A Feasibility Analysis of Power-Awareness and Energy Minimization in Modern Interconnects for High-Performance Computing

Reza Zamani[1], Ahmad Afsahi[2], Ying Qian[3], Carl Hamacher[4]

Department of Electrical and Computer Engineering, Queen's University
Kingston, ON, Canada K7L 3N6

[1]reza.zamani@ece.queensu.ca
[2]ahmad.afsahi@queensu.ca
[3]ying.qian@ece.queensu.ca
[4]carl.hamacher@queensu.ca

***Abstract*— High-performance computing (HPC) systems consume a significant amount of power, resulting in high operational costs, reduced reliability, and wasting of natural resources. Therefore, power consumption has become an increasingly important design constraint in high-performance clusters. In this regard, research on power-aware HPC has emerged. While most research has focused at understanding and utilizing applications' behavior to scale down the CPU for energy savings, this paper demonstrates the positive impact of modern interconnects in delivering energy-efficiency in high-performance clusters. In this work, we first present the power-performance profiles of the Myrinet-2000 and Quadrics QsNet^{II} at the user-level and MPI-level in comparison to a traditional, non-offloaded Gigabit Ethernet. Such information enables us to devise a power-aware MPI runtime library that automatically and transparently performs message segmentation and re-assembly in order to increase energy savings. Secondly, by designing and evaluating a number of all-gather collectives, we argue that it is possible to increase the energy-efficiency of a cluster by optimizing its messaging layers.**

I. Introduction and Motivation

Research in the area of advanced computer architecture has been primarily focused on performance. Nowadays, *high-performance computing* (HPC) systems consist of clusters of hundreds to tens of thousands of servers tackling supercomputing applications as well as supporting low-end applications. However, because the computing, storage, memory, and networking components of a high-performance cluster consume a significant amount of power, power consumption has become an increasingly important constraint in HPC centers. Power consumption is critical since it also affects the cooling systems, uninterruptible power supplies and backup power generations. One should also take into account the energy for air circulation and power delivery. Moreover, if the nodes are not cooled down aggressively, the temperature rises rapidly doubling the failure rate for every 10° C increase in temperature [16, 9].

Ignoring power consumption as a design constraint has resulted in high operational and maintenance costs as well as reduced reliability in HPC centers. Not to mention, it wastes natural resources and pollutes the environment. Therefore, it has become vital for researchers in academia and industry to come up with novel ideas to manage power and energy consumption in high-performance clusters. In this regard, two distinct directions have been proposed: (1) a low-power, architectural approach, and (2) a power-aware approach.

Along the first path, some organizations have invested in building HPC platforms by adopting low-power processors. *Green Destiny* [36] was an experimental, low-power and compact cluster at Los Alamos National Laboratory. *MegaProto* [30] is a prototype cluster based on low-power Transmeta processors that could deliver reasonable performance. Perhaps, the most famous, low-power HPC machine is the *IBM BlueGene/L* [1], a massively parallel computing system built with PowerPC processors. However, there has been some concern within the research community that such low-power architectural approaches sacrifice too much per-node performance to achieve their low-power goals.

To address the problems with the first approach, research in power-aware HPC [24],[35],[20],[16],[9]-[15],[22],[17],[8] has emerged. This approach is largely architecture-independent, and starts with power-hungry, high-performance commodity components, and makes them operate in low-power by devising power-aware algorithms, applications, middleware, and run-time systems. Power management schemes try to transition the hardware components, such as the CPU, from high-power mode, where they are active, to low-power mode when they are idle.

Dynamic voltage and frequency scaling (DVFS) and *clock throttling* are known as the most effective methods to reduce the CPU power consumption, unfortunately at the expense of performance degradation. In fact, the semiconductor industry has introduced such technologies into their chips, known as *SpeedStep* technology from Intel and *PowerNow!* and *Cool'n'Quiet* from AMD. Many works have been reported in utilizing such features to reduce power and energy consumption. They include devising a compiler algorithm for optimizing single-threaded programs for energy usage on laptops [18], power and energy management techniques for servers and data centers [5], to high-performance computing

workloads in SMP and AMP servers [8],[14] and in high-performance clusters [24],[35],[20],[16],[9]-[13],[15]-[17],[22].

Research on saving energy in clusters running *Message Passing Interface* (MPI) [25] applications is evolving. Such research is mainly focused at understanding and utilizing applications' behavior to scale down the CPU for energy savings. For instance, when the CPU is not in the critical path (such as blocking on an off-chip memory access, disk access, or waiting for a message to arrive over the network) it can be scaled down. While these works have made important contributions to the state of the art in power-aware HPC, unfortunately most of them have considered clusters based on notebooks and desktops interconnected by a Fast Ethernet network. Such systems are not high-performance nor balanced. They do not represent high-performance clusters with their server-class nodes and modern interconnects such as *Myrinet* [29], *Quadrics* [4] and *InfiniBand* [19].

Such modern interconnects are the backbone for scalable, high-performance clusters, offering an extremely low latency and high bandwidth communication. They use programmable *network interface cards* (NICs) that off-load protocol processing from the host processor, providing excellent opportunity for communication libraries such as MPI to overlap computation with communication. They also support *Remote Direct Memory Access* (RDMA) operations, allowing direct data transfer from the source buffer to the remote destination buffer without the host CPU intervention or intermediary copies. This inherent *zero-copy* feature of RDMA, in concert with the OS bypass mechanism, effectively helps save CPU cycles and memory bandwidth. Considering all these advanced features, it is therefore very important to understand the impact of contemporary networks, in comparison to traditional non-offloaded Ethernet networks, in delivering energy-efficiency in high-performance clusters.

In this work, we report on our experimentation with a 16-processor cluster consisting of eight two-way Intel Xeon SMP servers interconnected by Myrinet-2000, Quadrics QsNet^{II} and Gigabit Ethernet. The first contribution of this work is to present the power-performance profiles of such interconnects for point-to-point and collective communications at the user-level and MPI-level. We also study the impact of host CPU frequency scaling. The second contribution of this paper is to devise a power-aware MPI library that automatically and transparently performs message segmentation and re-assembly for point-to-point communications in order to boost the energy savings. Thirdly, by presenting the power-performance profiles of a number of all-gather collective communication algorithms proposed in [33], we argue that it is possible to increase energy savings by carefully selecting the right algorithm. To the best of our knowledge, this paper is the first study of its kind in power-aware high-performance computing.

Our experimental results indicate that the Quadrics QsNet^{II} enjoys a better energy savings than Myrinet-2000 at the user-level. At the MPI layer, Gigabit Ethernet has the worst performance and energy-efficiency amongst the three networks. However, it consumes the least power. Myrinet consumes the most power within the networks, and has a slightly worse energy-efficiency compared to QsNet^{II}. In terms of frequency scaling, Myrinet seems to be more promising than the others in improving the energy-efficiency. Gigabit Ethernet has no room for energy savings with freq. scaling. Overall, we conclude that modern networks with their advanced features can substantially help the energy-efficiency of a cluster. Our segmentation technique can improve the energy efficiency of the Gigabit network by up to 17% and of the Myrinet network by up to 21% for 32KB and 64KB messages. The proposed all-gather collective algorithms consume different energy for different message sizes. By selecting the most energy-efficient algorithm for each message range, one could achieve 13% average energy savings compared to the native *elan_gather()*.

