,[3] that are required to synchronize the transmitting and receiving links before data transmission. In the case of Fast-Wake, while some components are powered down, the PMA and PMD remain active, continually transmitting signals between transceiver and receiver, hence maintaining synchronization. This allows for a link wake-up in a few hundred nanoseconds, rather than microseconds. Specifically, studies show that the link could wake up from Fast-Wake mode in 250 ns to 500 ns, with power savings of 20–40%, compared to an active link [11]. Recent specifications show that Deep-Sleep may not be currently compatible with OTN based optical networks,[4] however, the standards expect to incorporate Deep-Sleep in these devices in the future.

Figure 1 shows an example of a link that uses both Deep-Sleep and Fast-Wake. Here in Figure 1a, after an active period, the link remains on, consuming full (or 100%) power, until the Stall-Timer expires (Stall-to-Shallow). Following the expiry of the Stall-to-Shallow timer, the link switches to Shallow-Sleep, where it consumes 60% energy (as mentioned above). The link remains in Shallow-Sleep until the global Stall-Timer expires where it drops down to Deep-Sleep, consuming 10% power. The arrival of a frame during Deep-Sleep triggers a full wake-up, after which the link transmits the frame. Figure 1b, shows a case where a frame arrives during the Shallow-Sleep period, where a Fast-Wake is used to quickly

[2]The Stall-Timer essentially *stalls* the link from going to sleep immediately after a frame transfer.

[3]Physical Medium Attachment and Physical Medium Dependent sublayers.

[4]Optical Transport Network.

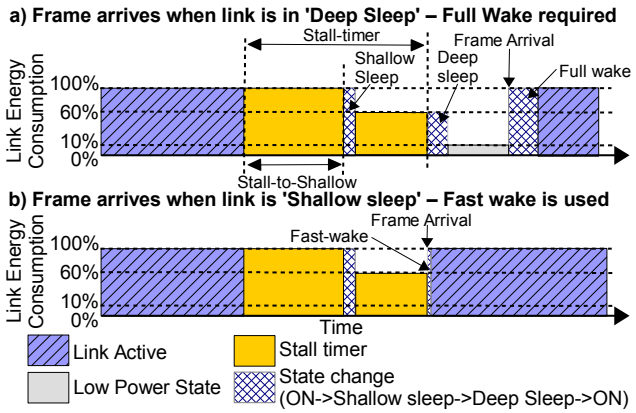

Fig. 1: **Example timeline illustrating various low power and Stall-Timer states of an EEE link**

power up the link for frame transmission. Note that designers of EEE in switches have the option of only using Fast-Wake or Deep-Sleep, or a combination of both as we show in Figure 1.

Every year, the Ethernet family of interconnects is deployed in a large share of systems (about 40%) in the TOP500. The popularity of Ethernet in HPC, coupled with the need for energy proportionality, makes a strong case for understanding and using the inherent power-savings techniques available in these devices, while ensuring low performance overheads. We believe that the insights presented in this paper could help vendors in designing EEE technology for HPC and could inform the upcoming standardization efforts that introduce EEE for 40/100/400Gb links.

B. *Related Work*

Here we present related work on energy efficient interconnects. Saravanan, et al., [7] present analysis of Energy Efficient Ethernet for HPC workloads. They show that increase in latencies due to link wake-up could be harmful to certain HPC applications. They propose the use of a Stall-Timer to have their links remain on but idle to reduce overheads. This work lead to PerfBound, an mechanism to automatically find the Deep-Sleep Stall-Timer, subject to a performance overhead bound [8]. This work however does not consider multiple sleep modes. Jian Li, et al., [6] discuss on/off networks that use snoop messages that arrive at the NICs as an indication of an impending message. In nodes that have snoop-based coherence, snooping messages would arrive at the link before an impending message, which could be used to trigger the link on, before the actual arrival of the message. They also propose the use of an always-on control network that sends control signals through the routing path of a message to wake up subsequent links. They further propose software enhancements which would have programmers annotate the code signalling an impending message. Similarly, Soteriou, et al., [12] show that on/off networks incur a large performance penalty and hence, they propose software mechanisms such as parallelizing compilers for network power savings.

Gupta, et al., in their work [16], show that opportunistic sleeping of links is possible, but their technique increases the mean latency. Vassos, et al., [13] discusses a design space analysis for on/off based links. They propose using multiple routing paths available in torus like networks to shut down parts of the network during periods of low load. They evaluate their

proposal with message arrivals following a Poisson process. Similarly, Alonso, et al., [15] propose shutting down redundant links (sub-trees) in their fat-tree system to save energy. These proposals do not discuss the performance impact of bursty communications that are typical of HPC. Similarly Ethernet evaluation reports [9, 10] also use synthetic benchmarks to evaluate on/off networks. Relevant work on Energy Efficient Ethernet [2, 9, 10, 18, 19, 20] provide detailed evaluations on EEE for its potential for desktop and IT based systems, but they do not target HPC workloads. Totoni, et al., [21] show that not all links of a network executing an HPC application are utilized, so they propose runtime techniques to find links in the network that are never utilized, in order to turn them off. Their work, however, does not adaptively turn on/off links.

