Power Shifting in Thrifty Interconnection Network

Jian Li   Wei Huang   Charles Lefurgy
IBM Research - Austin
Austin, TX
jianli, huangwe, lefurgy@us.ibm.com

Lixin Zhang*
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
zhanglixin@ict.ac.cn

Wolfgang E. Denzel
IBM Research - Zurich
Rueschlikon, Switzerland
wde@zurich.ibm.com

Richard R. Treumann
IBM System & Technology Group
Poughkeepsie, NY
treumann@us.ibm.com

Kun Wang
IBM Research - China
Beijing, China
wangkun@cn.ibm.com

Abstract

This paper presents two complementary techniques to manage the power consumption of large-scale systems with a packet-switched interconnection network. First, we propose Thrifty Interconnection Network *(TIN), where the network links are activated and de-activated dynamically with little or no overhead by using inherent system events to timely trigger link activation or de-activation. Second, we propose* Network Power Shifting *(NPS) that dynamically shifts the power budget between the compute nodes and their corresponding network components. TIN activates and trains the links in the interconnection network, just-in-time before the network communication is about to happen, and thriftily puts them into a low-power mode when communication is finished, hence reducing unnecessary network power consumption. Furthermore, the compute nodes can absorb the extra power budget shifted from its attached network components and increase their processor frequency for higher performance with NPS. Our simulation results on a set of real-world workload traces show that TIN can achieve on average 60% network power reduction, with the support of only one low-power mode. When NPS is enabled, the two together can achieve 12% application performance improvement and 13% overall system energy reduction. Further performance improvement is possible if the compute nodes can speed up more and fully utilize the extra power budget reinvested from the thrifty network with more aggressive cooling support.*

*This work was performed while Lixin Zhang was with IBM Research - Austin.

1 Introduction

Power and energy consumption has become a mounting problem in servers, data centers and other large-scale systems. Energy use in these systems is projected to increase to 100 billion kWh in 2011, which will equate to 2.5% of the total U.S. electricity consumption and $7.4 billion in energy bills [5]. The looming system energy crisis and its corresponding power delivery and cooling requirement demand future computer systems to be built with much stringent power constraint than those nowadays. This trend is even more true for conventional power-hungry supercomputing systems, which accommodate High Performance Computing (HPC) workloads. As a result, low power has become a first-class design requirement for HPC systems [11].

A large-scale system running HPC workloads, can include hundreds of thousands of processing nodes connected via a large packet-switched interconnection network. A closer look into these systems reveals that the power consumption of interconnection links (including link controllers) constitutes not only a majority of the switch power, but also a substantial percentage of the total system power. For instance, the links in an IBM 8-port 12X switch can take 64% of the switch power [18]. The power consumption of the interconnection network in HPC systems can contribute up to around 30% the total system power [23]. When compute nodes are energy proportional with respect to the computation that they are performing, the network power consumption can be an even more significant fraction of the total cluster system power, e.g., 50% [3]. Current high-speed links in the interconnection networks require continuous pulse transmission to keep both ends synchronized, even when no data is transmitting. Therefore, the average power consumption of such links is almost identical to their

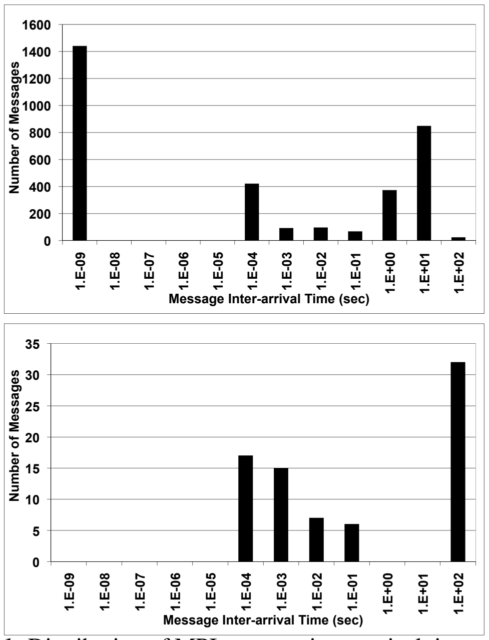

Figure 1: Distribution of MPI message inter-arrival time of SPPM (top) and WRF (bottom).

worst-case power consumption [34].

Current HPC system deployment typically assumes a worst-case scenario for the power sources and cooling systems. We observe that HPC workloads rarely operate all system elements at their maximum capacity simultaneously. For example, infrequent communication patterns exhibited by many workloads allow provisioned network links to stay idle for long periods of time. This is similar, but at a different scale, to what happens within a local compute node, where its processors and memory subsystems typically are not both busy.

Fig. 1 shows the distribution of MPI (Message Passing Interface) message inter-arrival time (time between arrival of two messages) for SPPM [27] and WRF [26]. The underutilization of interconnection networks is apparent in this figure. Most messages finish transmission long before their next message requests, resulting in a lot of I/O idle time. In SPPM, a majority of messages are separated by 100 μs or longer, while, in WRF, all the messages are separated by at least 100 μs. Two main reasons contribute to these phenomena. First, overlapping computation and communication is generally hard for many HPC workloads [2]. Consequently, they often exhibit distinct and bursty computation and communication phases at run time. Second, many scalable HPC workloads are often compute bound to sustain the scalability and do not require much inter-node communication. As a result, a large portion of the power provisioned for communication-intensive workloads (e.g., FFT, PTRANS and GUPS) at HPC centers, may be wasted. The power paid to provision resources that are not transformed into useful computation or communication is called *stranded power*.

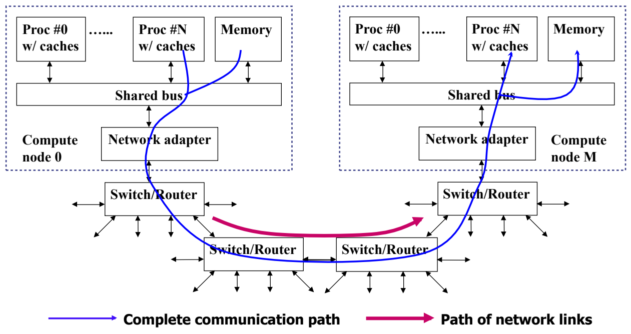

Figure 2: A data transmission path in a large-scale interconnected system.

Fig. 2 illustrates a data communication path in a system from a sender compute node (compute node 0) to a receiver compute node (compute node M). In this figure, each compute node is an SMP (Symmetric Multi-Processor). When a sender processor (processor N in compute node 0) initiates a transfer, it fetches data from its local memory and sends it through a bus shared with other processors in the SMP node. The network adapter transforms the data messages into network packets. The packets then go through a number of switches (i.e., network hops) in a multi-stage network, before reaching the destination node. At the destination, the network packets are reformatted by the receiving network adapter and then sent to the receiving processor (processor N in compute node M). The receiving data is normally copied to the local memory of the receiving processor. Note that some systems may have integrated switches and routers, a.k.a. switchless network, but networking functions still exist.