The rest of this paper is organized as follows. In Section II, related work is discussed. Section III defines the metrics used in this work. We cover our experimental framework in Section IV. Section V and Section VI present and analyze the power-performance profiles of the networks under study at the user-level and MPI-level, respectively. In Section VII, we propose two different techniques to improve the energy-efficiency of the messaging layers in high-performance clusters. Finally, Section VIII concludes the paper.

II. Related Work

While there has been a large body of work in saving energy in the areas of mobile computing and embedded systems to extend battery life, energy conservation has recently become important in servers and clusters with their non-interactive high-performance computing workload in MPI and OpenMP [31]. In such systems, the focus is to improve reliability and to reduce the cost of powering and cooling.

Freeh and his colleagues [10] studied the energy-time trade-off in HPC applications. They found for memory-bound or communication-bound applications, a power-scalable cluster could save significant energy with only a minor time penalty. Along the same line, Feng et al. [9] presented a framework for direct, automatic profiling of HPC applications. They provided profiles by component, by node, and by system scale. They concluded in their applications, increasing the number of nodes always increases energy consumption but does not always improve performance. Hsu and Feng [17] confirmed a significant amount of energy could be saved in clusters while sustaining high performance. Our work in this paper is along the same objectives, however we consider the impact of modern networks in energy savings rather than characteristics of the applications.

Freeh et al. [12] extended the idea of using a single reduced frequency to run an application to using several frequency-voltage settings (gears) for different phases of the application. They then devised a heuristic to choose the assignment of frequency to each phase. They discovered that more than half their applications incur less energy using multiple gears than using a single gear. On the same token, Hotta and others [15] proposed an optimization algorithm to select a gear for each phase using the execution and power profile by taking the DVS transition time into account.

Kappiah and others [20] studied the opportunity that exists for energy savings with little performance loss when the computational load is not balanced in clusters. They monitor the time a program waits for an inter-node communication, and use DVFS to minimize the wait time and energy consumption. In [13], Ge and associates proposed distributed performance-directed DVS scheduling to scale down the CPU when peak CPU performance is not necessary. In a recent work, Freeh et al. [11] studied the combination of DVFS and CPU packing in scientific and commercial applications.

Springer et al. [35] used performance modeling, performance prediction, and program execution to choose the number of nodes and CPU frequency that simultaneously satisfy an upper limit for energy consumption and minimizes the execution time. Lim and others [24] proposed an MPI runtime system that dynamically reduces CPU performance during communication phases. This is done by amortizing the cost of DVS switching over several MPI calls or communication regions. They monitor and recognize the MPI calls, and use performance counters to choose the best power states. While this paper argues that there are opportunities for energy reduction in the communication, they look at it at the application layer. Moreover, they use a slow Ethernet network in their study, and do not consider energy savings through messaging layers in modern networks.

Hsu and Feng [16] proposed a transparent and self-adapting run-time system for power-awareness in clusters. Their algorithm is an interval-based algorithm that explicitly monitors the intensity level of off-chip accesses during each interval to make DVS scheduling decisions.

Kondo and associates [22] proposed a new cluster system design with adaptive power control. They kept the overall power consumption of the cluster below a threshold by adaptively managing the number of nodes and the clock frequency of the processors.

On energy savings for multi-threaded HPC applications in OpenMP, Grant and Afsahi [14] explored the power-performance efficiency of Hyper-Threaded asymmetric multiprocessor servers, and proposed a new scheduling algorithm to reduce the overall power consumption of a server while maintaining a high level of performance. Curtis-Maury and others [8] analyzed the energy-performance trade-off of varying thread granularity to reduce power and execution time.

Some work has addressed power efficiency in networks through simulation. Kim et al. [21] optimized the energy consumption of links and switches by using a dynamic link shutdown technique using adaptive routing. In [2], Alonso and others presented a mechanism to dynamically switch on and off network links as a function of traffic. Conner et al. [7] explored opportunities for link shutdown in a 3-D torus network during collective communication operations. None of the above works has experimentally evaluated modern networks in real clusters with their messaging layers.

III. METRICS

Metrics such as performance, power, energy and energy-delay are traditionally used in studying the power-performance characteristics of applications running on a system. However, to better understand the efficiency of communication subsystems in modern networks, we propose normalized metrics for performance, energy, and energy-delay. In the following, we define these metrics.

A. Delay and Delay per Byte

High-performance computing has always been concerned with performance. Performance (latency) of a point-to-point or a collective communication operation is given by wall-clock execution time, or simply *delay, D*. The normalized performance per byte metric is defined as the delay divided by the message size (*M*) in the communication call:

$$\hat{D} = \frac{D}{M} \qquad (1)$$

In essence, the *delay per byte* metric, $\hat{D}$, shows the performance efficiency of the communication subsystem. For a point-to-point communication, it is the time required to send a byte in an *M*-byte message transfer. It is in fact the inverse of the bandwidth. Similar comments can be made about a (regular) collective communication.

B. Power

Power, *P*, is the consumption at a discrete point in time. In this work, we focus on the node/cluster power consumption. However, the CPU is a major power consumer in a node/cluster. The power consumed by a CMOS processor (ignoring the power expended due to short-circuit current, and the power loss from leakage current), in watt, is equal to the activity factor of the system (percentage of gates that switch for each cycle) multiplied by the capacitance of the CPU times the voltage squared times the frequency [28]. This is shown in Equation 2.

$$P = \alpha C V^2 f \qquad (2)$$

Frequency scaling technologies can be effectively used to reduce the power consumption of the CPU during idle or slack times, where the CPU is busy waiting on slower components. This should not affect the performance drastically.

C. Energy and Energy per Byte

Energy is the cost during the execution time, *D*, and is shown as:

$$E = \int_0^D P dt = P_{avg} \times D \qquad (3)$$

The *energy per byte*, $\hat{E}$, is defined as the energy divided by the message size (*M*) in the communication call. If $\hat{E}_{M_1} \le \hat{E}_{M_2}$, then it is more energy-efficient to send a message with size M_1 than M_2. The energy per byte is formally defined as:

$$\hat{E} = \frac{P_{avg} \times D}{M} = P_{avg} \times \hat{D} \qquad (4)$$

D. Energy-delay and Energy-delay per Byte

The energy-delay metrics allow choosing the operating points at which maximum energy savings can be achieved with acceptable performance degradation. We use the energy-delay product to quantify the power-performance efficiency, as shown below:

$$E.D = E \times D = P_{avg} \times D^2 \quad (5)$$

$$\hat{E}.\hat{D} = \hat{E} \times \hat{D} = P_{avg} \times \hat{D}^2 \quad (6)$$

IV. Experimental framework

The experiments were conducted on a dedicated SMP cluster consisting of eight two-way Intel Xeon servers (Dell PowerEdge 2650). Each server has two 2.0GHz Intel Xeon MP processors (Prestonia), with 12KB shared execution trace cache, 8KB L1 shared data cache with 4-way associativity, and a 512KB shared and unified 8-way associative L2 cache. It has 1GB of DDR SDRAM on a 533MHz Front Side Bus, and two 64-bit/133MHz PCI-X slots.

The nodes run the Vanilla Linux kernel version 2.6.9. Using the LMbench [27], the L1 and L2 cache and main memory latencies are 1.11ns, 9.43ns, and 166.11ns, respectively. The main memory read and write bandwidths are 1488.71 MB/s and 820.75 MB/s, respectively.

A. Networks

We have experimented with three different networks: Myrinet-2000, Quadrics QsNet^{II} and Gigabit Ethernet. Note that in our experimentation, only one of the Myrinet-2000 or QsNet^{II} NICs was used at a time. In the following, we provide an overview of each interconnect.

