Evaluating the Impact of Energy Efficient Networks on HPC Workloads

Giorgis Georgakoudis[†], Nikhil Jain[*], Takatsugu Ono[‡], Koji Inoue[¶], Shinobu Miwa[||], Abhinav Bhatele[†,§]

[†]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory
[*]NVIDIA, Inc.
[‡]Department of Advanced Information Technology, Kyushu University
[¶]Department of Informatics, Kyushu University
[||]Department of Information System Fundamentals, The University of Electro-Communications
[§]Department of Computer Science, University of Maryland, College Park
E-mail: georgakoudis1@llnl.gov, nikhijain@nvidia.com, takatsugu.ono@cpc.ait.kyushu-u.ac.jp, inoue@ait.kyushu-u.ac.jp, miwa@hpc.is.uec.ac.jp, bhatele@cs.umd.edu

***Abstract*—Interconnection networks grow larger as supercomputers include more nodes and require higher bandwidth for performance. This scaling significantly increases the fraction of power consumed by the network, by increasing the number of network components (links and switches). Typically, network links consume power continuously once they are turned on. However, recent proposals for energy efficient interconnects have introduced low-power operation modes for periods when network links are idle. Low-power operation can increase messaging time when switching a link from low-power to active operation.**

We extend the TraceR-CODES network simulator for power modeling to evaluate the impact of energy efficient networking on power and performance. Our evaluation presents the first study on both single-job and multi-job execution to realistically simulate power consumption and performance under congestion for a large-scale HPC network. Results on several workloads consisting of HPC proxy applications show that single-job and multi-job execution favor different modes of low power operation to have significant power savings at the cost of minimal performance degradation.

***Index Terms*—Network simulations; Energy-efficient networks; Power and performance modeling; Multi-job workloads**

I. Motivation

Supercomputer networks scale by adding more switches (routers) and links to connect more compute nodes together and increase bandwidth. Popular network topologies [1], [2] deploy high-radix routers to increase the number of links for larger systems and higher bandwidth. Presently, network links consume full power even when idling, i.e., when no packets are being transmitted, thus wasting power during the computational phases of applications. By contrast, other system components, such as CPU and memory, conserve power when underutilized. Reportedly [3], interconnection networks consume up to 12% of the total system power, when the system is fully utilized, and this percentage becomes larger in periods of low system utilization. Reducing the power consumed by the network can reduce the overall power budget for more efficient HPC, by not only saving network power but also by enabling spending extra power for computational capacity.

Existing proposals for improving power and energy efficiency include adaptive link rate (ALR) [3], [4], [5] and low-power idle (LPI) modes [6], [7]. Both proposals incur a transmission time penalty, either due to reducing link speed, as in ALR, or due to transitioning the link to a low-power mode. LPI is typically an improvement over ALR because it incurs less delay (microseconds vs. milliseconds) and offers maximal power savings – up to 90% less power consumption than fully active mode. LPI saves power by shutting down transceiver components when a link is idling, whereas ALR still consumes power during idle periods, albeit at a reduced rate by reducing the link speed. Because of those advantages, LPI has been ratified by the Ethernet consortium [8] as Energy-Efficient Ethernet. In particular, LPI defines different low-power modes that expose different trade-off points for power saving and impact on performance. Specifically, a link can enter *deep-sleep* operation when idle, saving maximum power but adding an extended wake-up delay on packet transmission. In contrast, there is a *fast-wake* operating mode, during which the links stays partially active, thus saving less power than deep-sleep, but has modest wake-up delay. We provide details on low-power modes in the background section (Section II).

Recent studies [9], [10], [11], [12], [13] on LPI have shown great potential for power savings, up to 70%, for a negligible degradation in performance (less than 1%). However, those results come from simulating single-job workloads on a small cluster of 256 nodes with a fat-tree network topology. In practice, large supercomputers comprise of thousands of nodes, and execute multiple jobs, which share the network. Different installations also deploy different network topologies besides fat-tree, such as dragonfly [14] and HyperX [15]. The results and insights of previous work are limited due to missing these important variations.

In this paper, we extend previous studies on energy-efficient networks by simulating large-scale supercomputer networks executing realistic, multi-job workloads. We extend the state-of-the-art network simulator TraceR-CODES with power consumption modeling and experiment with three different network topologies: dragonfly, fat-tree, and HyperX. We use HPC proxy applications to create representative, multi-job workloads running on HPC systems.

Specifically, the paper makes the following contributions:

- A power model of link operation that is configurable for timing parameters, when transitioning between low-power and active modes, and for power consumption. The model and methodology are based on existing proposals for low-power operation [8], [12] and are flexible to model different link technologies.
- The design and implementation of this power model in the TraceR-CODES network simulator. We choose TraceR-CODES to leverage its extensive library of validated network models for simulating communication, on top of which we add our power models.
- An extensive evaluation of low-power modes of link operation by using three different topologies (dragonfly, fat-tree, and HyperX), simulating a large supercomputer network of 4,608 compute nodes and deploying single-job and multi-job workloads comprised of HPC proxy applications. Results of single-job workloads validate our power model by comparing them with existing literature.
- New insights from the results of multi-job simulations, which show that single-job and multi-job execution have different power consumption and performance characteristics when deployed on energy-efficient networks.

II. Energy-Efficient Link Operation

This section provides an overview on low-power operation of network links. It introduces the two proposed low-power modes – *fast-wake* and *deep-sleep*. Notably, these modes are ratified in the IEEE Ethernet Standard [8], proposed by the Energy Efficient Ethernet (EEE) working group.

Network links operate always-on to maintain synchronization between the transmitter and receiver endpoints of the link. This approach keeps the link in a ready-to-transmit state but wastes power and energy during idle periods when there is no communication. Specifically, transmitter and receiver components of the switches at the endpoints of a link signal continuously, thus consume full power, all the time, even if no packet is transmitted. Recently, low-power link operation during idle periods has been proposed to address this inefficiency. In low-power mode, links turn off parts of the transmitter and receiver components on opposite sides of the link to reduce power consumption. In this mode, links are unavailable to transmit or receive packets and resuming active operation incurs a delay to wake up the link, re-activate transmitting and receiving components, and ramp up to full power.

This *wake-up delay* adds overhead to packet transmission when transitioning from the low-power to active mode. The duration of the delay depends on the physical and electrical characteristics of the links and transceivers. Existing proposals define two modes for low-power operation, namely *fast-wake* (FW) and *deep-sleep* (DS) with different wake-up delay and power consumption characteristics. Fast-wake has a lower wake-up delay for modest power savings, by keeping only the transmitter component on and allowing the receiver to partially shut down, for maintaining link synchronization. In contrast, deep-sleep provides maximal power saving by shutting down both transmitter and receiver components, incurring extra delay to wake-up.