Based on the insights shown in Fig. 1 that interconnection network can be heavily underutilized, we explore a technique to power up the network links, e.g., those connecting switches and routers in Fig. 2, from their low-power states right before they are needed with no or very little overhead, by relying on hints from built-in system events. For instance, a data transfer is often preceded by a series of special commands, such as memory-mapped I/O writes, that sets up the switch. The first such inherent command can be viewed as an indication that a data transfer is about to happen, thus a link activation sequence should be started. In addition to the power management of the interconnection network, we propose to shift the "saved power" from the underutilized network components (e.g., switch and links) to their corresponding *local* compute nodes, with which they share the same power supply or Power Distribution Unit (PDU). Such power shifting reduces the stranded power associated with the interconnection network, when it is idle. We then increase the processor frequency of the compute nodes for improved performance under the same overall power budget.

Overall, this paper makes the following contributions:

- We propose *Thrifty Interconnection Network* (TIN), which includes a hardware approach that overlaps inherent system events with link transition delay for significant network power reduction without noticeable network performance overhead. It also includes a software extension that uses software-initiated commands as hints to activate and release links.
- We propose *Network Power Shifting* (NPS) that dynamically shifts the limited power budget between compute nodes and their corresponding *local* network components for the maximum application performance within a given overall power budget.
- We simulate our techniques with a set of traces from real-world MPI-based HPC workloads and demonstrate that the proposed techniques have the potential of significant power-performance benefits.

2 DESIGN

2.1 Thrifty Interconnection Network

Conventionally, the links in an interconnection network are continuously in an operational state, i.e., full-power data transmitting state. Alternatively, some researchers have proposed to maintain the links at different low-power modes based on the prediction of the intensity of future data traffic [32]. Those proposals typically require sophisticated prediction hardware to sustain the same communication performance level; therefore, their effectiveness is largely dependent on their prediction accuracy. However, unlike many commercial workloads, HPC workloads typically cannot tolerate performance loss due to misprediction [3]. Furthermore, some emerging workloads, such as financial computing, cannot tolerate such performance loss either. Therefore, in this work, we focus on non-prediction power-saving techniques.

Our approach is based on an observation that significant delay can exist from when a switch sees the first command of a data transfer to when it sends out the first data packet. For instance, a data transfer to a remote node may take the following steps to start up: (1) The sender node sends a series of memory-mapped I/O writes to set up the network adapter; (2) The network adapter interprets the commands and determines the physical addresses of the sending message; (3) The network adapter issues appropriate read requests to fetch message data from local memory or caches; (4) The switch sends out the fetched message data packetized by the network adapter.

In a node with a snoop-based coherence protocol, each operation between the network adapter and the processor or memory can include a snoop phase, a response generation phase, and a data transfer phase, each of which may take many hundreds of cycles. We observed in some large-scale SMP systems that the snoop latency could be in the range of hundreds of processor cycles and memory fetch latency can be over a thousand processor cycles. As a result, each of the write and fetch operations in aforementioned steps (1) and (3), respectively, can take a thousand processor cycles or more to complete. Multiple thousands of cycles may have elapsed from when the network adapter sees the first special write command to when the switch sends out the first packet, even without contention. Therefore, we propose to use the snoop request of the first write command in the aforementioned step (1) as a trigger to start activating the proper link or links at the first switch after the network adapter. At each intermediate hop, a switch treats the trigger signal as a special single-flit message to wake up the proper links, which can be decided by the specific routing schemes in place (Section 2.1.1). At the destination node, the special message is discarded.

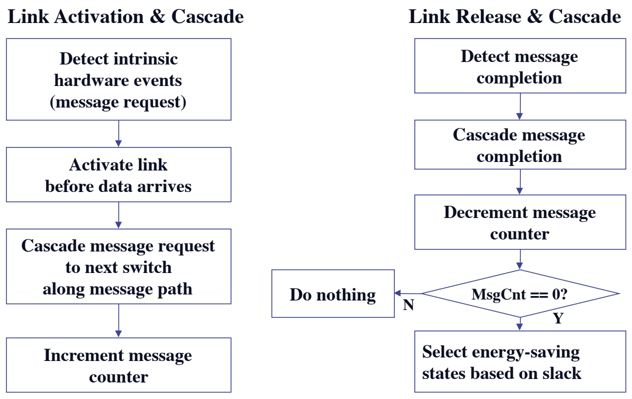
Link Activation & Cascade
Detect intrinsic hardware events (message request)
Activate link before data arrives
Cascade message request to next switch along message path
Increment message counter
Link Release & Cascade
Detect message completion
Cascade message completion
Decrement message counter
MsgCnt == 0?
N
Do nothing
Y
Select energy-saving states based on slack

Figure 3: Link management policy for link activation (left) and release (right).

Note that our proposal differs from prior art in that we do not use prediction to start link transition before data communication arrives. In addition, since the technique is at the granularity of links and the system node they attach to, it is independent of the choice of network topology and routing schemes.

2.1.1 Policy

Fig. 3 illustrates a high level flowchart of providing link services in the thrifty network. First, when the network adapter in the sender node detects a write from the local node to start a data transmission, it immediately starts activating the local switch. Second, when the adapter receives enough write commands such that it can determine the destination address, it generates a special single-flit message as a link activation message, to pass onto its local switch. The switch determines the next hop of the transmission and activate the associated link or links. It then forwards the link activation message to the next switch in the path.

In order for the link activation message to traverse to the next switch while the data network path is in a low-power mode, a separate *control network* is needed and has to be always powered on for the link activation message to flow through. The control network is a companion network of the data network that has the same reach of the latter. The control network may be implemented by using designated narrow links or consuming a small fraction of the data network link width. Either way, it consumes much less power than the data network.

When a next switch receives a link activation messages from the control network, it powers up its own data links and also forwards the activation message to the next hops on the path. The cascading process continues until the activation signal reaches the destination node, forming a *wavefront* of cascading link activations followed by data transmission. The link activation message is discarded at the destination node.

Choosing a data network path to be woken up by the control network is straightforward in a static routing scheme, since it is deterministic. In an adaptive routing scheme, the path to forward can be unpredictable. Since the role of the control network is to wake up the data network in time to sustain performance, we propose to wake up all the necessary data paths that an adaptive routing scheme may use. The control network routing scheme obtains the path information from the one in the data network. It then marks all the legitimate ports for the message and initiates the activation in the corresponding data network links. The hardware support is minimal for marking the to-be-woken-up routes since existing data network routing schemes are already there. Correspondingly, the data network routing hardware has to be always on as the control network; however, the extra power consumption of these components are much smaller compared to the others, e.g., switch crossbars, port buffers and link controllers.