1) Myrinet-2000: For the Myrinet tests, we experimented with a 16-port Myricom Myrinet-2000 [29] network, consisting of a 16-port switch with fiber ports (M3F-SW16) and eight Myrinet two-port "E card" (M3F2-PCIXE-2) network interface card (NIC). The Myrinet E-card has a 64-bit/133MHz PCI-X interface, and a programmable Lanai-2XP RISC processor operating at 333MHz with 2MB local memory. Each port has a 2.0+2.0 Gbps data rate. The standard firmware executing in the Lanai-2XP distributes packets adaptively across the two ports, such that the two ports act as a single 4.0+4.0 Gbps data-rate port. It also provides multi-path dispersive routing in the switch fabric that diffuses hot spots and increases the utilization of the network bisection for large network. The high-availability is instantaneous; by removing the fiber cable from either port, communication will continue using the remaining port.

We used MPICH-GM version 1.2.6 [29] on top of GM-2.1.21 user-level messaging layer for Myrinet. GM supports both *send/recv* and *Remote Direct Memory Access* (RDMA) operations. The send/recv mode is a two-sided operation, where both the sender and the receiver of a message are involved in the communication using *gm_send()* and *gm_recv()* primitives. However, RDMA is a one-sided operation, allowing direct transfer of data from a source process virtual address to a destination process virtual address without the host processor intervention or any intermediate copy. Operations finish without the remote side being involved. The RDMA operations include RDMA Write, *gm_put(),* and RDMA Read, *gm_get()*. In both send/recv and RDMA communication models, communication buffers must be registered prior to using them.

MPICH-GM uses the *Eager* protocol for sending small messages (less than 32K bytes) and the *Rendezvous* protocol for sending large messages. In the *Eager* protocol, the sender sends the entire message to the receiver, where the receiver needs to provide sufficient buffering space for the incoming message. The *Rendezvous* protocol is used for large messages, where the cost of copying is prohibitive. The sender and the receiver negotiate the availability of the buffer at the receiver side before the actual message transfer begins.

2) *Quadrics QsNet^{II}:* For the Quadrics network, nodes are connected to an 8-port QS8A-AA QsNet^{II} E-series switch through a QM500-B QsNet^{II} NIC. QsNet^{II} [4] is the latest generation interconnect from Quadrics. It consists of two ASICs: Elan4 and Elite4. The network is built out of an Elite4 switch component, where it can switch eight bi-directional channels. Elan4 is the communication processor on the NIC, and has a 64-bit/133MHz PCI-X interface. It provides a protected user-level access to the NIC so that user processes can share it simultaneously. QsNet^{II} differs from other contemporary interconnects, such as InfiniBand and Myrinet, in the sense that it integrates a node's local memory into a globally shared, virtual-memory space. There is no need for memory registration.

QsNet^{II} supports RDMA Write and Read operations via *elan_put()* and *elan_get()* functions, respectively. The MPI point-to-point implementation provided by Quadrics runs on top of a network programming interface called *Tagged Ports*, or *Tports*. Tports provides similar two-sided message-passing semantics as in MPI. While QsNet^{II} has hardware support for broadcast and barrier operations, a number of collectives such as reduce, gather, all-gather and all-to-all are implemented in its Elan user-level library. These collectives are directly used by their MPI counterparts. A number of RDMA primitives such as *elan_put()* and *elan_doput()* are used in the implementation of these collectives. *elan_doput()* is similar to *elan_put()* with the exception that it sets a destination event on completion. These functions are non-blocking, and *elan_wait()* may be used to wait on for the completion of the transfer.

Our Quadrics software is the recent "Hawk" distribution with kernel patch qsnetp2, kernel module 5.10.5qsnet, QsNet Library 1.5.9-1, and QsNet^{II} Library 2.2.11-2. Test codes were launched by the *pdsh* [32] task launching tool, version 2.6.1. The MPI implementation is the Quadrics MPI, version MPI.1.24-49.intel81. The MPI *Eager/rendezvous* protocol switching point in our system is at 100KB.

3) Gigabit Ethernet: For the Gigabit Ethernet, nodes are connected to a 16-port NETGEAR GS516T switch through the on-board Broadcom NetXtreme BCM5701 controllers. In

terms of software, we used MPICH 1.2.7.p1 [26] as the message-passing library for the Gigabit Ethernet network. The MPICH over Ethernet uses three different communication protocols: *short* (for messages less than 16KB), *Eager* (between 16KB and 125KB), and *Rendezvous* (for messages larger than 125KB). In the *short* protocol, the data is delivered within the message envelop itself. In essence, when the cost of extra data copy equals the latency to send a separate message, the protocol switches to the *Eager* mode.

B. Clock Throttling

To evaluate the energy-time trade-off of different networks, we created static configurations on our platform through clock throttling. Clock throttling is a clock gating technique that has a similar impact on performance as reducing the frequency [3]. In clock throttling, the duty cycle of the processor is reduced. This corresponds to a heat and energy savings proportional to the change in duty cycle. The available operating points on our platform are 2.0GHz, 1.75GHz, 1.5GHz, 1.25GHz, 1.0GHz, 750MHz, 500MHz and 250MHz. However, for legibility, we only present the results for 1.0GHz to 2.0GHz range.

The Linux 2.6.9 kernel supports clock throttling through a *sysfs* interface with appropriate drivers. We built the p4-clockmod driver into the kernel, and the standard sysfs interface was used to enable CPU frequency scaling using clock throttling. The frequency governor was set to user-space control, creating a static operating point in which the duty cycle is set only once before the application run.

C. Power Measurement

Our measurement infrastructure consists of a Keithley 2701/7710 digital multi-meter (DMM), a shunt resistor and a profiling PC, as shown in Fig. 1. We measure the power/energy consumption of a single node, as well as the entire cluster with its switch, by measuring the voltage of a (different) shunt resistor placed between the wall power outlet and the node/cluster. Knowing the value of the resistor, we first calculate the current and then the power and energy consumption of the node/cluster. We read 60 AC-voltage samples per second. In the DMM, the signal first goes through an internal analog RMS-converter, where 1000/60 DC samples is read out and averaged for each AC sample.

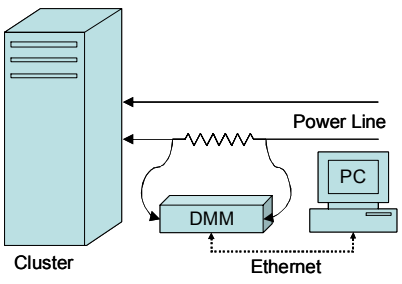

Fig. 1 Power measurement methodology

Using the Keithley communication library, we have developed software that runs on a profiling PC. The code is a sequence of commands that configures the DMM, starts the voltage sampling and then runs the micro-benchmark under study. Sampling is stopped when the micro-benchmark finishes, and then the voltage and performance readings are saved and post-processed on the profiling PC. We run the micro-benchmarks in a long loop so that the effect could be seen on the power line.

V. User-level Power-Performance Profile

Many MPI implementations and data center applications employ the advanced features of high-performance networks, such as RDMA and send/recv communications. For instance, Myrinet-2000 uses the user-level send/recv communication for short messages and RDMA for long messages. QsNet^{II} also uses Tports and RDMA in the implementation of its point-to-point and collectives. While previous research on high-performance networking have compared these communication models in terms of latency and bandwidth [6],[23],[34],[37], it is important to understand which model is more energy-efficient. This information is invaluable to the research community as well as information technology business, given such interconnects are largely being deployed in HPC and data centers.