TABLE I: Wake-up delays and packet transmission latency (in µs) for deep-sleep and fast-wake

Link speed	$T_{w\|DS}$	$T_{w\|FW}$	T_x Latency MTU 1500B
25GBASE-R	14.25	0.34	0.48
40GBASE-R	5.50	0.34	0.30
100GBASE-R	5.50	0.34	0.12

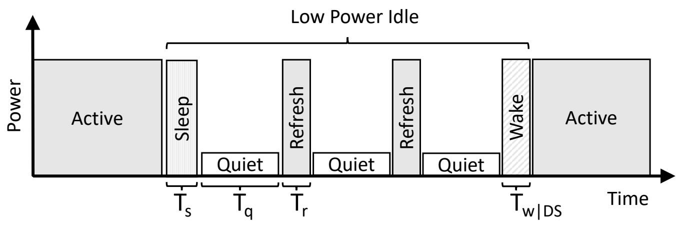


Fig. 1: Overview of the deep-sleep operation

The motivation for different low-power modes comes from the observation that the wake-up delay becomes a significant overhead as link speeds increase. Table I shows wake-up delays for both fast-wake and deep-sleep, and packet transmission latency for different speeds of backplane links, using data from the Ethernet standard. By design, fast-wake is intended as a compromise between deep-sleep and active operation. The wake-up delay of deep-sleep is an order of magnitude higher than transmit latency, whereas fast-wake delay is comparable to that. The following sections elaborate more on the operation and power characteristics of fast-wake and deep-sleep modes, and a hybrid mode combining the two.

A. Deep-sleep operation

Figure 1 shows an overview of the deep-sleep operation. The link enters low-power mode starting with a sleep signaling phase of T_s duration, during which the link operates at peak power. In that phase, the transmitter sends a sleep signal to the receiver and ceases operation after T_s. Next, the link enters quiet operation, where both the transmitter and receiver have shut down. In addition, the link periodically enters a refresh signaling phase, consuming full power, to maintain synchronization at the physical layer. In the figure, the quiet period is defined as T_q, while the duration of refresh is T_r. The length of quiet and refresh periods depends on the technology characteristics of the link. When the link needs to transition to active operation for transmitting a packet, there is a wake-up delay, denoted as $T_{w|DS}$, before resuming active operation.

By present standards and technology, T_s, T_r, and $T_{w|DS}$ are on the order of microseconds, whereas T_q is on the order of milliseconds. Previous analysis [9], [11] has shown that deep-sleep operation saves up to 90% of power consumption compared to active mode. It is worth noting that although the

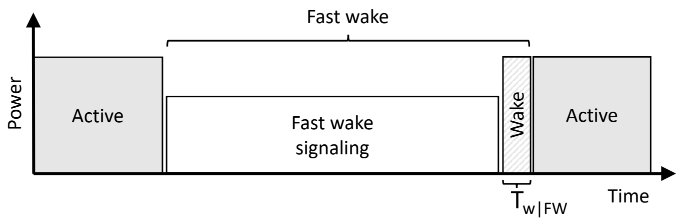


Fig. 2: Overview of the fast-wake operation

standard specification defines the link operation during low-power mode, it is up to the data link layer to decide when to enter this low-power mode, either immediately when idling or after a *hold* period that reduces transitions between active and low-power modes. We elaborate on this topic later.

B. Fast-wake operation

Figure 2 shows the fast-wake operation for low power consumption. As opposed to deep-sleep, the transmitter stays on, continuously signaling to the receiver. The receiver partially shuts down to save power, keeping only the minimum number of active components to negotiate physical layer synchronization with the transmitter. Due to this partly active operation, the wake-up delay, denoted as $T_{w|FW}$, is significantly shorter compared to that in the deep-sleep mode, on the order of nanoseconds instead of microseconds. However, power savings are less, up to 40% compared to active mode.

C. Extensions to low-power operation

As mentioned earlier, link specifications describe the mechanisms of low-power operation but do not dictate when to enter that mode when idling. Previous studies [11], [13], [10], [16], [12] have identified wake-up delay as a possible source of significant network performance overhead, leading to application performance degradation. Specifically, if a link enters low-power mode immediately after becoming idle, bursty traffic of few packets, a single packet at worse, suffers the wake-up delay on every transmission of a packet. To alleviate that, existing approaches propose inserting a *hold* interval (also referred to as a stall timeout) to keep the link active for a certain period of time before it transitions to low-power mode. During the hold interval the link consumes full power, being in active mode, but it is ready to transmit a packet without added delay. The optimal selection of the hold timeout to maximize power savings, while minimizing performance penalty, has been a topic of research [12].

In the next section, we discuss the design and implementation of the power and performance model. The model applies to both existing approaches for low-power link operation, namely deep-sleep and fast-wake modes, and also it is flexible to include extensions, such as hold timeouts, to investigate performance and power trade-offs.

III. Modeling and Simulation Methodology

In this section, we present the methodology to model low-power operation, including its effects on power consumption

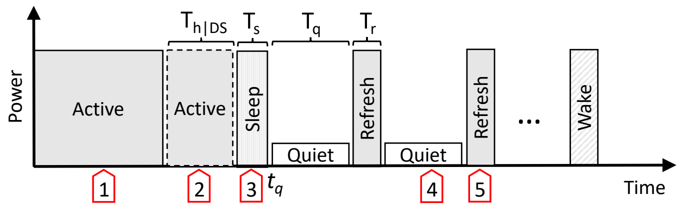


Fig. 3: Various states of a link in the deep-sleep mode. Markers show possible times of packet transmission.

TABLE II: Deep-sleep model timing analysis

T_x marker	Overhead (T_x)	Time of P_{ds}
1	0	$\varnothing$
2	0	$\varnothing$
3	$T_{w\|DS}$	$\varnothing$
4	$T_{w\|DS}$	$\frac{t_4-t_q}{T_q} - \lceil \frac{t_4-t_q}{T_q} - 1 \rceil$
5	$T_{w\|DS}$	$\frac{t_5-t_q}{T_q} - \lceil \frac{t_5-t_q}{T_q} - 1 \rceil$

and performance. In order to do this, we extend the network modeling capabilities in the state-of-the-art network simulator TraceR-CODES. Our extensions closely follow the definition of low-power link operation in the IEEE Ethernet Standard, which is the state-of-the-art specification for low-power operation. Nevertheless, the power model is general, and can apply to any kind of networking technology, by being parameterized with respect to power consumption and delay parameters.