When the network adapter of the sender node finishes a data transfer, it sends out a link release message through the path. To remember the number of outstanding data transfers and support link de-activation, each link maintains a message counter. The counter is incremented whenever an activation signal is received (message initiation) and decremented whenever a release signal is received (message completion), on the same *local* link. When the counter of a link reaches zero, the link and its controller fall into a low-power mode, albeit the control network is always on. When the counters of all links controlled by a switch reach zero, the switch itself goes to a low-power mode. In the case of unsolicited MPI messages, the message counter works in the same fashion, incremented at a message initiation and decremented at a message completion. When a link powers up, its message counter is reset to zero. The number of in-flight messages on the same link should be well covered by a 32-bit message counter, which is not critical in either chip area or circuit timing. Counter overflow triggers a fault

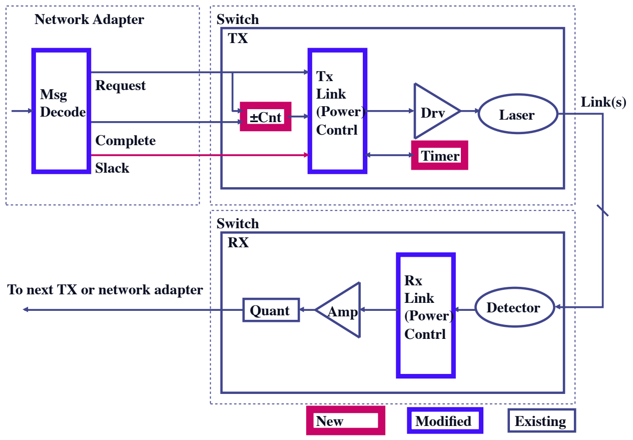

Figure 4: Hardware support for link management.

signal to the switch firmware.

In practice, the power-down of a link can start immediately or be delayed for a short period of time using a *timer* (Section 2.1.2) in case a new message will be passing through very soon. A particular low-power state can also be selected based on the information contained in the *slack* time signal (Section 2.1.2).

2.1.2 Hardware Support

Fig. 4 illustrates the hardware support for the thrifty network link management. We use optical links as an example, since it is more popular than electrical links to build large-scale interconnection networks. The transmitter module includes two new components, a message counter and a timer. In addition, power-aware support is required in the message decoder of the network adapter, the sender link controller and the receiver link controller. Other components are largely unchanged, such as the driver and laser in the transmitter module, the photo detector, amplification and quantization units in the receiver module.

The message decoder in the network adapter generates three signals to the link power controller. The message request signal is triggered when the sender compute node sets up the network adapter using a series of special writes (step 1 in Section 2.1), which will be followed by the network address that relates to the link paths (step 2 in Section 2.1). The message completion signal is typically triggered by the tail of a message. A *slack* signal infers the inherent system overhead. It is predetermined by the choice of the intra-node communication protocol, which indicates the slack time between the command to send data and the completion of assembling and packetizing the data, i.e., the latency of moving data to the network adapter. It provides a hint to the link power controller to choose a low-power state for the installed system, such that the link activation completes before data transfer happens.

The message counter monitors the propagation of the data messages through the switch. The counter is incremented by the assertion of a message initiation signal and decremented by the assertion of a message completion signal. A zero output signal of such a counter is indicative of a completion of the message flow (i.e., data traffic) through the switch. In case of application errors or compute node failures, the message counter may be in incorrect states. We assume the recovery mechanism at the faulty source is able to send a correction message (i.e., extra initiation or completion signals) to inform the downstream switches to adjust the message accounting, albeit after some delay. If a disastrous failure happens, the whole system including TIN will have to be restarted. Protocols and mechanisms to efficiently handle fault tolerance in TIN are interesting areas for further study.

The timer receives the output count signal of the message counter. In response to a zero output count signal, after a pre-determined time delay, it produces a signal, which prompts the power controller to change the power state of the associated link (for example, to switch the link from an operational state to a low-power state).

The transmitter link power controller receives link initiation signal (message request), link release signal from the message counter, and a slack time signal from the timer. The power controller can put a link in an operational (i.e., full power) state or one of the predetermined low-power states. It also passes the link initiation, link release and slack time signal to the next hop via the control network.

2.1.3 Software Extension

It is possible that some systems may not have sufficiently long intervals in the intra-node protocol to completely overlap the time required to initiate a link into the operational mode from certain deep low-power mode, e.g. one that requires firmware support. In that case, prolonged link transition time will adversely affect performance. This activation delay may be reduced or eliminated if run-time or programs are able to issue link activation commands sufficiently earlier. In this section, we discuss how software support for link activation and release can be implemented in MPI. We propose two new MPI primitives: one to activate a communication path (LINK_ACTIVATION) and one to release it (LINK_RELEASE). They can be generated by certain MPI run time system as in [24], or implemented as macros in MPI code. In this paper, we use macros in MPI source code as a case study.

Fig. 5 shows the pseudo MPI codelet for the software initiation of the thrifty network. A LINK_ACTIVATION macro with tag L is placed before MPI_CALL_A to wake up the corresponding links. The programmer needs to place the command properly to ensure links are fully activated before data transmission; otherwise, the system may run slower due to insufficient leeway in the hardware. This special command triggers a library call that bypasses the

```
......
LINK_ACTIVATION(L)  /* Command to inform network adapter to activate links that MPI_CALL_A will use. */
......
MPI_CALL_A( , , , )
LINK_RELEASE(L, SHUTDOWN)  /* Command to hint network adapter to shutdown corresponding links. */
......
LINK_ACTIVATION(M) /* To activate a different set of links. */
......
MPI_CALL_B( , , , )
LINK_RELEASE(M, KEEP_ON) /* If too close together, this command and the next, LINK_ACTIVATION(M),
can be eliminated. */
......
LINK_ACTIVATION(M)
......
MPI_CALL_C( , , , )
LINK_RELEASE(M, SHUT_DOWN)
LINK_ACTIVATION(N) /* To activate a different set of links. */
......
MPI_CALL_D( , , , )
LINK_RELEASE(N, SHUT_DOWN)
......
```

Figure 5: Pseudo MPI codelet for the software initiated link activation/de-activation.

MPI protocol layers and directly communicates with the network adapter. The network adapter then issues a special single-flit message to the local switch for the proposed on-demand (data) link services, via the always-on control network. A LINK_RELEASE macro with the same tag L is placed right after the MPI_CALL_A, which decrements the message counters (Section 2.1.2). However, link shutdown is only triggered by the link message counter in the hardware (Section 2.1.2). *Because multiple software threads may communicate via the same link at run time, only the hardware message counter can guarantee all the traffic over the link is completed.*

If two adjacent MPI calls share the same communicator and rank, e.g., the same tag M in Fig. 5, the programmer can merge the adjacent LINK_RELEASE and LINK_ACTIVATION macros to reduce unnecessary link transitions. Therefore, a pair of LINK_ACTIVATION and LINK_RELEASE can specify the start and end of a communication phase that consists of multiple message transmissions.

2.2 Network Power Shifting

The thrifty network not only offers potential for power reduction in the network, but also opens the opportunity for extending the idea of power shifting within a compute node [16] to power shifting between compute nodes and their interconnection network. Power shifting was originally proposed to trade off performance for reducing peak power consumption. In this work, we instead investigate the potential of shifting the excess power budget from the interconnection network back to the compute nodes for better performance improvement.