Message latency is an important metric for many parallel and distributed computations. Latency is defined as the time it takes for a message to travel from the sender process address space to the receiver process address space. In the *unidirectional* micro-benchmark, the sender transmits a message repeatedly to the receiver, and then waits for the last message to be acknowledged. This is repeated sufficient number of times to eliminate the transient conditions of the network (note we have also experimented with the *bidirectional* and *both-way* traffics, however space limitation does not allow including their results. We will point out the significant differences in the text, if any).

Fig. 2 presents the user-level node power-performance characteristics of the Myrinet (with both ports active) and Quadrics networks under the unidirectional traffic (Ethernet does not support user-level networking). Note the Y-axis for delay per byte and energy per byte plots is in logarithmic scale. As shown in Fig.2, *elan_put()* outperforms all other user-level communications. The *Tports send/recv* and the *elan_get()* incurs 31% and 21% longer delays than the *elan_put()* on average across all message sizes, respectively. The GM user-level operations take 50-58% longer than the *elan_put()*.

Myrinet cluster consumes between 174W to 178W. While QsNet^{II} cluster consume the same amount of power for up to 2KB messages, power consumption jumps to 194W for larger messages. However, this increase in power consumption does not outpace Quadrics' better performance; that is why Quadrics RDMA Write shows a better energy-efficiency. In fact, on average and across all message sizes, Myrinet consumes 27% more energy.

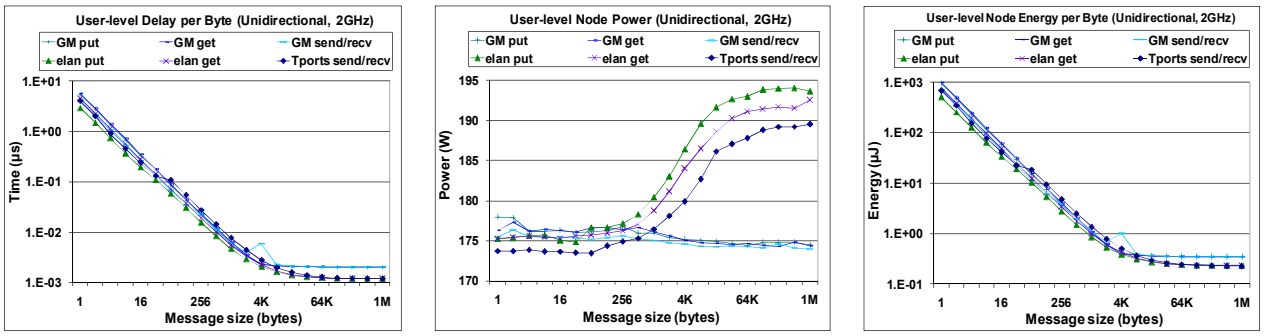

Fig. 2 User-level delay per byte ($\hat{D}$), node power, and node energy per byte ($\hat{E}$) of different networks

In both networks, RDMA Write shows a better energy-efficiency than the RDMA Read operation. While *gm_get()* is only 2% less energy-efficient than *gm_put(), elan_put()* is 20% more energy-efficient than *elan_get()*. When comparing RDMA versus send/recv operations, Quadrics RDMA shows a better energy-efficiency. However, Myrinet send/recv is slightly more energy-efficient than its RDMA. This is somehow justified as the Myrinet communication subsystem (including the new MX-10G) is mainly designed around send/recv operations that mimic MPI two-sided communications.

Contrary to Myrinet, Quadrics support a number of collectives directly at the elan level. In this regard, we will compare the native all-gather collective in *elan_gather()* with our collectives in Section VII.B. It should be mentioned that the energy plots follow the same trends in performance curves in almost all of our results. Therefore, from now on, and due to space limitations, we will not show the performance results.

VI. MPI Power-Performance Profile

Our main objective is twofold. First, we want to compare modern networks in terms of power/energy consumption. We would also like to know how host CPU frequency scaling might affect the power-performance profiles of the different networks. Secondly, having this knowledge, we would like to discover how messaging layers could be optimized for energy-efficiency. For this, we present two different techniques in Section VII.

A. Point-to-point Communication

Fig. 3 presents the power-performance profile of our networks at the MPI layer under the unidirectional communication traffic for *MPI_Send()*. Similar results have been observed for *MPI_Recv()* and their non-blocking counterparts. The Gigabit Ethernet, expectedly, has the worst performance (largest delay) amongst the three networks (not shown). As the message size increases above 4KB, the performance gap between the Gigabit Ethernet and those of QsNet^{II} and Myrinet also increases. Quadrics and Myrinet enjoy up to 7.5 and 13 times smaller delay than the Gigabit, respectively. The performance gain of QsNet^{II} over the Myrinet is between -40% to +130% across all message sizes, with an average gain of 22%.

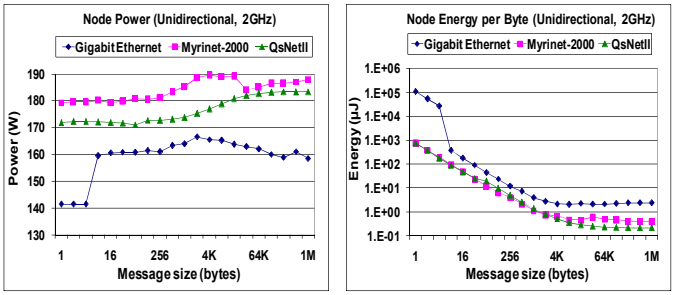

Fig. 3 MPI node power and node energy per byte of different networks

One can easily observe from the Gigabit energy curve that the system performs strangely for up to 4-byte messages. We repeated our tests a number of times with different number of loop iterations, however we observed the same trend. It should be noted this behavior is not observed in the bidirectional and both-way traffics. We are currently investigating the reason behind this, but we are speculating it could be due to the MPICH implementation (MPICH developers have since shifted their attention to MPICH2 over Ethernet). We chose MPI-1 for Ethernet for compatibility reasons with the other two MPI implementations in this paper.

It is evident the Gigabit Ethernet network has the lowest power consumption among the three networks. Our communication micro-benchmark is intensively communication bound. QsNet^{II} and Myrinet-2000 are high-performance networks, where they both off-load protocol processing from the host CPU. Therefore, messages are sent and received over the network in a much faster pace than for the Gigabit. This makes the host CPU work harder per instant of time, essentially increasing the instantaneous node power consumption. The reverse is true for Gigabit, where the network is slow. Evidently, the Gigabit network shows the poorest energy-efficiency among all networks, while QsNet^{II} has a slightly better energy-efficiency than Myrinet.

1) Frequency Scaling: We turn our attention to understand the impact of clock gating on power and energy consumption of different networks under unidirectional traffic. Fig. 4 presents node power and energy consumption values at different frequencies and messages normalized to those at 2.0GHz. Reducing the frequency of the system from 2.0GHz to 1.75GHz, 1.50GHz, 1.25GHz, and 1.0GHz increases the

delay of the Gigabit network by up to 15%, 34%, 61%, and 98%, respectively (not shown in the Figure). As discussed in Section VI.A, Gigabit network does not offload protocol processing from the host CPU. Slowing down the host CPU will slow down the execution of the benchmark, as well as TCP protocol stack computation. As Myrinet and QsNet^{II} offload their communication protocol processing from the host CPU, we expect to see less increase in the delay when the host CPU is scaled down.