D. Abts et al., [5] proposed energy proportional interconnects based on reducing the link rates of aggregated links. In their approach, during periods of inactivity, link rates are reduced to a lower link bandwidth to save energy. Work by Kim, et al., [17] evaluate energy proportional networks and compare links based on dynamic voltage scaling and on/off links. They show that dynamic voltage scaling in links causes a significant increase in latency and show that on/off based techniques perform comparatively better. The difference between this work and the above proposals is that we focus on HPC workloads and specifically multiple sleep modes. The novelty of our work is that to the best of our knowledge we are the first to analyze the recently proposed Fast-Wake (or an intermediate sleep state) in the context of HPC.

III. Exploring Fast-Wake

This section presents our analysis of Fast-Wake on HPC applications. We present, below, our experimental methodology, followed by a comparative analysis of Fast-Wake and Deep-Sleep. We make the case for a combined use of both Fast-Wake and Deep-Sleep. We analyze the various parameters, specifically, other energy consumption levels, wake-up time and power/performance behavior on higher network layers.

A. Methodology

For our experiments, we used an extension of the Dimemas cluster simulator, which has been found to be accurate to within 10% and validated against production supercomputers, including Blue Gene/L, P, Q, and three generations of MareNostrum [22, 23, 24].We modified the network model to support a hierarchical network, with on/off links as specified by the Energy Efficient Ethernet protocol. The simulation infrastructure is driven by traces, which record CPU intervals and MPI events, independent of the network configuration, measured from a real execution on MareNostrum. The CPU intervals are scaled by relative CPU performance. MPI events imply dependencies, which ensure correctness. Link energy consumption is modeled as 100% when "on" or during transition between on/off states, 60% in Fast-Wake mode and 10% in Deep-Sleep mode.[5] All energy figures are normalized to a percentage of the original energy-to-solution.

The simulator is configured to model a cluster with a three-level hierarchical network. Applications are executed on 64, 128 or 256 nodes, grouped into 8, 16, or 32 nodes per rack, respectively, forming eight racks in total. Nodes are connected to the top-of-rack switch, which is in-turn connected to a two-level, four-node 2-ary fat tree. The network is statically routed, cut-through flow-control with fully duplex links. Each node is a two-socket high-end CPU with 225 GFlops (based on TOP500 machines with two Intel Xeon sockets). The switch latency is configured at 320 ns for the first hop and 80 ns for subsequent hops. Edge links are configured at 20 Gb/s, while the higher two levels are 40 Gb/s and 100 Gb/s respectively. The two directions of the full-duplex links can be turned on and off separately. We use the wake-up and sleep times of 4.16 µs and 3.88 µs for Energy Efficient Ethernet [2, 9] and 250 ns for Fast-Wake [11].

We used fifteen HPC applications. The original traces were large, on the order of hundreds of gigabytes, so simulation was done for a few iterations of the outer loop. The traces for ALYA [25], LINPACK [30], BT [26], CG [26], FT [26] and MG [26] were executed on 256 nodes, QUANTUM [34], WRF [35] MILC [31], GROMACS [29] and GADGET [28] on 128 nodes, and NAMD [32], PEPC [33], SP and LU [26] on 64 nodes.

B. Approach to investigating Fast-Wake in HPC

Before presenting our analysis of Fast-Wake, we first define the mechanisms and the specific terminology we use in the subsequent sections below,

1) **Deep-Sleep**: We define a link to be in Deep-Sleep when it powers down to consume 10% energy compared to when the link is active. We interchangeably use the terms Deep-Sleep and Deep-Sleep mode to either mean the power saving mechanism or the state of the link, depending on the context. With Deep-Sleep mode, we define *Stall-to-Deep*, where the link remains on (at 100% power) before the the link switches to Deep-Sleep. Note that a *Full-Wake* (of 4.18 µs) is required to wake-up from Deep-Sleep.
2) **Fast-Wake**: In Fast-Wake mode, the link behaves exactly as Deep-Sleep except that the link wake-up requires a few hundred nanoseconds as opposed to Deep-Sleep's few microsecond wake-up. With Fast-Wake, the link only powers down to 60% compared to when the link is active. Here we define *Stall-to-Shallow*, where the link remains on (at 100% power), before switching to *Shallow-Sleep* (at 60% power). Note that a frame arrival during Shallow-Sleep would require Fast-Wake before transmission, whereas a frame arrival during Deep-Sleep would require Full-Wake before transmission.
3) **Deep-Sleep + Fast Wake**: This mode which represents a combination of Deep-Sleep and Fast-Wake as shown in Figure 1. Here, when the link becomes idle, the link remains on until Stall-to-Shallow, after which the link enters Shallow-Sleep. At Shallow-Sleep, the link waits for Stall-to-Deep timer to expire before entering into Deep-Sleep. Here, it is important to note that both Stall-to-Deep and Stall-to-Shallow start as soon as the link goes idle and for this reason Stall-to-Deep must always be greater than or equal to the Stall-to-Shallow timer.