We use realistic parameters taken from the ratified Ethernet standard to test the model, and to be able to compare with literature on energy-efficient networks.

A. Modeling Different Low Power Modes

As opposed to a link running at full power in the default case, we modify the network models in TraceR-CODES to keep track of the *power state* or current mode of each link. Specifically, for each link, the model segments the total time of the simulation into periods where the link is operating in different power modes. When a request for packet transmission arrives, the current power mode of the link determines any delay to packet transmission due to low-power operation, and the corresponding power consumption of the link. Below, we provide details about each low power mode and its model in the simulation framework.

1) Deep-sleep mode: Figure 3 shows the various states in which a link can be at different times when modeling deep-sleep operation. Markers show possible times of packet transmission. Note that besides including the intervals for low-power link operation, the model includes a hold interval of duration T_h to model stalling the switch to low-power mode. This is as per the extensions to link operation for reducing overheads, as described in Section II-C. Additionally, Table II lists the increase in transmission time (T_x) and the accumulated time spent in low-power mode, for a single packet transmission and each marked occasion.

Briefly, transmitting a packet during either the active interval (1) or the hold interval (2) has no transmission overhead and

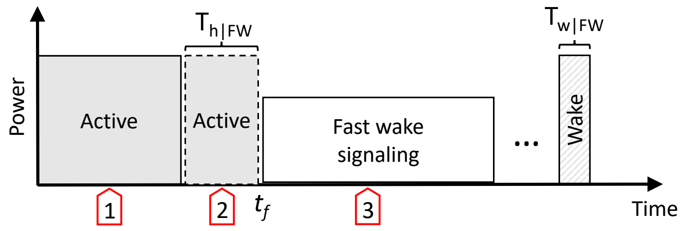


Fig. 4: Various states of a link in the fast-wake mode. Markers show possible times of packet transmission.

TABLE III: Fast-wake model timing analysis

T_x marker	Overhead (T_x)	Time of P_{fw}
1	0	∅
2	0	∅
3	$T_{w\|FW}$	$t_3 - t_f$

also no power savings. Transmitting a packet after the link has entered the sleep interval (3) incurs the overhead of waking up the link. However, there are no power savings since the link stays in active mode during this sleep signaling phase. After the link enters low-power operation (t_q), transmitting a packet, either during a quiet interval (4) or a refresh interval (5), has the added delay of waking up the link. Nevertheless, in both those cases the link saves power, since it has been in quiet mode, and the time interval of reduced power consumption (P_{ds}) is shown in the table.

2) Fast-wake mode: Similarly, Figure 4 and Table III show the model for fast-wake, including the impact on transmission time and power consumption. Again, the model includes a hold interval of active operation. If a packet transmits during the active (1) or hold (2) interval, there is no transmission overhead and no power savings. After the link enters fast-wake signaling (t_f), transmitting a packet has a delay equal to the wake-up time of fast-wake and link operation saves power during the fast-wake signaling interval up to transmission time.

3) Combining deep-sleep and fast-wake: Previous work [16], [12] shows that combining deep-sleep and fast-wake modes in a *hybrid*, low-power operation improves either power or performance over using one of them alone. In this hybrid mode of operation, fast-wake precedes entering deep-sleep, thus it reduces the overhead in transmission compared to immediately entering deep-sleep mode. Also, it saves additional power than the fast-wake operation alone, by entering the deep-sleep mode for long periods of inactivity.

Figure 5 presents the model of hybrid, low-power operation, while Table IV shows the timing analysis of the model. In this mode, the hold interval for deep-sleep operation becomes the interval during which the link operates in fast-wake mode. Table IV shows the detailed analysis on timings.

B. Implementation

Our implementation extends the network model at the *link level*, thus it is applicable to any network topology. It extends the simulator data structures of each link to include timestamp

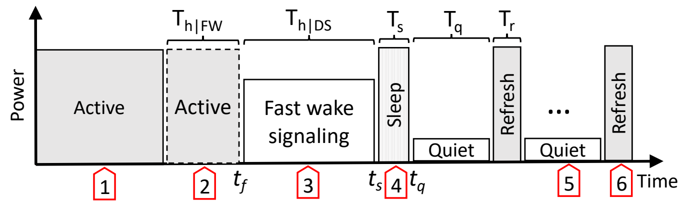


Fig. 5: Various states of a link in the hybrid fast-wake + deep-sleep operation. Markers show possible times of packet transmission.

TABLE IV: Fast-wake + deep-sleep model timing analysis

T_x marker	Overhead (T_x)	Time of P_{ds}	Time of P_{fw}
1	0	∅	∅
2	0	∅	∅
3	$T_{w\|FW}$	∅	$t_3 - t_f$
4	$T_{w\|DS}$	∅	$t_s - t_f$
5	$T_{w\|DS}$	$\frac{t_5-t_q}{T_q} - \lceil \frac{t_5-t_q}{T_q} - 1 \rceil$	$t_s - t_f$
6	$T_{w\|DS}$	$\frac{t_6-t_q}{T_q} - \lceil \frac{t_6-t_q}{T_q} - 1 \rceil$	$t_s - t_f$

variables, to keep track of the time intervals defined in the model, i.e., t_f, t_s, and t_q. Also for each link, we add counter variables to accumulate time spent in the possible modes of operations – active, deep-sleep, or fast-wake, and report these in the output at the end of the simulation.

Whenever a packet arrives at a link for transmission, the simulator decides whether the packet will be delayed or not, depending on the operation mode the link is in. At transmission time, the simulator also updates the time counters, accumulating time for the different modes of operation of the link, and updates the timestamp variables, moving them in future time in anticipation of the next packet transmission. At the end of the simulation, the time counters are updated one last time, to count the timing intervals after the last transmission. Notably, the implementation is configurable for all the timing parameters, hence applicable to any link technology that fits the model, present or future.

IV. Experimental Setup

This section describes network configurations, other simulation parameters, and the workloads of deployed applications.

