Per-call Energy Saving Strategies in All-to-all Communications[*]

Vaibhav Sundriyal and Masha Sosonkina

Department of Electrical and Computer Engineering
Ames Laboratory
Iowa State University
Ames, IA 50011 USA
`vaibhavs,masha@scl.ameslab.gov`

Abstract. With the increase in the peak performance of modern computing platforms, their energy consumption grows as well, which may lead to overwhelming operating costs and failure rates. Techniques, such as Dynamic Voltage and Frequency Scaling (called DVFS) and CPU Clock Modulation (called throttling) are often used to reduce the power consumption of the compute nodes. However, these techniques should be used judiciously during the application execution to avoid significant performance losses. In this work, two implementations of the all-to-all collective operations are studied as to their augmentation with energy saving strategies on the *per-call* basis. Experiments were performed on the OSU MPI benchmark as well as NAS and CPMD application benchmarks, in which power consumption was reduced by up to 10% and 15.7%, respectively, with little performance degradation.

Keywords: Collective Communications, MPI, DVFS, CPU Throttling.

1 Introduction

Power consumption is rapidly becoming one of the critical design constraints in modern high-end computing systems. While the focus of the high-performance computing (HPC) community has been to maximize the performance, the system operating costs and failure rates can reach a prohibitive level.

The Message Passing Interface[1] has become a *de facto* standard for the design of parallel applications. It defines both point-to-point and collective communication primitives widely used in parallel applications. This work examines the nature of all-to-all communications because they are among the most intensive

[*]This work was supported in part by Iowa State University under the contract DE-AC02-07CH11358 with the U.S. Department of Energy, by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC02-05CH11231, and by the National Science Foundation grants NSF/OCI – 0749156, 0941434, 1047772.

[1]MPI Forum: http://www.mpi-forum.org

and time consuming collective operations while being wide-spread in parallel applications. By definition, a collective operation requires the participation of all the processes in a given communicator. Hence, such operations incur a significant amount of the network phase during which there exist excellent opportunities for applying energy saving techniques, such as DVFS and CPU throttling.

The all-to-all operation is studied here on the *per-call* (fine-grain) basis as opposed to a "black-box" approach, which treats communication phase as indivisible operation contributing to the parallel overhead. In this work, the energy saving strategies are incorporated within the existing all-to-all algorithms.

CPU Throttling and DVFS in Intel Architectures. The current generation of Intel processors provides various P-states for DVFS and T-states for throttling. In particular, the Intel "Core" microarchitecture, which provides four P-states and eight T-states from T_0 to T_7, where state T_j refers to introducing j idle cycles per eight cycles in CPU execution. The delay of switching from one P-state to another can depend on the current and desired P-state and is discussed in [2]. The user may write a specific value to Model Specific Registers (MSR) to change the P- and T-states of the system.

Infiniband has become one of most popular interconnect standard marking its presence in more that 43% of the systems in the TOP 500[2] list. Several network protocols are offloaded to the Host Channel Adapters (HCA) in an Infiniband[3] network. Here, MVAPICH[4] implementation of MPI, which is designed for Infiniband networks, is considered. MVAPICH2 uses "polling" communication mode by default since a lower communication overhead is incurred with polling when an MPI process constantly samples for the arrival of a new message rather than the with "blocking", which causes CPU to wait for an incoming message.

1.1 Effect of CPU Throttling on Communication

Since point-to-point communication operations underlie collectives, it is reasonable to analyze the CPU throttling effects on them first. Fig. 1(a) shows the point-to-point internode communication times for the communicating processes at T-states T_0 and T_5. Similarly, Fig. 1(b) depicts the change in intranode communication time for the states T_0 and T_1. It can be observed that the effect of throttling on internode communication is minimal. In fact, the average performance loss was just 5% at state T_5 for various message sizes. However, introducing just one idle cycle per eight cycles degrades the intranode communication considerably (about 25%). This is expected since intranode communication uses more CPU cycles for a message transfer whereas in internode transfers RDMA offloads a large part of the communication processing to the NICs [13].