With respect to the above, our approach towards investigating Fast-Wake is as follows. We first compare the

[5] Section III-F evaluates the energy savings for differing energy consumptions in the Fast-Wake mode.

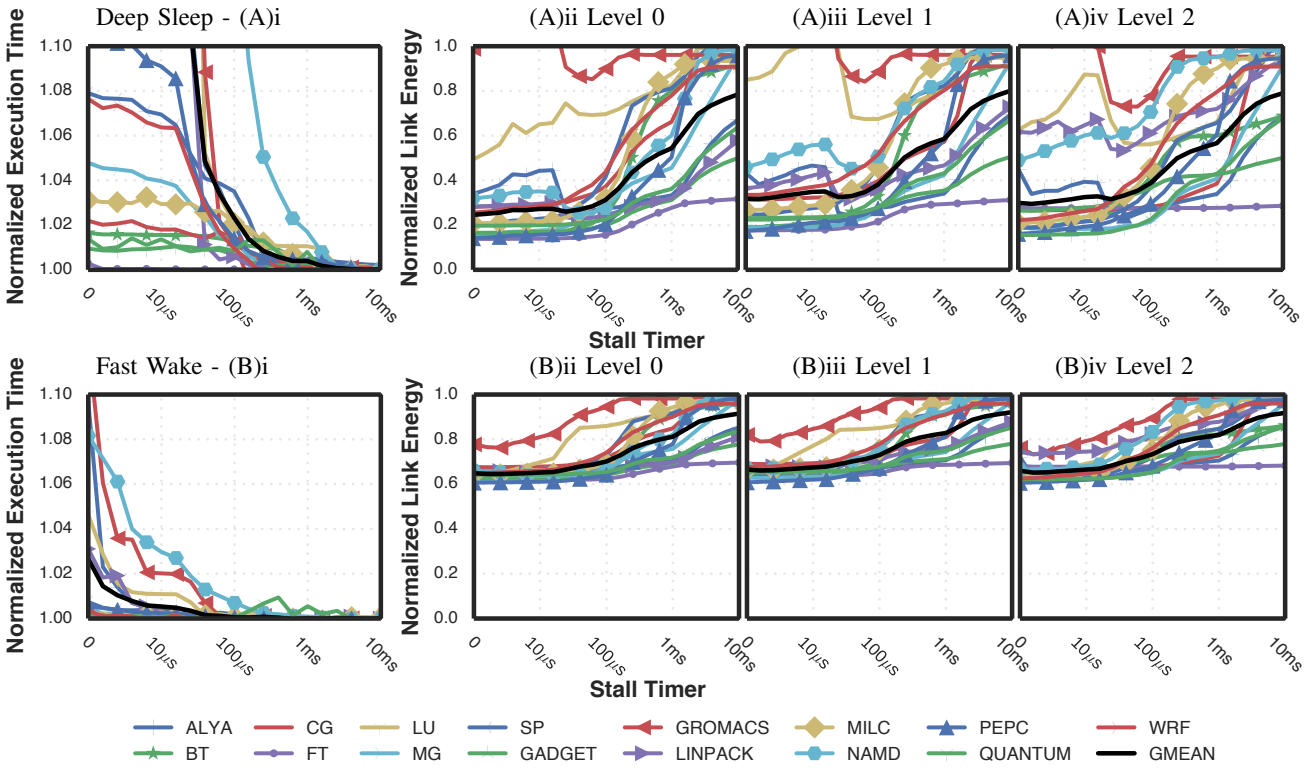

Fig. 2: **Power and Performance of using Deep-Sleep and Fast-Wake as a function of their Stall-Timers**

individual power and performance characteristics of Fast-Wake and Deep-Sleep. Based on which, we present a case for a combined approach that uses both Fast-Wake and Deep-Sleep. We further investigate the following parameters: 1. Energy consumption during Fast-Wake mode and 2. Link wake-up duration for Fast-Wake. We discuss how obtaining optimal power and performance trade-offs requires careful adjustment of respective Stall-timers. Note that in our experiments, Stall-Timers are constant across all links of the network, since investigating optimal Stall-Timers for every link in the network is not feasible for the sensitivity analysis presented below.