A. Network parameters

Table V summarizes the network configurations used in simulations. In brief, the target is to simulate a contemporary, large supercomputer installation. For that, we assume a switch radix of 48 ports and 4,608 compute nodes (similar to LLNL's Sierra cluster, which has 4,320 nodes, and ORNL's Summit cluster, which has 4,608 nodes). Simulation evaluates the following topologies: (i) a 3-level fat-tree, (ii) a row-column (RC) dragonfly, and (iii) HyperX. Fat-tree and dragonfly topologies are widely used in HPC datacenters, while HyperX is recently proposed as a scalable alternative that achieves similar performance with fewer switches. The choice of how many compute nodes to attach to each switch of different

TABLE V: Network configurations used in the simulations (switch radix=48, link speed=100 Gbps)

Topology	Fat-tree	RC dragonfly	HyperX
Compute nodes per switch	24	8	12
Number of switches	192	576	384
Number of compute nodes	4,608	4,608	4,608
Description	3-level	6×16	$8 \times 8 \times 6$

TABLE VI: Parameter values used in the power models (based on 100GBase-*R)

Technology parameters	
T_s	1.1 µs
$T_{w\|DS}$	5.5 µs
$T_{w\|FW}$	0.34 µs
P_{DS}	0.1 × P_{ACT}
P_{FW}	0.6 × P_{ACT}
Power configuration	
Mode	Hold
Active (ACT)	–
Deep-sleep (DS)	0, 1, 2, 4 ($\times T_s$)
Fast-wake (FW)	0, 1, 2, 4 ($\times T_s$)
Hybrid (FW+DS)	1, 2, 4 ($\times T_s$)

topologies ensures that the injection and global bandwidths are balanced, see [17]

B. Power model parameters

Table VI shows the values for the parameters of the power model, described in detail in section III-A. We base those numbers on the IEEE Ethernet specification [8] for 100GBase, copper links. Our motivation for this choice is to use realistic parameters ratified by the Ethernet standard and to be able to compare with other literature on energy-efficient networks.

Those technology parameters set the delay of signaling time T_s and wake-up time $T_{w|DS}$ for deep-sleep operation, and $T_{w|FW}$ for fast-wake, respectively. Power consumption of deep-sleep mode and fast-wake mode is parameterized on the power consumption of active mode. Specifically, the power consumption of the deep-sleep mode is 0.1× of the power consumption of active mode, while that of fast-wake is 0.6× of active mode, following observations reported in literature [18].

Further to the technology parameters for low-power operation, the table also shows the experimentation parameters of all possible power modes. In more detail, we perform experiments in active (ACT) mode, which means links are always on without implementing low-power operation, deep-sleep (DS), fast-wake (FW), and the hybrid mode which combines fast-wake and deep-sleep. In low-power modes, we try different *hold* values, defined as multiples (0×, 1×, 2×, 4×) of the technology parameter T_s, which sets a lower bound for the hold time need to enter deep-sleep operation. The configurations of the hybrid power mode assume that $T_{h|FW} = T_{h|DS}$ to have a manageable experimentation space. Note, that hold value 0 for hybrid mode is unused, since it implies only deep-sleep operation.

TABLE VII: Workload configurations

Job mappings	linear, random	
	Single-job	Multi-job
Number of processes	256, 512, 1024, 2048, 4096	4096 Random $a \in \mathcal{A}$ and $r \in 256, 512, 1024$ $\sum r_i = 4096$

Formally, the total average power consumption is

$$\overline{P} = \sum_{i=1}^{L} \frac{P_{ACT} \times t_{i|ACT} + P_{DS} \times t_{i|DS} + P_{FW} \times t_{i|FW}}{t_{i|ACT} + t_{i|DS} + t_{i|FW}}$$

where L is the number of links and $t_{i|x}$ denotes the time spent by link i in power mode x. Depending on the power mode (active, deep-sleep, fast-wake, hybrid) different components of the equation are relevant. Power consumption of a low-power mode is a fraction of the power consumption of the active power mode. We report power savings as percentages compared to the power consumption in the active mode, i.e.

$$\frac{\overline{P}_{ACT} - \overline{P}_x}{\overline{P}_{ACT}} \times 100\%$$

C. Applications and workloads

Table VIII presents in detail the applications used in simulations and table VII shows information on the deployed workloads. All of the programs are HPC proxy applications, part of the ECP proxy application suite [19], that correspond to realistic, large-scale HPC applications.

We classify the communication patterns as bursty or continuous to help interpret the results of the evaluation. Whether an application classifies as bursty or not depends on the application's communication pattern itself, and its scaling, based on its input and number of processes, which affect the ratio of computation to communication. In this setup, *Kripke*, *ExaMiniMD* and *miniFE* have bursty communication intermittent to long computation periods. By contrast, the applications *Laghos* and *miniAMR* communicate frequently. The remaining applications, *AMG* and *SW4lite*, fall in between.

We collect traces, using ScoreP [20], by running a *single iteration* of the main computational loop, which is representative of the computation and communication patterns of the application, given iterative execution. Moreover, traces include runs with different numbers of processes, from 256 up to 4,096 processes. Inputs to the applications are chosen to represent realistically sized problems, based on previous reports [21], with either strong, weak, or mixed scaling. For brevity, table VIII shows the exact input for 256 processes, which is scaled accordingly for runs of more processes.

For single-job workloads, we experiment by simulating a single application, scaling it from 256 up to 4,096 processes. We generate multi-job workloads by simulating multiple applications co-executing. The applications and their number of processes are selected at random, with the restriction that the total number of processes for the workload sums up to

TABLE VIII: Applications used in the simulations and their inputs at 256 nodes used for generating the traces.

Application	Input Parameters	Time (s)	Scaling
Kripke	–procs 8,8,4 –zones 256,256,128 –quad 16 –niter 1	5.54	Weak
Laghos	-pt 221 -pa -p 1 -tf 0.6 -no-vis -m cube01_hex.mesh –cg-max-steps 50 –max steps 4 -ok 2 -ot 1 -rs 3 -rp 2	6.35	Mixed
AMG	-problem 2 -n 32 32 32 -P 8 8 4	1.20	Weak
ExaMiniMD	box 100×100×100, 4×10^6 atoms	1.83	Strong
SW4lite	grid x=1.0 y=1.0 z=1.5 h=0.00226	0.62	Weak
miniFE	-nx 512 -ny 512 -nz 256	2.03	Weak
miniAMR	–npx 8 –npy 8 –npz 4 –nx 8 –ny 8 –nz 8	0.032	Weak

4,096 processes. Specifically, each application may have either 256, 512, or 1,024 processes. Multi-job workloads are more realistic because supercomputers execute multiple jobs sharing the entire machine for most of their lifetime. The network simulator replays previously collected traces, and executes multiple iterations of each job of a particular application and number of processes to maximize co-execution time of all jobs. That is, short-running jobs execute multiple times to have approximately the same execution time of larger, co-executing jobs. Importantly, we generate and simulate 20 multi-job workloads to have a statistically sufficient sample.