Interestingly, Myrinet shows almost no increase in the delay for small message sizes up to 1KB, when the host CPU is scaled down even to 1.0GHz. Surprisingly, delay is improved by up to 13% for 2KB to 16KB messages. For messages larger than 16KB, the maximum jump in the delay is not more than 10%. This behavior shows the excellent job done by the Myrinet offload engine and its messaging layer. The situation for QsNet^{II} is not as good as for Myrinet. For messages larger than 8KB, QsNet^{II} incurs up to 11% increase in delay. However, for shorter messages, the delay is increased by up to 13%, 29%, 49%, and 71% for the 1.75GHz, 1.50GHz, 1.25GHz, and 1.0GHz frequencies, respectively.

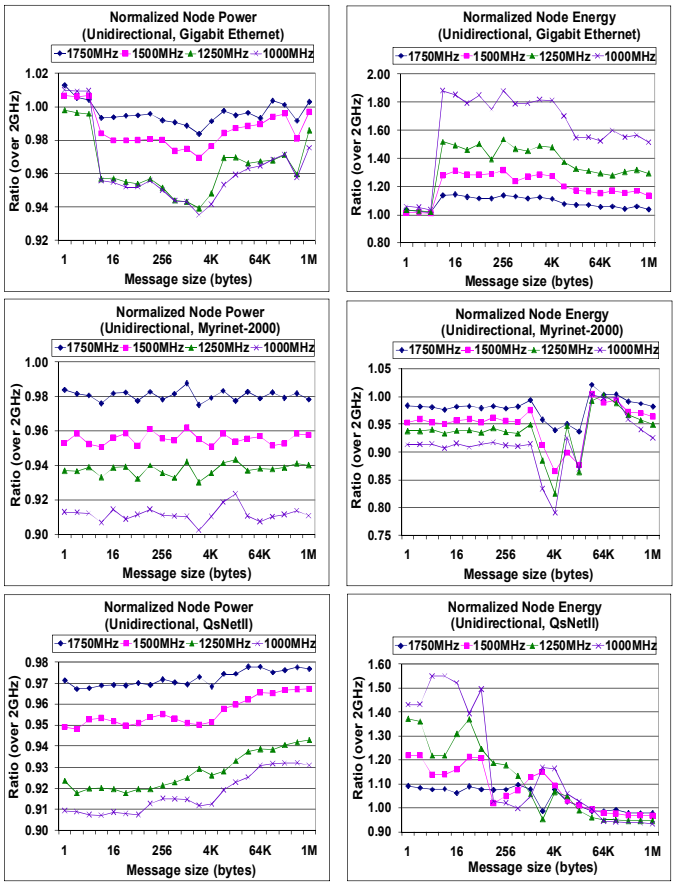

Fig. 4 Effect of frequency scaling on MPI node power and node energy per byte of different networks

Slowing the host CPU reduces the instantaneous power consumption of the system for all the three networks by a small margin. Fig. 4 shows a relatively flat power curves. In the energy per byte plots, we can see that Gigabit Ethernet does not show any energy savings for any message size. In fact, energy consumption increases. This is due to the significant increase in the delay. For Myrinet, one could observe that up to 21% energy can be saved by scaling the CPU down to 1.0GHz. QsNet^{II} can only save energy by up to 7% for messages larger than 16KB. For smaller messages, energy loss due to CPU scaling could be up to 55%.

B. Collective Communication

Fig. 5 presents the detailed results for only two collectives: broadcast and all-to-all. As usual, Gigabit shows the poorest performance (not shown). QsNet^{II} incurs the lowest delay for almost all collectives including broadcast and all-to-all, except for a few message sizes for the all-reduce collective. This is because Quadrics implements some of its collectives directly at the user-level. Specifically, broadcast and barrier are implemented directly at the hardware level.

As in the point-to-point communication, Myrinet is the most power hungry network, while Gigabit Ethernet consumes the least instantaneous power. The Gigabit network shows the worst energy-efficiency, while QsNet^{II} enjoys the best energy-efficiency. Myrinet-2000 is less energy-efficient than QsNet^{II} for all collectives except for all-reduce collective with messages larger than 4KB.

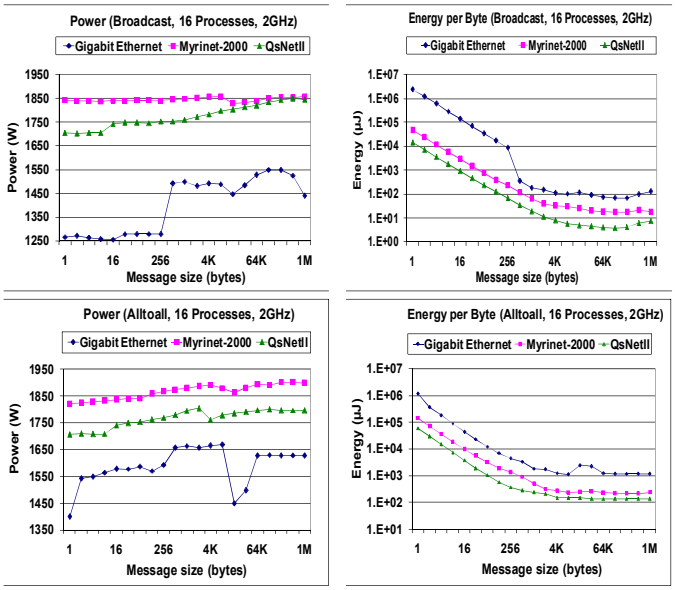

Fig. 5 Cluster power and energy per byte for MPI broadcast and all-to-all

The average power and energy consumption for the Myrinet across all message sizes for different collectives (broadcast, scatter, gather, reduce, all-to-all, all-gather, and all-reduce) is 126% and 17% of the Gigabit Ethernet, respectively. For QsNet^{II}, these numbers are 119% and 16%, respectively. For the broadcast, gather, all-to-all, and all-gather collectives, QsNet^{II} cluster on average uses 43% of the energy used on the Myrinet cluster. However, for the scatter, all-reduce, and reduce collectives, it uses up to 148% of the energy that the Myrinet system consumes.

1) Frequency Scaling: In this section, we study the effect of frequency scaling on broadcast and all-to-all. An energy-

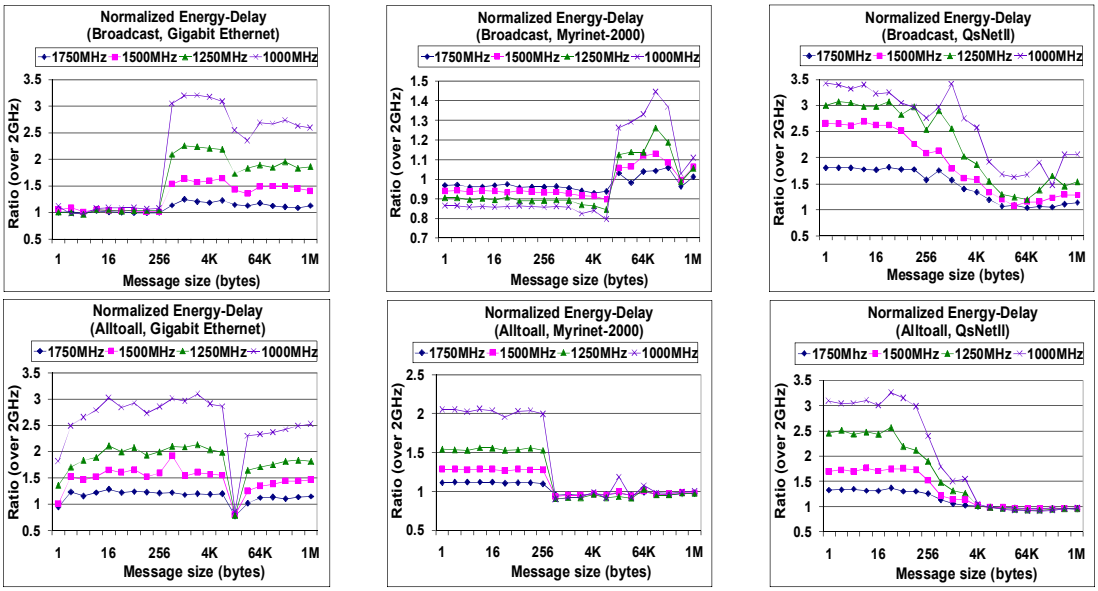

Fig. 6 Normalized energy-delay (over 2.0GHz) for broadcast and all-to-all collectives of different networks