C. Deep-Sleep and Fast-Wake vs Stall-time

Figure 2(A) presents an analysis of the performance and energy consumption using only the Deep-Sleep mechanism, as a function of its corresponding Stall-Timer (Stall-to-Deep). Specifically, Figure 2(A)(i), shows the normalized execution time for each application referred to in the previous section, relative to the same with an always-on interconnect. Figures 2(A)(ii), (iii) and (iv) are similar in that they show the average energy consumption of links at each level of the network, where level-0 is connected to nodes and level-2 is the highest.

Varying Stall-to-Deep and Stall-to-Shallow demonstrates interesting trade-offs between power and performance, as seen in Figures 2(A) and (B), respectively. Figure 2(A)(i) shows that, for all applications, the performance overhead reduces as the Stall-Time increases. This is because, as the Stall-Timer increases to infinity, the network tends towards being always-on. This means that the Stall-Timer would eventually be larger than all idle periods between frame arrivals, and hence no subsequent frame arrival would ever incur a wake-up delay, rendering zero performance overhead. This however increases the energy consumption in all levels of the network, since an always-on network consumes 100% power relative to its baseline. In Figure 2(A), we see that lower values of the Stall-Timer (0 to 100 µs) have relatively large performance overheads (greater than 10%) but high energy savings (70% on average). For higher values of the Stall-Timer (100 µs to 1 ms), the performance overhead drops to below 1%, while the energy savings are still about 50% to 60% (average), for all levels of the network.

Figure 2(B) shows a similar analysis, but for Fast-Wake. Here, the Stall-Timer represents Stall-to-Shallow, where the link drops from 100% power to 60% power, and requires a Fast-Wake to wake-up. Figure 2(B)(i) is similar to Figure 2(A)(i) in that performance overheads decrease with an increase in the Stall-Timer. We see, however, in Figure 2(B)(i), that the performance overhead drops below 1% (say) at a much lower value of the Stall-Timer compared with Figure 2(A)(i). Correspondingly the energy consumption as shown in Figure 2(B) begins at 60% and increases towards 100%. The link energy consumption of applications in Figures 2(B)(ii), (iii) and (iv) start at 60% since the idle power of Fast-Wake is 60% of that of an active link.

On comparing the two mechanisms for application NAMD, for example, we find that for a performance overhead of less than 1% with Deep-Sleep (Figure 2(A)), a Stall-Timer larger than 1 ms is required, for which the energy consumption is about 85–90% (for Level-0 links). With Fast-Wake, as seen in Figure 2(B), we see that application NAMD falls below 1% at about 100 µs, where the energy consumption is about 65–70%, for the same Level-0 links. We also see similar behavior for higher network level links. Clearly for NAMD, Fast-Wake is

the better mechanism. In contrast to the above, if we consider application PEPC, we see that PEPC requires a Stall-Timer of about 100 µs for an overhead of less than 1%, as shown in Figure 2(A), but it only consumes 20% link energy. We see with Figure 2(B) however, that PEPC requires a Stall-Timer of less than 10 µs for overheads less than 1%, but its energy consumption is now above 60%. In the case of PEPC we therefore see that Deep-Sleep is the better mechanism for energy savings and low performance overhead.

Figures 2(A) and (B) show that, for both Deep-Sleep and Fast-Wake, adjusting the Stall-Timer varies power and performance, through an increase in energy savings at the expense of performance, or vice-versa. It is clear that since Fast-Wake powers back the link at a faster rate, relatively small Stall-Timers are required for low performance overheads. However, with Deep-Sleep, we see that larger Stall-Timers are required to reduce performance overheads. It is interesting to see that for a fixed performance overhead point (1% say), some applications such as NAMD, GROMACS, SP and MILC tend to have better energy savings with Fast-Wake while others such as GADGET, FT, QUANTUM and PEPC are better with Deep-Sleep. Specifically, applications that have low network utilization and that are not latency sensitive are better with Deep-Sleep compared to Fast-Wake. These applications do not require the faster wake-up time, so they can benefit from Deep-Sleep's lower energy consumption.

D. Pareto optimal analysis of Fast-Wake and Deep-Sleep

The analysis with Figures 2(A) and (B) clearly shows the need for a combined approach. As shown in Figure 1 using both Deep-Sleep and Fast-Wake would require adjusting two Stall-Timers: Stall-to-Deep, which expires to Deep-Sleep and Stall-to-Shallow, which expires to Shallow-Sleep. Together, these two Stall-Timers create a 2-dimensional search space.