In summary, for single-job execution, we perform 3 (topologies) $\times$ 2 (mappings) $\times$ 7 (applications) $\times$ 5 (process counts) $\times$ 12 (power mode and hold combinations) = 2520 experiments. For multi-job execution, we perform 3 (topologies) $\times$ 2 (mappings) $\times$ 20 (workloads) $\times$ 12 (power mode and hold combinations) = 1440 experiments. Each single-job simulation needs a maximum of 2 hours to complete, depending on the required simulated time and communication intensity. Each multi-job simulations needs a maximum of 12 hours to complete depending on the consisting workload. Due to the large number of experiments, the simulations have been executed in parallel, on a cluster to speed up their completion. The next section discusses the results of those experiments.

V. Results of Single Job Execution

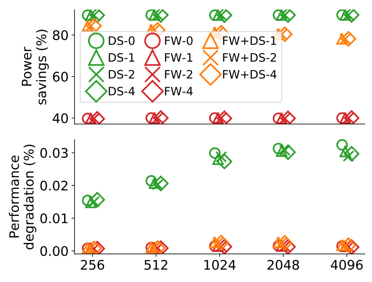
Due to the large volume of experiments, we show indicative plots to highlight the most important findings. Figure 6 shows power savings and performance degradation compared to executing with always-on, active power for all applications and number of processes used. The figure shows results on the dragonfly topology only, both linear and random mappings, since results are similar for the fat-tree and HyperX topologies.

Expectedly, power savings are maximal for the deep-sleep power mode, followed by the hybrid (FW+DS) one, and last is the fast-wake only mode. Notably, power savings for deep-sleep and fast-wake modes approach the possible maximum, which is 0.9$\times$ and 0.4$\times$ of active power, respectively. Regarding hybrid mode, power savings depend on the time spent in its fast-wake or deep-sleep components. A key observation is that this is application dependent. Specifically, the length and dispersion of communication periods, i.e., burstiness of transmissions, determines the time spent in active, fast-wake or deep-sleep mode. Referring back to figure 5, which details the hybrid model, if the period of communication bursts is less than the hold interval $T_{h|DS}$, low-power operation will spend most time in the fast-wake component. Alternatively, if communication periods are larger than this hold intervals, low-power operation will enter deep-sleep and remain there until the next transmission. For example, contrasting figures 6a and 6b on power savings, *Kripke* executing in hybrid mode has power savings close to deep-sleep only mode, thus hybrid low-power operation spends most of the time in the deep-sleep component, due to communication periods being larger than the hold interval. By contrast, *Laghos* in hybrid mode has identical power savings to fast-wake only mode, due to its short communication period that enables fast-wake low-power operation and returns to active mode on the next packet transmission. This observation on communication periods also justifies why *Kripke* has negligible performance degradation in any power mode, whereas *Laghos* is more communication intensive, hence deep-sleep degrades its performance up to 15%, by the added delay overhead from frequently switching between active and deep-sleep modes.

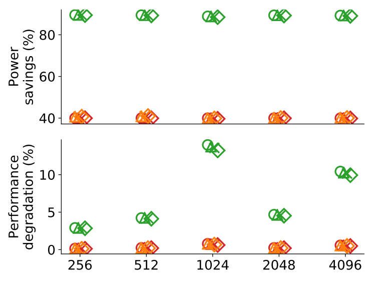
Elaborating on performance degradation for single-job execution, our results agree with literature, thus cross-validate with existing work. Specifically, as with results of previous studies, performance degradation is application dependent. It depends on the computation to communication ratio and on the burstiness of communication. Furthermore, performance degradation is analogous to the wake-up delay incurred by the power saving mode, with deep-sleep incurring the most delay, followed by fast-wake, whereas the hybrid mode will incur delay proportional to the percentage of time spent in each of its component low-power modes.

The performance degradation ranges from negligible, for example *Kripke* experience less than 0.03% across power modes, to significant. By contrast, *miniAMR* experiences up to 10$\times$ slowdown when the deep-sleep mode is enabled. Interestingly, fast-wake and hybrid modes have negligible performance degradation (<1%) for all programs. That suggests that they present an attractive deployment decision to enable power savings with little performance degradation. Notably, the deep-sleep mode may be prohibitive for certain applications that are communication-bound, as is the case for *miniAMR*. Notably, performance degradation is higher in random mapping, since packets may need to traverse more links to reach their destination, hence accumulate more latency by waking up links in low-power operation.

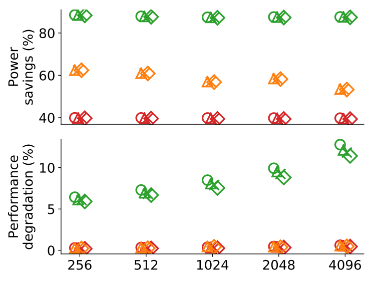
Summary: For single-job execution, fast-wake and hybrid modes present the best proposition because they show neg-



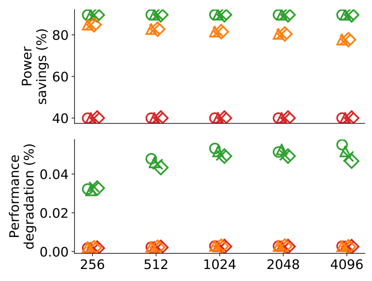
(a) Kripke (linear mapping)



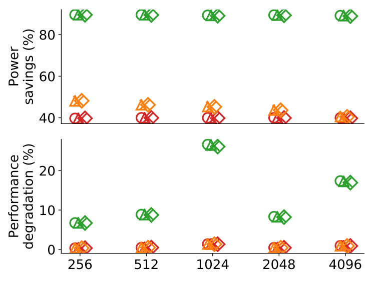
(b) Laghos (linear mapping)



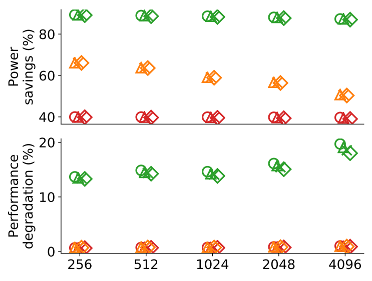
(c) AMG (linear mapping)



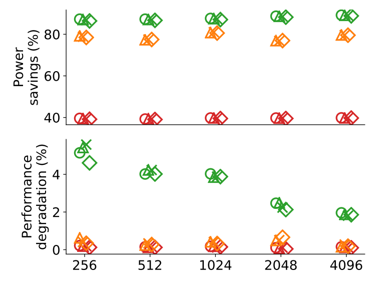
(d) Kripke (random mapping)