delay of less than one shows that the savings in energy consumption outpace the corresponding increase in execution speed. Values greater than one represents the case in which the energy savings are not sufficiently offset by the increase in execution time. A downward trend in the graphs indicates that the rate of decrease of energy consumption and the rate of delay related to said energy savings is diverging beneficially.

Fig. 6 illustrates the energy-delay of the collectives normalized to those running at 2.0GHz. Except for the broadcast and for very short messages (up to 256 bytes), the Gigabit Ethernet has no room for energy savings, as reducing the frequency increases the latency, thereby increasing the energy consumption.

The same story applies to Myrinet and QsNet^{II} when an all-to-all collective is run over the cluster. It is observed that under different frequencies, the delay does not change much, effectively neglecting minor changes in power consumption. Therefore, consuming around 90% of the energy at 2.0GHz, and achieving an energy-delay of almost one guarantee 10% energy savings with almost no time penalty.

Myrinet is more promising in improving the energy-efficiency. For the Myrinet, reducing the host CPU frequency can improve the energy-efficiency of the broadcast and all-to-all operations by up to 16% and 11%, respectively. QsNet^{II} shows no improvement for energy-efficiency of the broadcast operation. However, it improves the energy-efficiency of the all-to-all collective by up to 9% for large messages.

The power efficiency of the system improves for all the benchmarks on the Myrinet and the QsNet^{II} networks, regardless of the message size. For short messages and for some collectives such as reduce, scatter, and broadcast, the power consumption is not decreased on the Gigabit network. However, for other collectives, or larger message sizes, three of the four lower frequencies reduce the power consumption..

For short message sizes, frequency scaling improves the energy efficiency of the Myrinet for all the collectives except all-to-all and reduce. For medium size messages, Myrinet benefits from frequency scaling only in broadcast and all-to-all. For large messages, all-gather and all-to-all collectives reduce energy consumption. For QsNet^{II}, all-to-all, gather, and scatter under large messages improve energy efficiency.

C. *Multi-port vs. Single-port Communication*

Fig. 7 compares the energy consumption of different collectives (running with 16 processes) on our cluster with one-port and two-port Myrinet network configurations. Results clearly show that message striping or channel multiplexing starts at around 1KB. For messages larger than 1KB, energy-efficiency declines up to 80% due to the lower bandwidth of the one-port communication. The collective power consumption with one-port communication changes within 1-2% of the two-port configuration.

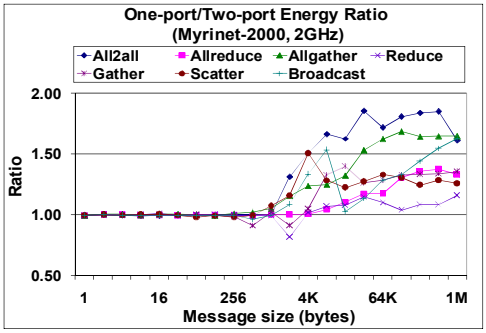

Fig. 7 Myrinet-2000 one-port/two-port energy comparison

D. *Interconnection Network Switch*

Our experimentation with the Gigabit and the Myrinet switches revealed that both switches do not have any power management technique and continuously consume 57W and 69W, respectively, regardless of the network traffic. However, the QsNet^{II} switch consumes 36W when idle, and 42W under

traffic. Figure 8 depicts the average power consumption of a 16-port Gigabit Ethernet switch, a 16-port Myrinet switch, and an 8-port QsNet^{II} switch, when they are under different communication-intensive micro-benchmarks. The *Idle-N* mode is when the switch is not under any load, and only *N* physical links are connected to the switch. The power consumption for the *Idle-0* configuration is 44W, 67W, and 34W for the Gigabit, Myrinet, and the QsNet^{II}, respectively. Results show that designing a switch with multiple power levels is vital for power-aware interconnection networks.

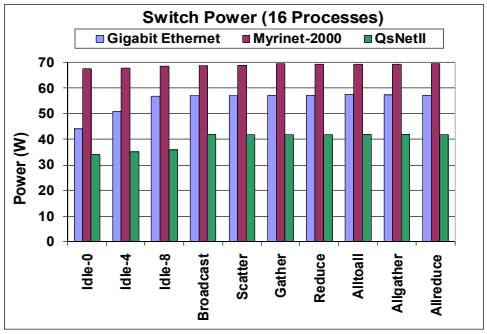

Fig. 8 Power consumption of different switches under different loads

VII. Software Power/Performance Enhancement

After analyzing the power-performance profiles of contemporary networks, we would like to elaborate on how one might be able to improve the energy-efficiency of a cluster by optimizing its messaging layer. We propose two approaches: 1) a message segmentation technique, and 2) an algorithmic approach.

A. A Message Segmentation Technique

Fig. 9 illustrates the normalized power and energy per byte of different networks in our cluster, when a point-to-point communication is under way. It is evident that increasing the message size does not affect the power consumption much. However, some interesting results can be observed from the energy curves. The decreasing energy trend of QsNet^{II} prompts that larger messages have a better energy-efficiency. The Gigabit energy curve does not show a decreasing trend, however. This means that larger messages are less energy-efficient than shorter messages. It implies message segmentation may produce higher energy-efficiency for Gigabit Ethernet clusters. Myrinet-2000 shows a decreasing trend in its energy curve, except for the 32KB message size. This is the *Eager/Rendezvous* switching point. Therefore, message segmentation may also be helpful at 32KB messages for Myrinet clusters. The decreasing trend of the QsNet^{II} curve does not suggest any energy improvement using the segmentation method.

We have developed an MPI run-time library, using the wrapper facility of MPI, which transparently segments large messages into smaller ones. It also re-assembles the messages at the other end. Fig. 10 shows the results with different chunk sizes. One can see that it is possible to improve the energy efficiency by up to 17% on the Gigabit network. For the Myrinet case, the improvement is up to 21% for 32KB and 64KB messages. As expected, there is no energy improvement for the QsNet^{II} network.