In this case, the stranded power saved by the network can be reused by compute nodes to achieve higher chip frequency and application performance. A related example is Intel's Nehalem® processor which can shift power away from underutilized cores and boost the frequency of the remaining core(s) by a large margin (60% or even more). Nehalem can optionally support chip-level frequency boost for all the cores and still do not exceed the thermal design

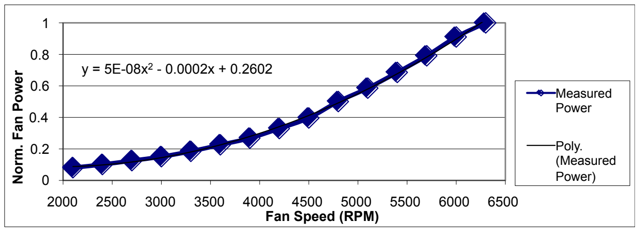

Figure 6: Measured fan power of a state-of-the-art system with series fans and its trend line.

power of the whole chip. Circuit timing due to overclocking can still be maintained as indicated from some Nehalem results. In our network power shifting mechanism, power delivery constraints are also met. This is because typically only a portion of the excess power from network nodes can be reclaimed by the compute nodes, while the two typically share the same power supply.

A critical issue to apply power shifting for performance improvement is the thermal constraint of the computing node that is limited by its cooling capacity. We propose to include cooling power in the overall node (compute and adapter/switch/links) power budget and shift the excess cooling power from an idle network node to its associated computing node as well. In particular, we take advantage of the reliability-aware redundant fan design in most modern server systems by turning on extra fan(s) for additional cooling capacity and increased processor frequency, under the same overall system power budget. The additional cooling power for the compute nodes comes from the excess cooling power shifted from the idle network switches and links. Fig. 6 shows the relationship between the measured fan power (normalized to its maximum) and speed in a state-of-the-art server system. Its trend line is also shown as a polynomial equation. Additionally, cooling capacity is proportional to fan speed. We then apply these relational functions to our simulated system configuration in our evaluation (Sections 3 and 4). With a cooling configuration of up to two series fans in operation, our analysis shows that up to 36% chip-level frequency boost can be achieved. Yet, we conservatively assume a two-fan cooling capability of up to 25% chip-level frequency boost in our evaluation.

In addition to the enhancement in the cooling system, we also apply two on-chip Vdd supplies, one for the nominal frequency and the other for the up-to 25% chip-level frequency boost. Such a multi-Vdd design significantly reduces the DVS transition time to well under one microsecond [4, 13, 21]. To fully explore the benefits of power shifting, the voltage regulator modules of the network components and particularly the computer nodes may have to be over-provisioned.

Power shifting can be done between a pair of sender and receiver and their interconnection network path, between a group of senders and receivers, or among the whole interconnected system. However, since the links in the data (and control) network typically do not follow the same topology as the power supply distribution network, it can be difficult to determine to which compute node (if any) a link's power budget can be shifted. Therefore, we consider network power shifting at the granularity of a stage, which includes one or more compute nodes and their attached network components (adapter, switch and links), in a multi-stage network path. In other words, a stage in a network path, is equivalent to a node machine in an interconnected cluster of machines.

When all the traffic has completely passed through the first stage of the network path, its network adapter triggers TIN to transition the locally attached network components (switch and links) into a low-power mode. At the same time, the network adapter informs the power management mechanism in the locally attached compute node(s) to crank up the processor chip frequency using the aforementioned multi-Vdd frequency scaling mechanism. The same process repeats at the remaining stages of the multi-stage network path, until it ends at the destination stage. On the other hand, when the network traffic is about to happen, the network adapter signals the locally attached compute nodes to scale back to nominal frequency before signaling the locally attached network components to power on, in order to maintain the overall power budget. The compute nodes keep operating during frequency scaling.

We consider an open loop control in this work. Therefore, the extra firmware overhead for run-time power measurement, which can be over a millisecond, can be avoided. Without a feedback control, the network adapter maintains an estimate of the power usage as network components are powered on and off over the time. It sends proper signals to its local compute nodes to request speed change in the processor such that the system peak power budget is not violated. This mechanism requires built-in hardware support for estimating the peak power of a network link and the peak power for the processor running at nominal and frequency-boost modes. This is similar to the power estimates that manufacturers publish in data sheets today.

3 Evaluation Method

3.1 Simulation Framework

We use the MARS simulation framework [12] to evaluate our designs. MARS is an end-to-end trace-driven simulation framework that can simulate systems with hundreds of thousands of interconnected processors. MARS simulates both the processors and the interconnection network in detail. It features several network topologies, flexible routing schemes, arbitrary application task placements, point-to-point statistics collection, and data visualization support. MARS itself is a parallel program that runs over MPI. It has been used for the performance evaluation of two industry

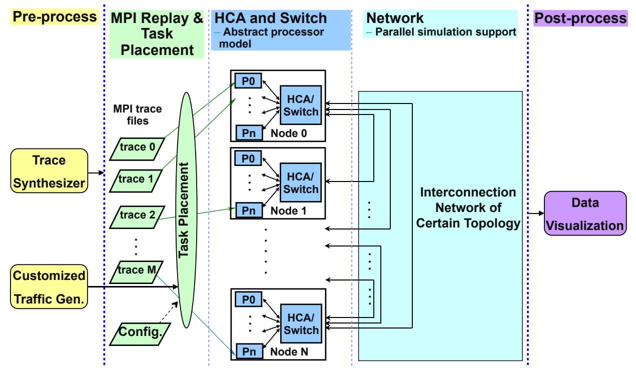

Figure 7: MARS: An end-to-end simulation framework for large-scale systems.

large-scale systems.

Fig. 7 illustrates the components of MARS. The core part of MARS is a compound module of detailed HCA (Host Channel Adapter, a.k.a., network adapter) and switch simulation models, which have an interface to a processor model. A few processors can share one HCA via a shared bus and form an SMP node. The input to MARS can be MPI traces collected from real hardware, traces generated by a trace synthesizer, or a customized traffic generator. More than one trace file (e.g., MPI task) can feed one processor. A task placement module handles the dispatch of traces to processors. The output of the simulation can include both summary and detailed statistics from each node. The statistics, e.g. network traffic patterns, can be analyzed by post-process tools and be visualized as movie clips.

In the MPI trace-driven simulation, MARS respects the communication dependency between MPI tasks, models the acknowledgments in MPI handshaking and other MPI protocol operations, which affects the semantic flow and timing of actions across the network and among compute nodes, in addition to the network simulation. The wall-clock timing dependencies in the collected traces do not affect the simulated semantic flow. Therefore, MARS is essentially an execution-driven simulator of sequences of MPI commands (not compute instructions) on top of the cycle-accurate network simulation. We annotate the traces to trigger activation events and generate special single-flit messages for link operations, taking into account the transition time of a low-power link state. In the simulation, a second link activation message collides with a first one if the link is still on. We made no changes to the assumptions about the underlying MPI library used in the system.

3.2 System Configuration

In this paper, we evaluate our techniques with a hierarchical direct interconnect architecture that has a point-to-point three-level complete graph topology, as illustrated in Fig. 8. A version of such a topology has been implemented in the IBM PERCS system for the DARPA funded HPCS program [8].