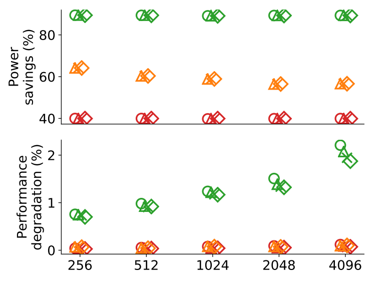
(e) Laghos (random mapping)



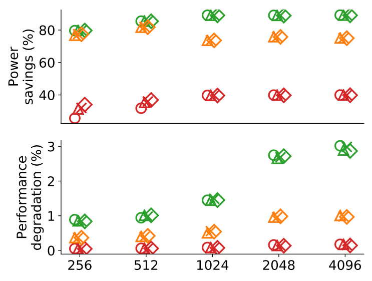
(f) AMG (random mapping)



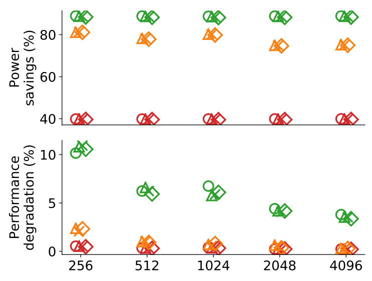
(g) ExaMiniMD (linear mapping)



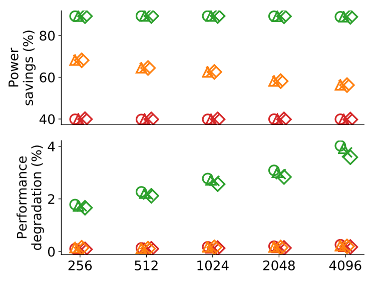
(h) SW4lite (linear mapping)



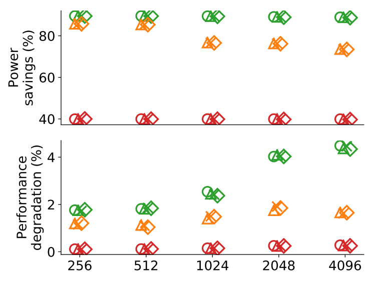
(i) miniFE (linear mapping)



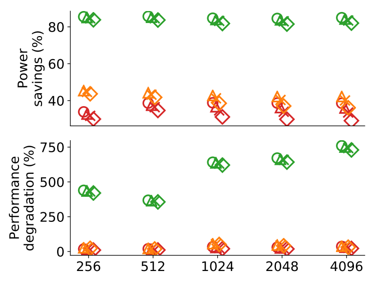
(j) ExaMiniMD (random mapping)



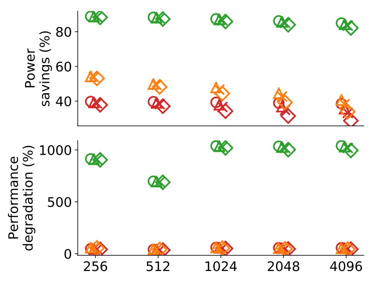
(k) SW4lite (random mapping)



(l) miniFE (random mapping)



(m) miniAMR (linear mapping)



(n) miniAMR (random mapping)

Fig. 6: Dragonfly: Power savings and performance degradation compared to execution with active power. X-axis shows number of processes. Markers show different power and hold modes

ligible performance degradation with power savings ranging from 40% to about 80%. The upper limit of power savings is possible in hybrid mode, depending on whether burstiness of communication enables the deep-sleep component. Deep-sleep provides maximal power savings of up to 90% but may slow down performance significantly (up to 10×) for communication-bound applications.

VI. Results of Multi-job Workload Execution

This section discusses the results and insight on multi-job workloads. This is the first time that a study on energy-efficient interconnects evaluates realistic, multi-job execution. Figure 7 presents power savings and performance degradation across the 20 multi-job workloads, showing the distribution of results in violin plots, for all power modes, topologies and mappings. Labels on the plot show the median value.

As a first observation, results are similar across topologies, thus the effects of low-power operation are independent of the particular topology. Deep-sleep modes present the highest power savings, between 82% and 90%, followed by the hybrid mode, between 38% and 43%, and last is the fast-wake mode, between 36% to 39%. The hold intervals have little impact to power savings and performance degradation. Notably, in all cases, the hybrid mode has similar power savings to the fast-wake only mode. This is because cross-traffic, from sharing the interconnect, keeps most links in fast-wake operation when idling rather than deep-sleep.

Interestingly, performance degradation is modest for deep-sleep, between 3% and 7%. There is the outlier of *miniamr*, which has performance slowdown of up to 2× (4×) in linear (random) mapping, since it it is communication bound, as discussed in section V. Nevertheless, this slowdown is much less than the 10× slowdown shown for single-job execution. Generalizing, performance degradation in multi-job execution for deep-sleep is less than that of single-job execution because cross-traffic from other executing jobs in the multi-job workloads keeps shared links active, thus avoids the latency penalty of those links. So, deep-sleep has maximal power savings on non-shared links with the added latency penalty and less power savings on shared links kept active from cross-traffic but without any performance penalty. This observation makes deep-sleep a more attractive proposition for multi-job workloads rather than single-job ones.

Also, performance degradation is negligible for fast-wake and hybrid modes (<1%) due to the limited delay overhead of fast-wake operation and cross-traffic.

Summary: Deep-sleep is an excellent proposition for significant power savings ($> 80\%$) with modest performance degradation ($< 7\%$) for other than communication bound applications. Fast-wake and hybrid modes provide very similar power savings of about 40% with negligible performance degradation for any type of applications – communication or computation bound.

VII. Related Work

This section summarizes the existing techniques for energy-efficient interconnection networks, network simulators and other methods for estimating the power and performance of interconnection networks.

A. Energy-efficient networking

Besides low-power idle operation, other approaches for reducing power consumption of interconnection networks propose reducing the link rate to throttle data transfer, thus reduce power consumption through demoting performance. For example, an optical link of the type 4× QDR InfiniBand consumes 80% or 90% of the power that 4× SDR link or a DDR InfiniBand links consumes, respectively. InfiniBand provides the ability to dynamically configure link rate [3], with the overhead of several nanoseconds to microseconds. A similar approach to InfiniBand, which is called ALR (Adaptive Link Rate), has been discussed in the IEEE802.3 Ethernet committee [4], [5]. Another approach [3] to improving the power efficiency of interconnection networks is to dynamically adjust the number of lanes used for communication. All these proposals have been shown to save power during low network load but fail to reduce the power of idle links, thus have modest power savings. In that sense, low-power idle operations is superior given it has little to no performance degradation in fast-wake modes.