We present an exhaustive search of this 2D search space in Figure 3. The x-axis is the performance overhead, up to 5%, as larger overheads would be unacceptable. The y-axis is the energy consumption of all links in Level-0 of the network (the behavior of higher network layers is discussed in the next section). The scatter points labeled "Deep Sleep + Fast Wake + Stall timer sweep" show all combinations found by varying both Stall-Timer values. The curve labeled "Deep Sleep + Fast Wake + Stall timer - Pareto Optimal" is the Pareto-optimal curve derived from only the points that are Pareto-optimal (which is roughly speaking only the points with lowest energy consumption for a given performance overhead). The other curve, labeled "Deep Sleep + Stall Timer", presents, for comparison, the power and performance obtained from varying Stall-to-Deep using only the Deep-Sleep mode. We compare our approach with Deep-Sleep because it represents the older approach to energy savings available in EEE switches before the introduction of Fast-Wake. This figure therefore presents the potential benefits of using both Fast-Wake and Deep-Sleep.

Figure 3 makes a strong case for using a combination of Deep-Sleep and Fast-Wake, since the Pareto-optimal curve is clearly below or overlapping the Deep-Sleep curve for all applications. We see that this technique is less useful for applications that are not latency sensitive as in the case of FT, QUANTUM and GADGET, where simply using Deep-Sleep is sufficient. However, with applications BT, CG, LU, SP,

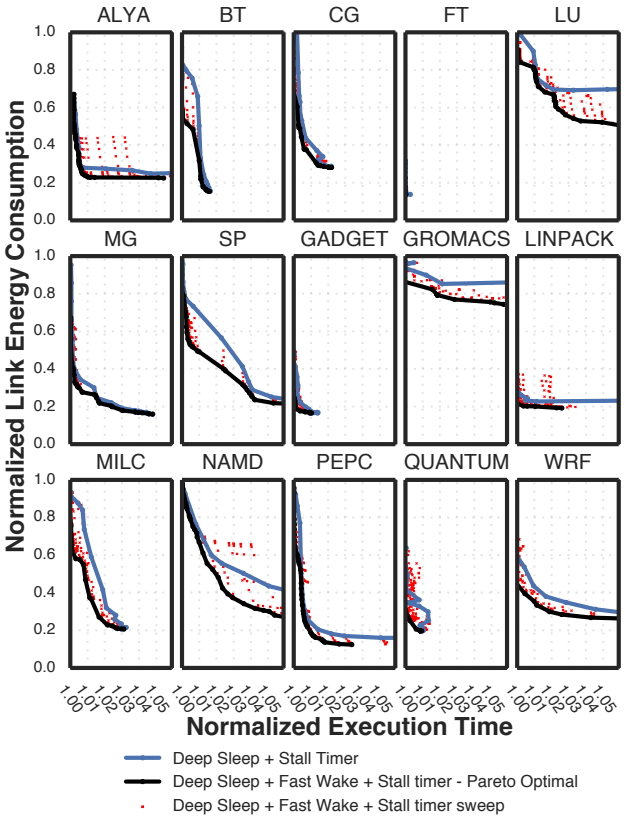

Fig. 3: **Pareto-optimal analysis of average (level-0) link energy and performance using both Deep-Sleep and Fast-Wake along with their corresponding Stall-Timers compared to only using Deep-Sleep and its Stall-Timer**

GROMACS, MILC, NAMD, PEPC and WRF, a combined approach clearly improves energy and performance overhead. We see with BT at 0.5% performance overhead has 45% energy savings with the Pareto optimal curve compared to only 20% with Deep-Sleep. Applications such as SP and MILC benefit most from the combined approach at low performance overheads, where a further reduction of performance overhead is possible at the same or similar energy savings. With GROMACS and WRF we see 5%–20% energy savings compared with all points of Deep-Sleep.

It is to be noted however that here we compare a Pareto-optimal curve, with performance and energy values obtained from simply adjusting the Stall-Timer of Deep-Sleep. The scatter points in Figure 3 clearly show that for many combinations of Stall-to-Shallow and Stall-to-Deep, higher performance overhead or much lower energy savings are obtained. Specifically for ALYA, choosing an incorrect combination of Stall-Timers values is highly likely since most other values perform worse compared to Deep-Sleep. Dynamically and automatically choosing these Stall-Timers for Fast-Wake and Deep-Sleep during runtime could be an interesting problem for further research.

E. Fast-Wake on higher level links - L1, L2

In Figure 4 and all following figures, due to constraints on the number of pages, we present a subset of the applications seen in Figure 3. Specifically, we show applications BT, SP, GROMACS, MILC and NAMD. We chose these applications