Each compute node is a multi-core chip in this system. A switch module is associated with one or a few computing nodes. A small set of switch modules are directly and fully interconnected via the first level links (level-1 links), forming first-level groups, e.g., boards. Multiple boards are directly and fully interconnected via a second level links (level-2 links), forming second-level groups, e.g., racks. Finally, multiple racks are directly and fully interconnected via the third level links (level-3 links), forming third-level groups, e.g., the full system.

Such a hierarchical complete-graph topology tries to balance the number of hops to reach other nodes and the total number of links required to build the network. Each network packet traverses at most one global channel, level-3 links in this case. By reducing global channels, such a hierarchical direct interconnect architecture can significantly reduce cost compared to other conventional topology such as Fat Tree, while maintaining the same bisection bandwidth. A general form of the topology has been studied by Kim et al [20].

A property of this architecture is that routing between arbitrary nodes may involve multiple intermediate hops via links at the different hierarchy levels. For example, to reach a destination in another rack, a level-1 link hop may be required to reach the appropriate switch module that has a level-2 link to the right first-level group (board) that has a level-3 link to the right second-level group (destination rack) and vice versa on the destination rack. Fig. 9 shows an example of the longest direct route that is possible in the system. In the figure, Node S is the source node, Node D is the destination node, Node I1-I6 are the intermediate nodes, L1 is a level-1 link, L2 is a level-2 link, and L3 is a level-3 link. In this case, a message traverses the network with seven hops passing one level-3 link, two level-2 links, four level-1 links, and six intermediate nodes. The reason for intermediate boards, Board I1 and I2, is that only these two boards may have the level-3 link connection between source and destination racks.

The size of the full-up system can be calculated as follows. If we leave out one level-1 port for simplicity, the number of compute nodes in a board is the same as the number of level-1 links (NL1) of a compute node, since all the compute nodes in a board are fully connected. Since each compute node has a number of level-2 links, the number of boards in a rack is the number of level-1 links (NL1) of a compute node times the number of level-2 links (NL2) of a compute node, (NL1)x(NL1), forming a two-level complete graph. Furthermore, the number of compute nodes in a rack is (NL1)x(NL2)x(NL1). Since each compute node also has a number of level-3 links, the number of racks in the full-up system of a three-level complete graph is the number of level-3 links (NL3) of a compute node times the number of

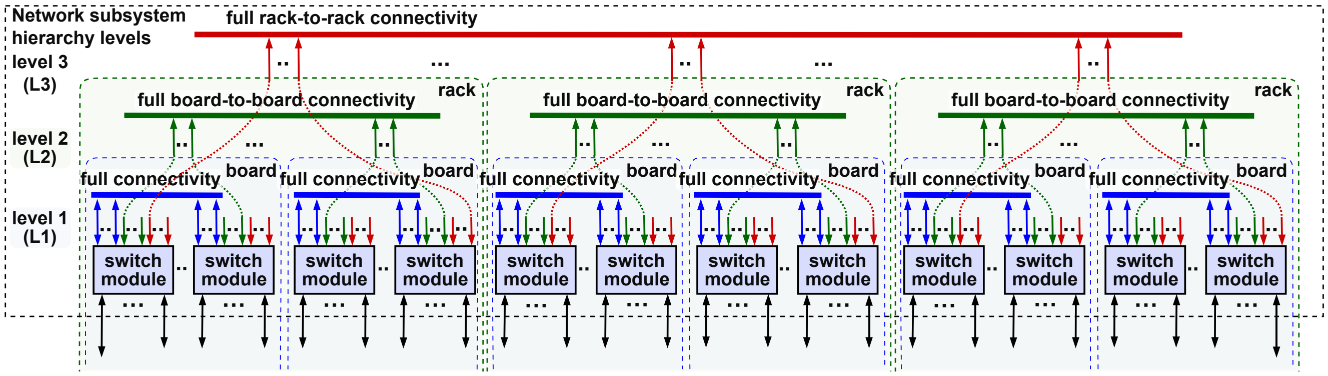
Network subsystem hierarchy levels
level 3 (L3)
level 2 (L2)
level 1 (L1)
full rack-to-rack connectivity
full board-to-board connectivity
rack
full connectivity
board
switch module

Figure 8: A hierarchical direct interconnect architecture with a three-level complete graph.

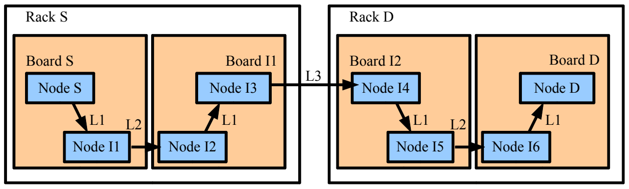

Figure 9: Example direct route in the hierarchical network topology of a three-level complete graph. S, I and D represent Source, Intermediate and Destination, respectively. E.g., "Node I1" means the first intermediate node.

Table 1: System configuration

Processor	POWER5-like @ 2 GHz with caches, 45 nm, 0.65v and 0.95v supply voltages, 120 W peak power
Switch	3-stage with buffers @ 2 GHz, 45 nm, 50 W peak power including link power 11 ports (4 + 2 + 4 + one to local compute node)
Level-1 link	4 @ 2 GB/s bidirectional, 3 ns latency, 3 feet long
Level-2 link	2 @ 2 GB/s bidirectional, 6 ns latency, 6 feet long
Level-3 link	4 @ 1 GB/s bidirectional, 30 ns latency, 30 feet long
Control link width	1/8 of its companion data link
Optical transceivers	3 W peak power
Memory/storage/etc	80 W peak power

compute nodes in a rack, (NL1)x(NL2)x(NL1). Therefore, the number of racks in the full-up system equals to (NL3) x (NL1)x(NL2)x(NL1). Finally, the size of the full-up system is the number of nodes in a rack times the number of racks in the full-up system, i.e., (NL1)x(NL2)x(NL1) x NL3 x (NL1)x(NL2)x(NL1) nodes.

The main parameters for the simulated system are listed in Tab. 1. The simulated processor model is an abstraction of an IBM POWER5-like processor®, with 2 GHz cores at 45 nm processor technology and corresponding cache memory. Apparently, setting nominal frequency at 2 GHz is very low for what 45 nm technology can achieve. The reason is that we expect future power-efficient HPC systems to be built with lower-frequency and power-efficient processors, similar to IBM's BlueGene® approach. Extrapolating from Intel Nehalem's turbo mode specification and our own experiments in Section 2.2, we assume the compute node can be over-clocked by up to 25% to 2.5 GHz. We scale processor frequency with two Vdd supplies, 0.65v and 0.95v for 2GHz and 2.5GHz, respectively. The switching delay between the two frequencies is well under $1\mu s$ [4, 13, 21]. The switches also run at 2 GHz. In this experimental setup under study, one switch chip is connected to one compute node for simplicity. Recall the topology in Fig. 8 and the analysis above, these parameters represent a system of up to (NL1)x(NL2)x(NL1) x NL3 x (NL1)x(NL2)x(NL1) = (4x2x4) x 4 x (4x2x4) = 4096 nodes connected by the corresponding three-level hierarchical direct network.