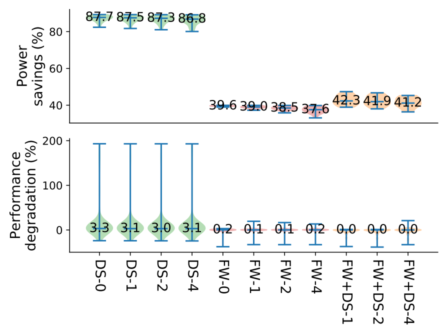
Li et al. [22] propose a technique similar to low-power idle operations, by turning links on and off. However, their proposal requires a separate control network to power up or down links. The control network needs to be always powered on, thus reduces the potential power savings. Using the same approach, Alonso et al. [23] propose an automatic way for turning on and off links, leveraging traffic information. Despite turning on and off links intelligently, the always-on operation of the control network limits power savings.

Reviriego et al. [9] present an experimental evaluation of energy-efficient Ethernet with low-power idle operation to show potential of power savings. We used insight from this study to motivate our work.

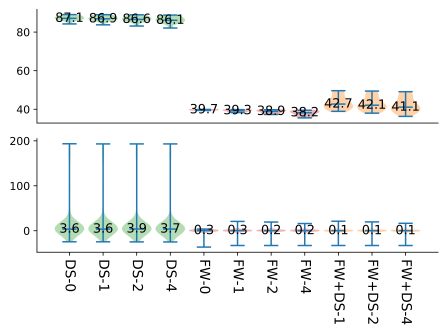
B. Network simulation

There are several simulators [24], [25], [26], [27], [28], [29], [30], [31], [32] for interconnection networks in HPC systems. Typically, they use communication traces collected by executing an application on a real system. Execution models among simulators vary on fidelity and accuracy, ranging from flit to packet-level simulation. Importantly, most of those simulators do not support multi-job workloads. TraceR-CODES can simulate multi-job workloads and has been validated in several studies [33], [34], hence our choice to extend it for modeling power consumption.

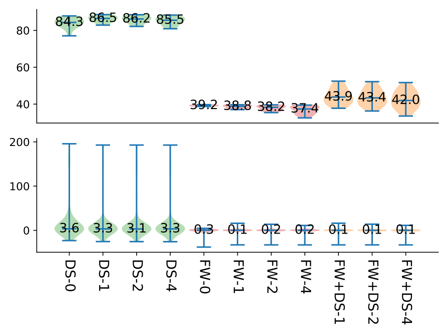
Closer to our approach is the work of Saravanan et al. [11], [13], [16], [12]. That work extends the Dimemas [26] simulator to model power consumption. However, its has limited scope because it only studies single-job workloads in a single fat-tree topology scaled up to 256 nodes. By contrast, our



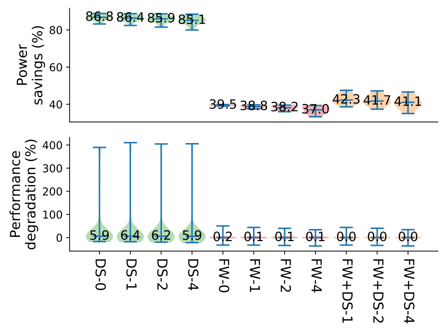
(a) Dragonfly (linear mapping)



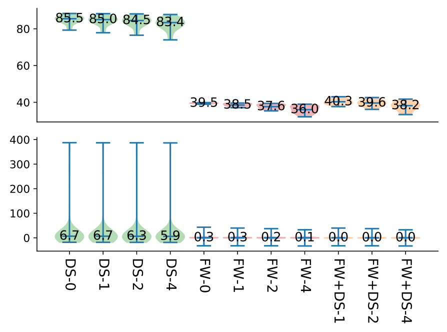
(b) Fat-tree (linear mapping)



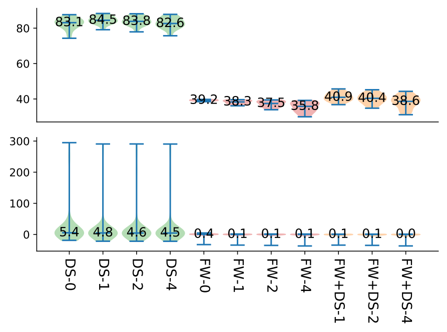
(c) HyperX (linear mapping)



(d) Dragonfly (random mapping)



(e) Fat-tree (random mapping)



(f) HyperX (random mapping)

Fig. 7: Power savings and performance distributions (violin plots) across 20 multi-job workloads for different power modes, split to different topologies and mappings

work includes multi-job workloads to realistically simulate network congestion and actual operation of a supercomputer and tests three state-of-the-art network topologies (dragonfly, fat-tree, HyperX) scaled to 4,608 nodes to evaluate actual network sizes. Our work validates previous results on single-job execution but crucially presents new results and provides novel insight on realistic, multi-job execution for energy-efficient networking.

Other approaches employ analytical modeling for performance estimation [35], [36] and power consumption [37], [6] of energy-efficient networks. However, those models abstract away congestion and other important effects to simplify estimation, thus suffer from inaccuracies.

VIII. Conclusion

We present the first study on power savings and performance implications of energy-efficient interconnects for realistic, multi-job workloads on large HPC systems. We extend the TraceR-CODES network simulator to model low-power operation alongside performance modeling. We perform experiments on three, contemporary network topologies, namely dragonfly, fat-tree, and HyperX, simulating a cluster of 4,608 compute nodes. Moreover, we experiment with different, state-of-the-art low-power modes (deep-sleep, fast-wake, and hybrid), which present different trade off points for saving power and increasing latency on communications. For the evaluation, we use traces from HPC proxy applications, executing either as a single-job or as a multi-job workloads, to emulate typical execution on a supercomputer.

Results on single-job execution validate existing literature for large network deployments: power savings are analogous to the low-power operation, and performance degradation is proportional to the communication intensity of the application. More importantly, results on multi-job workloads present new insights: performance degradation from deep-sleep operation is minimal due to cross-traffic that keeps shared links active, thus avoiding accumulating wake-up delays. For future work, we intend to explore the power and performance characteristics of future supercomputing installations by scaling up the number of compute nodes and deployed workloads, and to investigate the extent that power-aware routing algorithms provide additional gains.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DEAC52-07NA27344 (LLNL-CONF-791976).