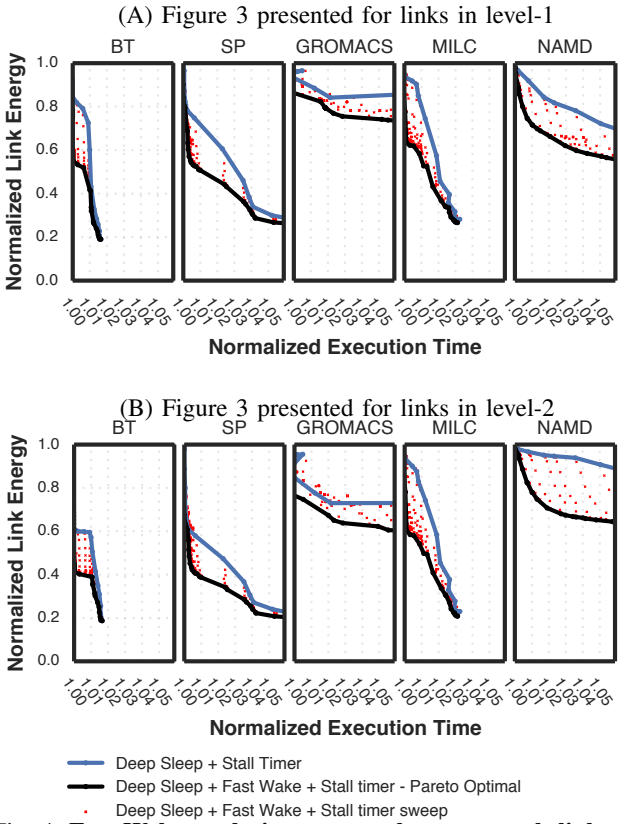

Fig. 4: **Fast-Wake analysis on upper layer network links**

because all other applications either behaved similarly, or, for FT, GADGET and QUANTUM, they do not benefit from Fast-Wake. Figure 4 is the same as Figure 3, except that it shows the normalized energy consumption of network Level-1 for Figure 4(A) and network Level-2 (highest) for Figure 4(B). Comparing Figures 4(A) and (B) with Figure 3 shows that the benefits of Fast-Wake plus Deep-Sleep at the higher levels of the network are similar to those at Level-0. NAMD particularly has increasingly better trade-offs with higher link levels compared to Deep-Sleep. Since NAMD utilizes its network sporadically rather than in bursts, we see higher benefits from Fast-Wake. At higher levels of the network, where the traffic is sporadic rather than uniform, large Stall-Timers for Deep-Sleep are required to maintain acceptable overheads. Having an intermediate state between these large Stall-Timers produces the better energy trade-off.

F. Fast-Wake energy ratio - 40/60/80% and Fast-Wake timing analysis - 500 ns

This section investigates how the conclusions in this paper would change, when varying the power consumption during Shallow-Sleep and its wake-up time. The power consumption during Shallow-Sleep has been shown to be 60% compared to when the link is active [11], but the precise value cannot be known for sure until products arrive on the market. Figures 5(A) and (B), are similar to Figure 3 in that they have a Fast-Wake time of 250 ns, and they show the energy and performance trade-offs for Level-0 links. However, unlike Figure 3, Figures 5(A) and (B) model the power during Fast-Wake to be 40% and 80%, respectively, compared to when the link is active. In Figure 5(A), where Shallow-Sleep only consumes 40% energy, we see higher energy to performance trade-