We carefully choose three low-power modes to quantify their power-performance tradeoffs, by consulting appropriate literature and industry data sheets [1, 9, 19, 22, 28, 34]. We assume optical links are deployed in the target system, each of which along with its transceivers consumes three watts [1]. Regarding link power management, links are re-synchronized at each activation and it may take a few hundred nanoseconds depending on the system size. On the other hand, powering down and up the majority of a (switch/router) chip in an SMP node with firmware support can take up to a millisecond. We conservatively take this assumption from existing server designs, while the exact time may vary dependent on the SMP node size. Therefore, we choose three transition time: $1\mu s$, $2\mu s$ and $1ms$. They correspond to three low-power modes of increasing power savings and transition delay: (1) power-gated links where the power supply to the links are shut off to save both dynamic and static power of the links of a switch (Low-Power Mode 1, LPM1); (2) clock-gated switch buffers and power-gated crossbar in the switches (Low-Power Mode 2, LPM2), in addition to power-gated links in LPM1; and (3) all network components, including all switch buffers, crossbars and links, are power-gated (Low-Power Mode 3, LPM3). A power-gated link (LPM1) means all data transmission related components are power off and consume zero power, except that the control link and its logic are always on, which use one-eighth of their corresponding data links width and consume one-eighth of the corresponding data link power. Recall that activation and release messages go through the control links. (Our evaluation in Section 4 takes this into account.) In LPM2, a switch with clock-gated buffers and power-gated crossbar consumes 50% of the nominal switch power [22, 28]. In LPM3, a power-gated switch (LPM3) consumes almost zero power and we ignore the residue power of the remaining monitor logic. Power consumption during a transition is modeled as a linear function between its start and end power states. We further assume 64% of the nominal switch power comes from link power. We also conservatively assume 20% of total system power budget is allocated to the interconnection network at the design phase for future large-scale systems [3, 8, 23]. Peak power consumption by main memory, storage and other node components are 80 W.

Since the overhead of frequency scaling with dual voltage supply is much smaller than the aforementioned link transition time, the chip frequency-boost transition time of processors is contained in the link transition time in our simulation.

We use MPI traces collected from a few supercomputing centers to drive the simulations. These workloads are: AMBER [10], GYRO [15], HYCOM [17], LAMMPS [31], POP [33], SPPM [27], UMT2K [25] and WRF [26].

4 SIMULATION RESULTS

In this section, we first evaluate the thrifty interconnection network and network power shifting techniques on a simulated 32-node cluster partition that runs 32 MPI tasks for each workload. In this 32-node partition, only level-1 and -2 links are used for data communication. We then use WRF, a popular weather forecast workload, to explore the power-performance of TIN and NPS in a larger-scale system, where all three levels of links are populated for data communication. We allocate one MPI task per compute node and use direct routes in the simulation experiments.

In addition to LPM1, LPM2 and LPM3 (Section 3.2), we add an Ideal low-power mode in this study. Ideal means both zero transition delay and zero power consumption in the links and switches. It also applies zero transition delay for the voltage and frequency scaling in the compute nodes. Ideal serves as the upper bound. To further understand the trade-off, in the following we also plot results of an "intelligent" scheme that always chooses the best of LPM1, LPM2 and LPM3. Effectively, this is the minimal power and energy consumption and maximum performance of the three. This scheme is the upper bound of adaptive power management that can choose one of the three low power modes at run time. Therefore, we call it Optimal Adaptive (O-Adaptive).

4.1 Network Power Reduction with TIN

Fig. 10 shows the average power consumption of TIN, normalized to the original network power without TIN support, in a 32-node system. The ideal thrifty network power is only 18% of the nominal power. This is because these

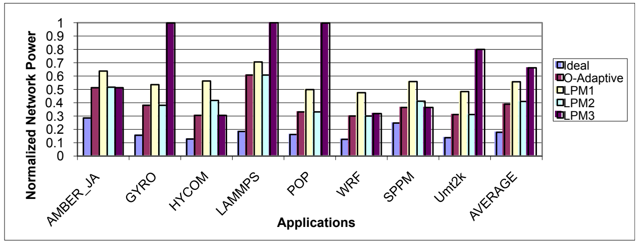

Figure 10: Normalized average power consumption of the thrifty interconnection network (lower is better).

applications present high parallel efficiency; therefore, the network is idle for most of the time. When the link activation/deactivation overhead is included, the network power increases to around 56%, 41% and 66% of the nominal power for LPM1, LPM2 and LPM3, respectively. Interestingly, LPM2 achieves the lowest overall power consumption among the three, even though its power consumption for each individual application may not be the lowest. Since LPM2 with 2us transition latency (1us one-way) performs best, it turns out that the software initiated network link service was hardly invoked to achieve the best power-performance on the studied workloads. In these workloads, the cut-off LPM latency for software initiation is around 10us. On the other hand, we expect software initiated link activation commands to be conservative to avoid shorter than expected time to activate links.

AMBER, HYCOM and SPPM prefer LPM3, which has the most power reduction even with the highest overhead. This is because these applications are dominated by long message intervals. However, GYRO, LAMMPS and POP have no power reduction at all with LPM3, due to dominating short message intervals.

If multiple low-power modes are allowed to adapt to the varied application communication patterns, further power reduction can be achieved, as the O-Adaptive scheme shows. Nonetheless, LPM2 alone achieves very close power savings to O-Adaptive, thanks to its balanced moderate transition time and power consumption.

Note that, our results do not include the savings for reducing the cooling system load and are therefore relatively conservative. In typical data centers, removing 1W of IT power reduces cooling power by 0.2W to 1W, depending on the efficiency of the data center facilities. Therefore, we expect more savings in data-center power and energy bill, when the cooling system power is included. On the other hand, we do utilize cooling power shifting for enhanced frequency boost as discussed in Section 2.2 and evaluate it next in Section 4.2.

4.2 Performance Improvement with TIN and NPS

Fig. 11 shows the application performance improvement with TIN and NPS, in a 32-node system. The extra power budget from the thrifty interconnect network is reinvested

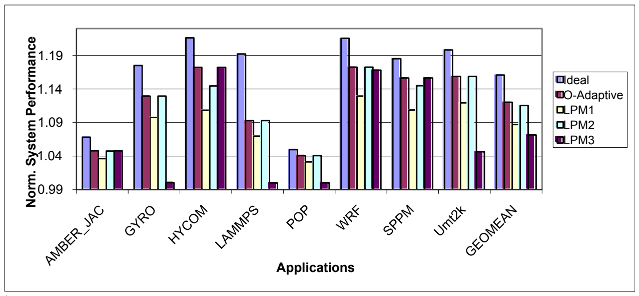

Figure 11: Normalized performance with power shifting over thrifty network (higher is better).