References

[1] C. E. Leiserson, “Fat-trees: universal networks for hardware-efficient supercomputing,” *IEEE Trans. Comput.*, vol. 34, no. 10, pp. 892–901, 1985.

[2] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, pp. 77–88, June 2008.

[3] 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, 2010, pp. 338–347.

[4] C. Gunaratne and K. Christensen, "Ethernet adaptive link rate: System design and performance evaluation," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 28–35.

[5] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of ethernet with adaptive link rate (alr)," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 448–461, April 2008.

[6] X. Pan, T. Ye, T. T. Lee, and W. Hu, "Power efficiency and delay tradeoff of 10gbase-t energy efficient ethernet protocol," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2773–2787, Oct 2017.

[7] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "Ieee 802.3az: the road to energy efficient ethernet," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 50–56, November 2010.

[8] "Ieee standard for ethernet," *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, pp. 1–5600, Aug 2018.

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

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

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

[12] K. P. Saravanan and P. M. Carpenter, "Perfbound: Conserving energy with bounded overheads in on/off-based hpc interconnects," *IEEE Transactions on Computers*, vol. 67, no. 7, pp. 960–974, July 2018.

[13] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, "A performance perspective on energy efficient hpc links," in *Proceedings of the 28th ACM 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

[14] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, June 2008, pp. 77–88.

[15] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Nov 2009, pp. 1–11.

[16] K. P. Saravanan, P. M. Carpente, and A. Ramirez, "Exploring multiple sleep modes in on/off based energy efficient hpc networks," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, Oct 2015, pp. 54–61.

[17] A. Bhatele, N. Jain, M. Mubarak, and T. Gamblin, "Analyzing cost-performance tradeoffs of hpc network designs under different constraints using simulations," in *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM-PADS '19. New York, NY, USA: ACM, 2019, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/3316480.3325516

[18] P. Reviriego, V. Sivaraman, Z. Zhao, J. A. Maestro, A. Vishwanath, A. Sánchez-Macian, and C. Russell, "An energy consumption model for energy efficient ethernet switches," in *2012 International Conference on High Performance Computing Simulation (HPCS)*, July 2012, pp. 98–104.

[19] O. Aaziz, J. Cook, J. Cook, T. Juedeman, D. Richards, and C. Vaughan, "A methodology for characterizing the correspondence between real and proxy applications," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2018, pp. 190–200.

[20] A. Knüpfer, C. Rössel, D. a. Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, and F. Wolf, "Score-p: A joint performance measurement run-time infrastructure for periscope,scalasca, tau, and vampir," in *Tools for High Performance Computing 2011*, H. Brunst, M. S. Müller, W. E. Nagel, and M. M. Resch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–91.

[21] D. Richards, O. Aziz, J. Cook, H. Finkel *et al.*, "Quantitative performance assessment of proxy apps and parents," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2018.

[22] J. Li, W. Huang, C. Lefurgy, L. Zhang, W. E. Denzel, R. R. Treumann, and K. Wang, "Power shifting in thrifty interconnection network," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, Feb 2011, pp. 156–167.

[23] M. Alonso, S. Coll, J. M. Martinez, V. Santonja, P. Lopez, and J. Duato, "Dynamic power saving in fat-tree interconnection networks using on/off links," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, April 2006, pp. 8 pp.–.

[24] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Jun. 2014. [Online]. Available: http://hal.inria.fr/hal-01017319

[25] G. Zheng, G. Kakulapati, and L. V. Kale, "Bigsim: a parallel simulator for performance prediction of extremely large parallel machines," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, April 2004, pp. 78–.

[26] R. M. Badia, F. Escalé, E. Gabriel, J. Gimenez, R. Keller, J. Labarta, and M. Müller, "Dimemas: Predicting mpi applications behaviour in grid environments," in *Workshop on Grid Applications and Programming Tools*, 2003.

[27] M. Hideki, S. Ryutaro, S. Hidetomo, H. Tomoya, M. Jun, Y. Makoto, K. Takayuki, A. Yuichiro, M. Ikuo, S. Toshiyuki, O. Yuji, A. Hisashige, I. Yuichi, I. Koji, A. Mutsumi, and M. Kazuaki, "Nsim: An interconnection network simulator for extreme-scale parallel computers," *IEICE Transactions on Information and Systems*, vol. E94.D, no. 12, pp. 2298–2308, 2011.

[28] A. Degomme, A. Legrand, G. S. Markomanolis, M. Quinson, M. Stillwell, and F. Suter, "Simulating mpi applications: The smpi approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2387–2400, Aug 2017.

[29] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Jun. 2014.

[30] C. L. Janssen, H. Adalsteinsson, S. Cranford, J. P. Kenny, A. Pinar, D. A. Evensky, and J. Mayo, "A simulator for large-scale parallel architectures," *International Journal of Parallel and Distributed Systems*, vol. 1, no. 2, pp. 57–73, 2010.

[31] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling parallel simulation of large-scale hpc network systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 87–100, Jan. 2017. [Online]. Available: https://doi.org/10.1109/TPDS.2016.2543725

[32] N. Jain, A. Bhatele, S. White, T. Gamblin, and L. V. Kale, "Evaluating hpc networks via simulation of parallel workloads," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16, 2016, pp. 14:1–14:12.

[33] X. Yang, J. Jenkins, M. Mubarak, R. Ross, and Z. Lan, "Watch out for the bully! Job interference study on dragonfly networks," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.

[34] N. Jain, A. Bhatele, L. Howell, D. Böhme, I. Karlin, E. Leon, M. Mubarak, N. Wolfe, T. Gamblin, and M. Leininger, "Predicting the performance impact of different fat-tree configurations," in *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. IEEE Computer Society, Nov. 2017, lLNL-CONF-736289. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126967

[35] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, "Loggp: incorporating long messages into the logp model—one step closer towards a realistic model for parallel computation," in *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, ser. SPAA '95. New York, NY, USA: ACM, 1995, pp. 95–105.

[36] K. L. Spafford and J. S. Vetter, "Aspen: A domain specific language for performance modeling," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12, 2012, pp. 84:1–84:11.

[37] S. Herreria-Alonso, M. Rodriguez-Perez, M. Fernandez-Veiga, and C. Lopez-Garcia, “A gi/g/1 model for 10 gb/s energy efficient ethernet links,” *IEEE Transactions on Communications*, vol. 60, no. 11, pp. 3386–3395, November 2012.