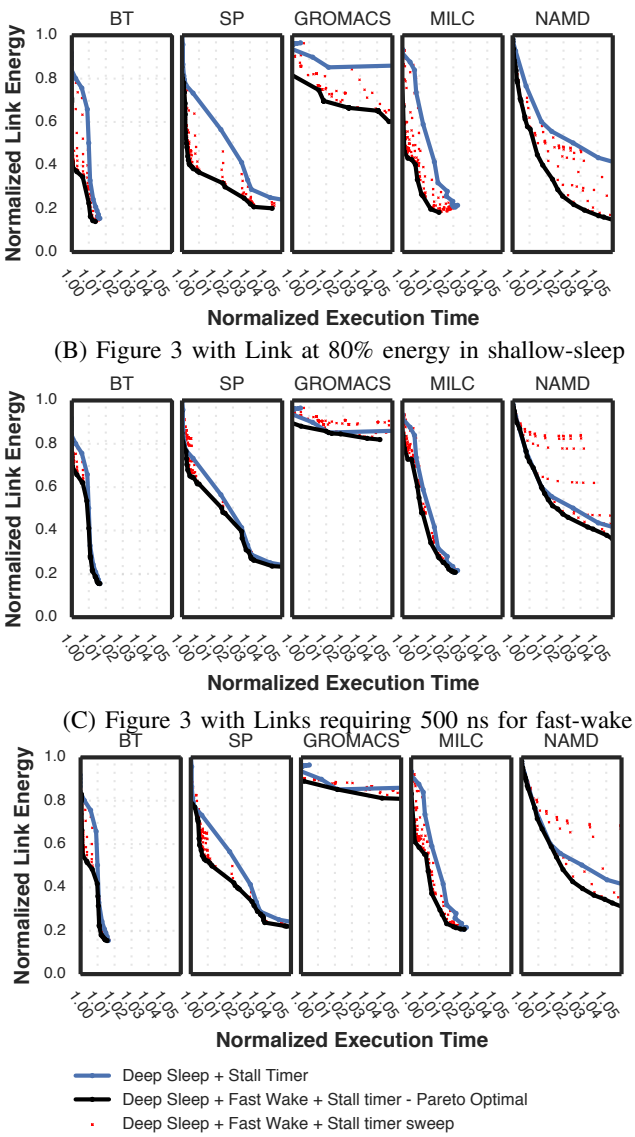

Fig. 5: **Fast-Wake analysis on link energy consumption and link wake-up time**

offs compared to Shallow-Sleep with 60% shown in Figure 3. Note that for a given performance overhead of 1%, the link energy savings for GROMACS increases from 15% (with 60% Shallow-Sleep as seen in Figure 3) to about 25%. These benefits indicate that a number of points in the Pareto-optimal curve have Stall-Timer values configured so that the link spends significant time in Shallow-Sleep. Consuming lower energy during Fast-Wake's Shallow-Sleep therefore contributes to significantly higher link energy savings. In Figure 5(B), however, where the power consumption in Shallow-Sleep is 80%, we see that the curve is similar to that of Deep-Sleep mode.

In Figure 5(C) the Shallow-Sleep energy consumption is modeled back to 60%, but we model the Fast-Wake time to be 500 ns, instead of 250 ns (used in all previous figures). We also investigated Fast-Wake to be 1 µs and 2 µs. As expected, our results with other values of Fast-Wake time show that the gap between the two curves of our Pareto-

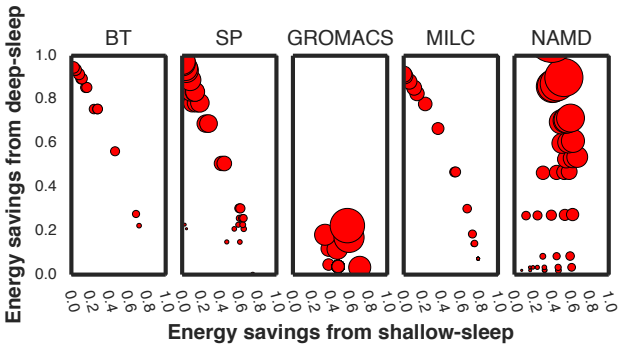

Fig. 6: **Contribution of energy savings with Pareto-optimal points from Figure 3 between Shallow-Sleep and Deep-Sleep**

optimality study, reduces, as the Fast-Wake time increases. Since Deep-Sleep requires $4.16\,\mu s$, increasing Fast-Wake time towards $4.16\,\mu s$, tends to make Fast-Wake closer to that of Deep-Sleep. However, between $250\,ns$ and $500\,ns$, we find a small reduction in energy savings or performance compared to Figure 3. It is conceivable that a higher wake-up time for Fast-Wake could allow for more components to be turned off, further reducing its energy consumption.

G. Ratio of energy savings between Fast-Wake and Deep-Sleep

In Figure 6, we show the proportion of energy savings from Fast-Wake and Deep-Sleep respectively. The figure shows a scatter plot of points obtained from the Pareto-optimal curve in Figure 3 for each corresponding application. In the x-axis, we show the proportion of energy savings obtained while the application is in Shallow-Sleep and in the y-axis, we show the proportion in Deep-Sleep. The size of the scatter point is proportional to its performance overhead, where larger points represent a higher overhead.

Figure 6, application NAMD, for example, shows that the higher the energy savings from Deep-Sleep, the higher the performance overhead. The size of the scatter points reduces with the reduction in overhead, as the contribution of energy savings from Deep-Sleep reduces. Reduced energy savings from Deep-Sleep also means that the total interconnect energy increases, due to higher Stall-to-Deep timer values. We see the same behavior with Shallow-Sleep where overhead correspondingly reduces with a reduction in energy savings from Shallow-Sleep. It is interesting to see that Stall-Timer values correspond to points in Figure 6. Choosing the correct Stall-Timers is critical to maximizing energy savings and reducing overheads.

IV. Conclusions

This paper presents an analysis of Fast-Wake for HPC workloads, examining its potential for energy savings and possible performance overheads. Our analysis showed that using both Fast-Wake and Deep-Sleep offers higher energy savings in links with lower average performance overhead than their use as standalone mechanisms. We showed that for optimal energy and performance, the corresponding Stall-Timers of Fast-Wake and Deep-Sleep must be chosen carefully. We also show that higher network layers can benefit more from the use of Fast-Wake with Deep-Sleep compared to edge links. With the ratification of Fast-Wake in March 2014 and 2015 for 40/100Gb Ethernet, and the ongoing standardization effort for 400Gb Ethernet, this analysis could could help interconnect vendors to build energy efficiency mechanisms that target HPC.