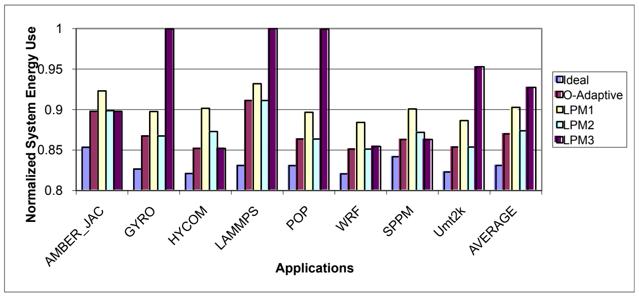

Figure 12: Normalized system energy use with power shifting over thrifty network (lower is better).

onto the compute nodes with up to 25% higher chip frequency. This corresponding compute speedup is bounded by frequency boost since memory access does not speed up accordingly. As a result, an average speedup of almost 17% (22% maximum with HYCOM and WRF) is obtained in the ideal case. Similar to the network power consumption in Section 4.1, LPM2 achieve an average performance boost of 12% that is very close to O-Adaptive.

Many of the power-performance results show that not all the extra power budget shifted from the interconnection network is used by the processors in the compute nodes, since we assume the frequency boost can only be up to 25% higher than the nominal. We expect more performance gain if this limitation is relaxed (Section 2.2) and the compute nodes can speed up more. Another interesting fact that may contribute to the sub-optimal performance improvement is an opportunity cost that, the speedup of a compute node in one compute phase effectively reduces the corresponding message inter-arrival time on its links, which leads to fewer opportunities for the same links to transition into a low-power mode. Nonetheless, our statistics reveal insignificant effect of this opportunity cost for the studied workloads and system setup.

Since we only increase chip frequency beyond its nominal, we do not observe compute performance loss when a compute node cannot boost its processor frequency in time. On the other hand, as long as the links are woken up in time with intrinsic system events for LPM1 and LPM2 or software hints for LPM3, data communication over the links is not delayed. As a result, we do not observe performance loss with TIN and NPS for the studied workloads.

Note that, it is also interesting to consider distributing excess power to other nodes when reaching a nodes frequency cap. We haven't experimented with this approach for two reasons: (1) Boosting the frequency of other nodes that are probably not in the critical path will not improve overall performance; (2) there is a potentially high overhead of tracking and scheduling other nodes not on the same paths.

4.3 System Energy Improvement with TIN and NPS

The power saved by the thrifty interconnection network cannot always be completely re-invested in the compute node speedup, due to the cap of 25% frequency boost. In addition, the performance improvement with power shifting shown in Fig. 11 reduces application execution time. As a result, total system energy is reduced.

Fig. 12 shows the system energy use with TIN and NPS, which is normalized to the original system, in a 32-node system. Since the performance, i.e., execution time, varies modestly among applications and their thrifty network modes (Section 4.2), the system energy comparison across them shows a profile that is similar to the network power comparison (Fig. 10 in Section 4.1), but to a smaller scale. Similar to Section 4.2, LPM2 achieves very close energy reduction of 13% to O-Adaptive.

4.4 Scalability Study

We use WRF, a popular weather code, as a case study for larger-scale systems. We conservatively choose a few strong scaling WRF traces, where the WRF problem size remains the same when the number of MPI tasks increases, to study the scalability of TIN and NPS. Note that the WRF problem size in this large-scale study is larger than the WRF problem size in previous sections. Fig. 13A, 13B and 13C shows the network power, application performance and overall system energy, respectively, of WRF with 512, 1024, 2048 and 4096 MPI tasks. For fair comparisons, the network power and system energy are calculated only for the compute and switch nodes that are in use.

Since we use strong scaling traces, the network communication becomes more pronounced in larger scale runs. As a result, network power usage with TIN steadily increases, yet with a much slower pace than the MPI tasks increase. The application performance improvement of NPS also decreases, due to the smaller weight of compute phase in overall execution time. As a result, the overall system energy use also gradually increases. We, therefore, expect TIN and NPS to be more effective in weak scaling than strong scaling WRF runs.

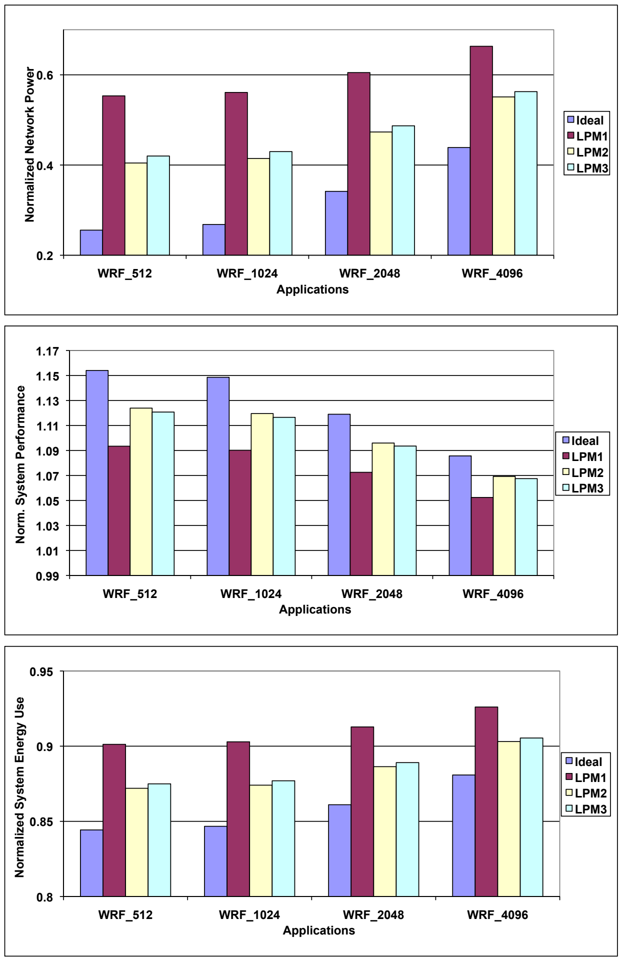

Figure 13: Network power (top, A, lower is better), system performance (middle, B, higher is better) and system energy (bottom, C, lower is better) of WRF with 512, 1024, 2048 and 4096 MPI tasks.

5 Related Work

Lim et al. proposed an MPI run time system that dynamically reduces CPU performance during communication phase of MPI programs [24]. The run time system identifies communication phases and selects processor frequency in order to minimize energy-delay product, without profiling or training. The analysis and subsequent voltage/frequency scaling are done within MPI. In our work, we shift the saved power budget from the thrifty interconnection network to boost the compute node frequency for better application performance. Our software support for the thrifty network can be also be implemented in an MPI run time system.

Peh and Dally proposed flit-reservation flow control, in which control flits traverse the network, reserve buffers and channel bandwidth before data flits arrive [29]. Duato et al. proposed to use probes that traverse a separate control network, set up circuits on the data network for subsequent messages and improve performance by removing flow control mechanisms from the data network [14]. In our work, the separate control network is used to trigger the activation and shutdown of the data network to improve network power efficiency without performance loss.

Kim et al. [19] proposed to use adaptive routing to route data traffic to a smaller set of links in *space* when the overall link utilization is low. The unused links can then be dynamically shut down to save power, while the active links operate at higher utilization. In our work, we propose to utilize the *time* slack between command and data to hide the power-mode transition delay of the same data link. The two approaches are complementary to each other.