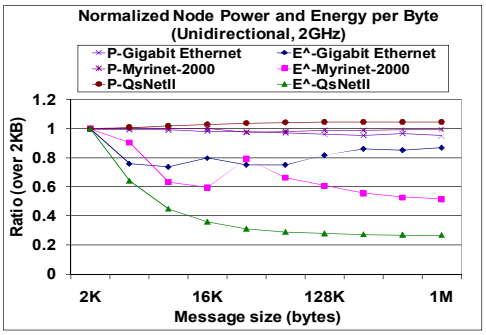

Fig. 9 Normalized node power and energy per byte for different networks

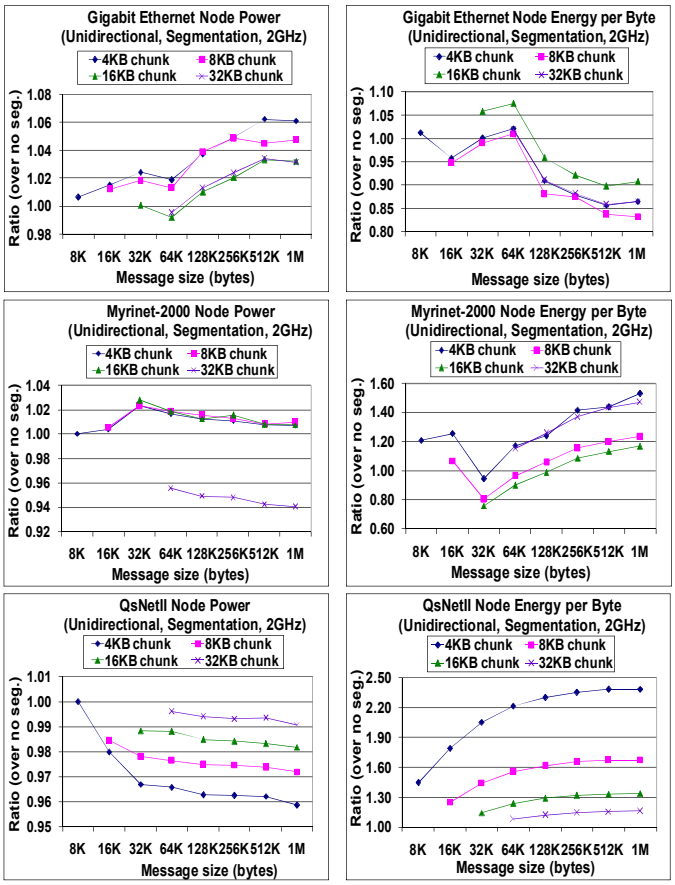

Fig. 10 Impact of the message segmentation on power and energy efficiency

B. An Algorithmic Approach

Our intention in this section is to show that messaging layers, whether at the MPI-level or user-level, play a key role in the energy-efficiency of a high-performance cluster. Specifically, we will present a number of collective communication algorithms [33] that are designed and implemented directly at the elan level over QsNet^{II}.

With the emergence of multi-core Symmetric Multiprocessors (SMP) nodes in clusters, there is now immense pressure on interconnection networks and their communication system software to support multiple intra-

node and inter-node communications among different cores efficiently. Utilizing shared memory communication among co-located processes on SMP nodes as well as RDMA operations for the inter-node communication and trying to overlap them will boost the performance of collective operations. The effect is much more pronounced when efficient multi-port collective algorithms on high-performance networks are devised and implemented. In this paper, we have ported the multi-port, multi-rail all-gather algorithms proposed in [33] to our single-rail cluster. Our algorithms are still multi-port algorithms; however, in porting them, messages that have to go over multiple rails will be now pipelined over the available single rail.

We propose and evaluate two classes of multi-port all-gather algorithms over QsNet^{II}, directly at the Elan level, based on the message size: an RDMA-only algorithm for medium to large messages, to be called *Direct* algorithm, and two combined RDMA-based and SMP-aware algorithms for small and medium size messages, to be called *SMP-aware Gather and Broadcast*, and *SMP-aware Direct*. In the SMP-aware algorithms, we distinguish between the intra-node and inter-node communications. However, we do not just simply replace the intra-node communications in the traditional algorithms with shared memory communications. In the following, we briefly discuss our algorithms.

1) Direct Algorithm. The Direct all-gather algorithm is the extension of the sequential tree algorithm for *k*-port modeling, and suitable for medium to large messages. In each step, each process sends its own message to *k* other processes in a wrap-around fashion. There are a total of $\lceil N-1/k \rceil$ communication steps for *N* processes involved in the operation.

2) *SMP-aware Gather and Broadcast Algorithm:* The algorithm essentially does an SMP-aware gather algorithm across all processes in the system and then broadcasts the gathered data to all processes, as shown below:

Phase 1: SMP-aware gather across all processes
  Step A: Per-node shared memory gather
  Step B: Inter-node gather among the Master processes
Phase 2: Broadcasting gathered data to all processes

3) SMP-aware Direct Algorithm: This algorithms is done in three phases:

Phase 1: Per-node shared memory gather
Phase 2: Inter-node all-gather among the Master processes
Phase 3: Per-node shared memory broadcast

4) Evaluation and Analysis: We have implemented the proposed algorithms directly at the elan-level, and compared them with the native all-gather algorithm in the *elan_gather()*. Fig. 11 presents the power and energy consumption of the different algorithms when they are run over our QsNet^{II} SMP cluster. The *Direct* algorithm has the lowest power consumption among our algorithms for messages smaller than 4KB. However, it also incurs a larger delay, and hence larger energy. The *elan_gather()* and the *SMP-aware Gather and Broadcast* algorithms show the lowest power consumption for larger messages.

For messages smaller than 16B and larger than 16KB, the *elan gather()* consumes the least amount of energy. For 16-512B and 16KB messages, the *SMP-aware gather and broadcast* enjoys the most energy efficiency. The *SMP-aware direct* algorithm has a better energy efficiency than the other algorithms for 1-8KB messages. The *Direct* algorithm has almost the worst energy efficiency among our algorithms. By selecting the most energy efficient algorithm for each message size, one could achieve 13% average energy savings compared to the native *elan gather()*. This is a significant achievement and justifies further work in this area.

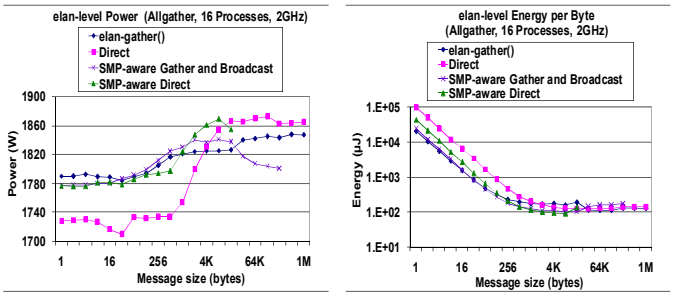

Fig. 11 Power and energy for different all-gather algorithms over QsNet^{II}

VIII. Conclusions and Future Work

Power consumption has become an important design constraint in high-performance clusters. Rather than focusing on the applications themselves, in this paper we have investigated the impact that a communication subsystem may have on the energy-efficiency of the entire cluster. For this, we have considered two modern networks and a traditional non-offloaded network in our study.

At the user-level, Quadrics QsNet^{II} enjoys a better energy savings than Myrinet-2000. While RDMA is more energy-efficient in Quadrics, efficiency of the send/recv communication in Myrinet is slightly better than its RDMA counterpart. At the MPI layer, Gigabit Ethernet has the worst performance and energy-efficiency amongst the three networks. However, it consumes the least power, so suitable for centers that have a cap on power consumption. Myrinet consumes the most power within the networks, and has a slightly worse energy-efficiency compared to QsNet^{II}. Interestingly, Myrinet does not show an energy increase for small messages, while Quadrics has a better energy efficiency for larger messages. For some collectives, Quadrics is more energy-efficient, while for others Myrinet is a greener consumer. In terms of frequency scaling, it seems Myrinet is more promising than the others to improve the energy-efficiency. Gigabit Ethernet has no room for energy savings with frequency scaling. Overall, we can conclude that off-loading protocol processing to the NIC can substantially help the energy-efficiency of a cluster.