V. Acknowledgments

This research was supported by the Ministry of Economy and Competitiveness of Spain under the contract TIN2012-34557, HiPEAC-3 Network of Excellence (ICT-287759), European Union's $7th$ Framework Programme [FP7/2007-2013] under project Mont-Blanc (288777), Generalitat de Catalunya (FI-AGAUR 2012 FI B 00644) and finally the Severo Ochoa Program (SEV-2011-00067) of the Spanish Government.

References

[1] IEEE Draft P802. 3az/D2. 3, Energy Efficient Ethernet, 2010.
[2] Active/Idle Toggling with Low Power Idle, IEEE 802.3az.
[3] DARPA. Ubiquitous High Performance Computing (UHPC) Broad Agency Announcement (BAA). 2010.
[4] Growth in Data center electricity use 2005 to 2010. Analytics Press. (http://www.analyticspress.com/datacenters.html)
[5] D Abts et al. Energy Proportional Datacenter Networks. In *Proc. International Symposium on Computer Architecture (ISCA)*, 2010.
[6] Li Jian et al. Power shifting in Thrifty Interconnection Network. In *Proc. High Performance Computer Architecture (HPCA)*, 2011.
[7] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, Power/Performance Evaluation of Energy Efficient Ethernet (EEE) for High Performance Computing. In *Proc. 2013 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*.
[8] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, A performance perspective on energy efficient HPC links. In *Proc. 28th ACM International Conference on Supercomputing (ICS '14)*
[9] Christensen Ken et al. IEEE 802.3az: the road to energy efficient ethernet. *Comm. Mag.*, 48(11):50–56, nov 2010.
[10] Reviriego P. et al. Performance evaluation of energy efficient ethernet. *Comm. Letters.*, 13(9):697–699, September 2009.
[11] Hugh Barrass, Options for EEE in 100G IEEE 802.3bj
[12] Vassos Soteriou et al. Software-directed power-aware interconnection networks. *TACO*, 2007.
[13] Soteriou et al. Design-space exploration of power-aware on/off interconnection networks. *International Conference on Computer Design*, 2004.
[14] V Soteriou et al. Dynamic power management for power optimization of interconnection networks using on/off links. In *Proc. IEEE High Performance Interconnects*, 2003.
[15] Alonso Marina et al. Dynamic power saving in fat-tree interconnection networks using on/off links. IPDPS, 2006.
[16] Maruti Gupta et al. Dynamic ethernet link shutdown for energy conservation on ethernet links. In *ICC'07*.
[17] Eun Jung Kim et al. Energy optimization techniques in cluster interconnects. In *ISLPED*, 2003.
[18] Chamara Gunaratne et al. Ethernet adaptive link rate (alr): Analysis of a buffer threshold policy. In *GLOBECOM*, 2006.
[19] Blanquicet Francisco et al. An Initial Performance Evaluation of Rapid PHY Selection (RPS) for Energy Efficient Ethernet. Local Computer Networks (LCN), 2007.
[20] Koibuchi M. et al. An on/off link activation method for low-power ethernet in PC clusters. *IPDPS*, 2009.
[21] Totoni et al. Toward Runtime Power Management of Exascale Networks by On/Off Control of Links. In *IPDPS-W'13*.
[22] Rosa M. Badia et al. Dimemas: Predicting MPI applications behaviour in Grid environments. GGF8 Workshop, 2003.
[23] Sergi Girona et al. Validation of dimemas communication model for mpi collective operations. In *EuroPVM/MPI'00*.
[24] J. Gonzalez et al. Simulating whole supercomputer applications. In *Micro, IEEE*, 2011.
[25] Alya Red:Computational Biomechanics for Supercomputers.
[26] NAS parallel benchmarks. Available online at http://www.nas.nasa.gov/publications/npb.html
[27] D. Marx and J. Hutter. Ab-initio Molecular Dynamics: Theory and Implementation, NIC. Forschungszentrum Jlich, 2000.
[28] V. Springel. The cosmological simulation code GADGET-2. Royal Astronomical Society 2005.
[29] Hess et al. GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of chemical theory and computation*, 2008.
[30] D. Teresa et al. High performance linpack benchmark: a fault tolerant implementation without checkpointing. In *Proc. ACM ICS '11*
[31] MIMD Lattice Computation (MILC) Collaboration.
[32] R. K. Brunner et al. Scalable Molecular Dynamics for Large Biomolecular Systems. In *Supercomputing Conference*, 2000.
[33] PEPC: Pretty Efficient Parallel Coulomb-solve, Interner Bericht Zentralinstitut fur Angewandte Mathematik.
[34] QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials.
[35] Michalakes et al. The Weather Reseach and Forecast Model: Software Architecture and Performance. In *ECMWF*, 2004.