Shang et al. proposed a hardware-based prediction mechanism to change the supply voltage and frequency of links dynamically as a function of traffic [32]. History-based prediction is used. Links never shut down to maintain periodical pulse transmission to keep both ends of a link synchronized. Soteriou and Peh also explored the design space of power-aware interconnection networks [34].

Alonso et al. proposed hardware-based mechanisms, as a function of traffic, to switch on and off all but one active links in a path. The always-on link in a path maintains the connectivity in the network. All the links together provide multiple alternative paths for the source and destination pairs [6, 7]. In our thrifty interconnection network, network links are activated and released with the transition time overlapping the inherent system events.

Pelley et al. proposed to reorganize power feeds to create shuffled power supply distribution topologies, and use *Power Routing*, a data-center level mechanism to schedule workload dynamically across redundant power feeds [30]. In our work, power shifting operates *locally* between a compute node and its attached network switch and links.

6 Conclusions

We have presented two complementary techniques to tackle the looming power crisis in future cluster systems. First, we propose *Thrifty Interconnection Network* (TIN), where the network links are activated and released with the transition time overlapping the inherent system events. Second, we propose *Network Power Shifting* (NPS), which dynamically shifts the total power budget between the compute nodes and its local network components. Our simulation results on an industry-strength simulation infrastructure with a set of real-world HPC workload traces show that, TIN can achieve on average 60% network power reduction, with the support of only one low-power mode, i.e., LPM2. When NPS is enabled, the two together can achieve 12% application performance improvement and 13% overall system energy reduction. Further performance improvement is possible if the compute nodes can speed up more and fully utilize the extra power budget reinvested from the thrifty network with more aggressive cooling support.

Acknowledgments

We owe a debt gratitude to Ram Rajamony for his guidance and support. Thomas F. Wenisch, Don DeSota, Evan Speight, Mark W. Stephenson, Edgar A. Leon Borja, Anthony N. Hylick, Chris Craik and Kshitij Sudan have offered helpful insights to refine the ideas in this work. We also appreciate the insightful comments and constructive suggestions from the anonymous reviewers. This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0002. Details presented in this paper may be covered by existing patents or pending patent applications.

References

[1] Avago Technologies data sheet of optical transmitters, receivers and transceivers. *http://www.avagotech.com/*.

[2] Don DeSota, Personal Communication, IBM System Technology Group.

[3] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. In *ISCA*, pages 338–347, 2010.

[4] K. Agarwal and K. Nowka. Dynamic power management by combination of dual static supply voltages. *ISQED*, pages 85–92, 2007.

[5] U. E. P. Agency. Report to Congress on Server and Data Center Energy Efficiency. *U.S. Environmental Protection Agency*, 2007.

[6] M. Alonso, S. Coll, J. M. Martínez, V. Santonja, P. López, and J. Duato. Dynamic power saving in fat-tree interconnection networks using on/off links. In *IPDPS*, 2006.

[7] M. Alonso, J. M. Martínez, V. Santonja, P. López, and J. Duato. Power saving in regular interconnection networks built with high-degree switches. In *IPDPS*, 2005.

[8] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, N. Ni, and R. Rajamony. The PERCS High-Performance Interconnect. In *Hoti: Proceedings of the 18th Annual Symposium on High-Performance Interconnects*, 2010.

[9] A. F. Benner, M. Ignatowski, J. A. Kash, D. M. Kuchta, and M. B. Ritter. Exploitation of optical interconnects in future server architectures. *IBM J. Res. Dev.*, 49(4/5):755–775, 2005.

[10] D. A. Case, T. T. C. III, T. Darden, and et al. The Amber biomolecular simulation programs. *J. Computational Chemistry*, 26(16):1668–1688, 2005.

[11] DARPA. Ubiquitous High Performance Computing (UHPC) Broad Agency Announcement (BAA). 2010.

[12] W. E. Denzel, J. Li, P. Walker, and Y. Jin. A framework for end-to-end simulation of high-performance computing systems. In *International conference on Simulation tools and techniques for communications, networks and systems (SIMUTools)*, pages 1–10, 2008.

[13] L. Di, M. Putic, J. Lach, and B. H. Calhoun. Power Switch Characterization for Fine-Grained Dynamic Voltage Scaling. In *ICCD*, pages 605–611, 2008.

[14] J. Duato, P. Lopez, F. Silla, and S. Yalamanchili. A high performance router architecture for interconnection networks. In *Proc. Int. Conf. On Parallel Processing*, pages 61–68, 1996.

[15] M. R. Fahey and J. Candy. GYRO: A 5-D Gyrokinetic-Maxwell solver. *SC Conference*, 2004.

[16] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *ICS*, pages 293–302, 2005.

[17] G. Halliwell, R. Bleck, and E. Chassignet. Atlantic Ocean simulations performed using a new hybrid-coordinate ocean model. In *EOS Transactions. American Geophysical Union (AGU), Fall 1998 Meeting*, 1998.

[18] IBM. IBM InfiniBand 8-port 12x switch. *http://www-3.ibm.com/chips/products/infiniband*.

[19] E. J. Kim, K. H. Yum, G. M. Link, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Yousif, and C. R. Das. Energy optimization techniques in cluster interconnects. In *ISLPED*, pages 459–464, 2003.

[20] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *ISCA*, pages 77–88, 2008.

[21] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *HPCA*, 2008.

[22] A. K. Kodi, A. Sarathy, and A. Louri. iDEAL: Inter-router Dual-Function Energy and Area-Efficient Links for Network-on-Chip (NoC) Architectures. In *ISCA*, pages 241–250, 2008.

[23] P. M. Kogge. Architectural Challengies at the Exascale Frontier (invited talk). In *Simulating the Future: Using One Million Cores and Beyond*, 2008.

[24] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal. MPI and communication - Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs. In *SC*, page 107, 2006.

[25] M. M. Mathis and D. J. Kerbyson. A general performance model of structured and unstructured mesh particle transport computations. *J. Supercomput.*, 34(2):181–199, 2005.

[26] J. Michalakes, J. Dudhia, D. Gill, and et al. Design of a next-generation regional weather research and forecast model. In *Proceedings of the Eighth Workshop on the Use of Parallel Processors in Meteorology*, pages 16–20, 1998.

[27] A. A. Mirin, R. H. Cohen, B. C. Curtis, and et al. Very high resolution simulation of compressible turbulence on the IBM-SP system. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, page 70, New York, NY, USA, 1999. ACM.

[28] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers. In *MICRO*, pages 333–346, 2006.

[29] L.-S. Peh and W. J. Dally. Flit-Reservation Flow Control. In *HPCA*, pages 73–84, 2000.

[30] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood. Power routing: dynamic power provisioning in the data center. In *ASPLOS*, pages 231–242, 2010.

[31] S. J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Computational Physics*, 117:1–19, 1995.

[32] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *HPCA*, 2003.

[33] R. D. Smith, J. K. Dukowicz, and R. C. Malone. Parallel ocean general circulation modeling. *Phys. D*, 60(1-4):38–61, 1992.

[34] V. Soteriou and L.-S. Peh. Design-space exploration of power-aware on/off interconnection networks. In *ICCD*, pages 510–517, 2004.