This paper argues that messaging layers can be optimized for energy efficiency. We proposed two techniques in this regard: 1) a runtime library for message fragmentation/de-fragmentation, and 2) a number of energy-efficient all-gather

algorithms. For the future work, we intend to study the emerging interconnects such as iWARP [34] along with InfiniBand in a larger testbed. We also want to optimize other aspects of messaging layers for energy efficiency.

ACKNOWLEDGMENT

This research is supported by the Natural Sciences and Engineering Research Council of Canada through grants RGPIN/238964-2005 and RGPIN/116925-2002, Canada Foundation for Innovation's grant #7154, and Ontario Innovation Trust's grant #7154.

REFERENCES

[1] G. Almasi et al., "Unlocking the performance of BlueGene/L supercomputer," in *Proc. ACM/IEEE Supercomputing'04 (SC'04)*, 2004.

[2] 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," *2nd Workshop on High-Performance, Power-Aware Computing (HP-PAC 2006)*, in *Proc. 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006.

[3] M. Annavaram, E. Grochowski, and J. Shen, "Mitigating Amdahl's law through EPI throttling," in *Proc. 32nd Annual International Symposium on Computer Architecture (ISCA'05),* pp. 298-309, June 2005.

[4] J. Beecroft, D. Addison, D. Hewson, M. McLaren, D. Roweth, F. Petrini, and J. Nieplocha, "QsNetII: Defining high-performance network design," *IEEE Micro*, 25(4):34-47, July-Aug. 2005.

[5] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *Computer,* vol. 37, pp. 68-74, 2004.

[6] R. Brightwell, D. Doerfler, and K.D. Underwood, "A comparison of 4X InfiniBand and quadrics elan-4 technologies," in *Proc. 2004 IEEE International Conference on Cluster Computing (Cluster 2004),* pp. 193-204, 2004.

[7] S. Conner, S. Akioka, M. J. Irwin, and P. Raghavan, "Link shutdown opportunities during collective communications in 3-D torus nets," *3rd Workshop on High-Performance, Power-Aware Computing (HP-PAC 2007)*, in *Proc. 21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007.

[8] M. Curtis-Maury, J. Dzierwa, C. D. Antonopoulos, and D. S. Nikolopoulos, "Online strategies for high-performance power-aware thread execution on emerging multiprocessors," *2nd Workshop on High-Performance, Power-Aware Computing (HP-PAC 2006)*, in *Proc. 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006.

[9] X. Feng, R. Ge, and K. W. Cameron, "Power and energy profiling of scientific applications on distributed systems," in *Proc. 19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, 2005.

[10] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," *IEEE Transactions on Parallel and Distributed Systems,* vol. 18, no. 6, pp. 835-848, June 2007.

[11] V. W. Freeh, T. K. Bletsch, and F. L. Rawson, "Scaling and packing on a chip multiprocessor," *3rd Workshop on High-Performance, Power-Aware Computing (HP-PAC 2007)*, in *Proc. 21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007.

[12] V. W. Freeh, F. Pan, N. Kappiah, and D. K. Lowenthal, "Using multiple energy gears in MPI programs on a power-scalable cluster," in *Proc. 10th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP'05),* pp. 164-173, 2005.

[13] R. Ge, X. Feng, and K. W. Cameron, "Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters," in *Proc ACM/IEEE Supercomputing'05 (SC'05),* 2005.

[14] R. E. Grant and A. Afsahi, "Power-performance efficiency of asymmetric multiprocessors for multi-threaded scientific applications," *2nd Workshop on High-Performance, Power-Aware Computing (HP-PAC 2006)*, in *Proc. 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006.

[15] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi, "Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster," in *Proc. 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS'06),* 2006.

[16] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high-performance computing," in *Proc. ACM/IEEE Supercomputing'05 (SC'05),* 2005.

[17] C.-H. Hsu and W.-C. Feng, "A feasibility analysis of power awareness in commodity-based high-performance clusters," in *Proc. 2005 IEEE International Conference on Cluster Computing (Cluster 2005)*, 2005.

[18] C.-H. Hsu and U. Kremer, "The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction," in *Proc. of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03),* pp. 38-48, 2003.

[19] InfiniBand Architecture. [Online]. http://www.infinibandta.org/

[20] N. Kappiah, V. W. Freeh, and D. K. Lowenthal, "Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs," in *Proc. ACM/IEEE Supercomputing'05 (SC'05)*, 2005.

[21] E. J. Kim, G. M. Link, K. H. Yum, N. Vijaykrishnan, M. Kandemir, M. J. Irwin and C. R. Das, "A holistic approach to designing energy-efficient cluster interconnects," *IEEE Transactions on Computers*, vol. 54, no. 6, June 2005.

[22] M. Kondo, Y. Ikeda, and H. Nakamura, "A high performance cluster system design by adaptive power control," *3rd Workshop on High-Performance, Power-Aware Computing (HP-PAC 2007)*, in *Proc. 21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007.

[23] J. Liu, B. Chandrasekaran, W. Yu, J. Wu, D. Buntinas, S. Kini, D.K, "Panda and P. Wyckoff, "Microbenchmark performance comparison of high-speed cluster interconnects," *IEEE Micro,* 24(1):42-51, 2004.

[24] M. Y. Lim, V. W. Freeh and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs," in *Proc. ACM/IEEE Supercomputing'06 (SC'06)*, 2006.

[25] MPI, A Message Passing Interface standard, Version 1.2, 1997.

[26] MPICH. [Online]. Available: http://www-unix.mcs.anl.gov/mpi/

[27] L.W. McVoy and C. Staelin, "Lmbench: portable tools for performance analysis," in *Proc. 1996 USENIX Annual Technical Conference,* pages 279-294, 1996.

[28] T. Mudge, "Power: a first class design constraint," *IEEE Computer*, 34(4):52-57, Apr. 2001.

[29] Myricom. [Online]. Available: http://www.myricom.com/

[30] H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka, D. Takahashi, and Y. Hotta, "MegaProto: 1 TFlops/10KW rack is Feasible even with only commodity technology," In *Proc. ACM/IEEE Supercompuing'05 (SC'05)*, Nov. 2005.

[31] OpenMP Architecture Review Board, OpenMP Specification Version 2.5, May 2005.

[32] PDSH. [Online]. available: http://www.llnl.gov/linux/pdsh/

[33] Y. Qian and A. Afsahi, "RDMA-based and SMP-aware Multi-port All-Gather on Multi-rail QsNet^{II} SMP Clusters," To appear in *Proc. 36th International Conference on Parallel Processing (ICPP'07)*, Sept. 2007.

[34] M. J. Rashti and A. Afsahi, "10-Gigabit iWARP Ethernet: Comparative Performance Analysis with InfiniBand and Myrinet-10G," *7th Workshop on Communication Architecture for Clusters (CAC 2007)*, In *Proc. of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007.

[35] R. Springer, D. K. Lowenthal, B. Rountree and V. W. Freeh, "Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster," in *Proc. 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'06),* pp. 230-238, 2006.

[36] M. Warren, E. Weigle, and W. Feng, "`High-density computing: a 240-node Beowulf in one cubic meter," In *Proc. ACM/IEEE Supercoputing'02 (SC'02)*, Nov. 2002.

[37] R. Zamani, Y. Qian, and A. Afsahi, "An evaluation of the Myrinet/GM2 two-port networks," in *2004 IEEE Workshop on High Speed Local Area Networks (HSLN '04)*, pp. 734-742, 2004.