The difference between the intra- and inter-node message transfer types with respect to CPU throttling becomes the basis for the energy saving strategy

[2]http://www.top500.org/

[3]http://www.infinibandta.org/

[4]http://mvapich.cse.ohio-state.edu/

proposed in this work. The appropriate T-state is selected depending on the communication type of *all* the cores in a socket. The Intel Xeon processor, which is used in this work, supports DVFS and throttling only on the socket level rather than on the core level of granularity. Hence, the lowest T-State T_0 is chosen for all the cores when *at least one* core on a socket is in the intranode communication. Conversely, a higher throttling state T_5 is selected when *all* the cores on a socket perform internode communication. In the experiments, a throttling higher than T_5 resulted in a significant performance loss, which is not desirable since the aim is to minimize the energy consumption without sacrificing the performance. However, if all the socket cores are idle during the collective operation, then they can be throttled at the highest state T_7. To summarize,
∘ All cores communicate internode $\rightarrow \quad T_5$;
∘ At least one core communicates intranode $\rightarrow \quad T_0$;
∘ None of the cores communicate $\rightarrow \quad T_7$.

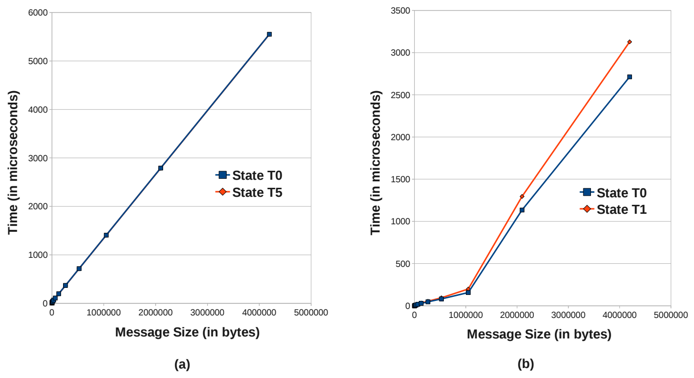

Fig. 1. Point-to-point communication with throttling (a) internode (b) intranode.

The rest of the paper is organized as follows. Section 2 describes the proposed energy savings in the all-to-all operation. Section 3 shows experimental results while Sections 4 and 5 provide related work and conclusions, respectively.

2 All-to-all Energy Aware Algorithm

MVAPICH2 implementations of all-to-all are considered in this work. They are based on three algorithms: 1) Bruck Index [14], used for small — less than 8KB — messages with at least eight participating processes; 2) Pairwise Exchange, used for large messages and when the number of processes is a power of two; 3) Send To rank $i + k$ and Receive From rank $i - k$, used for all the other processor numbers and large messages. These algorithms are referred further in text as BIA, PEA, and STRF, respectively.

Bruck Index first does a local copy with the upward shift of the data blocks from the input to output buffer. Specifically, a process with the rank i rotates its data up by i blocks. The communication starts such that, for all the p communicating processes in each communication step k $(0 \leq k < \lceil \log_2 p \rceil)$, process i, $(i = 0, \ldots, p-1)$, sends to $(i + 2^k) \bmod p$ (with wrap-around) all those data blocks whose kth bit is 1 and receives from $(i - 2^k) \bmod p$. The incoming data is stored into the blocks whose kth bit is 1. Finally, the local data blocks are shifted downward to place them in the right order. Fig. 2 shows a cluster having $N = 3$ nodes with $c = 8$ cores each placed on two sockets and the total number of processes $p = 8N$. The rank placement is performed in block manner using consecutive core ordering. Note that, until the kth step where $2^k < c$, the communication is still *intranode* for any socket in the cluster, considering all the cores on a socket collectively. However, after the kth step, the communication becomes purely *internode* for all the participating cores. Thus, from this step on, the throttling level T_5 may be applied to all the cores without incurring a significant performance loss.

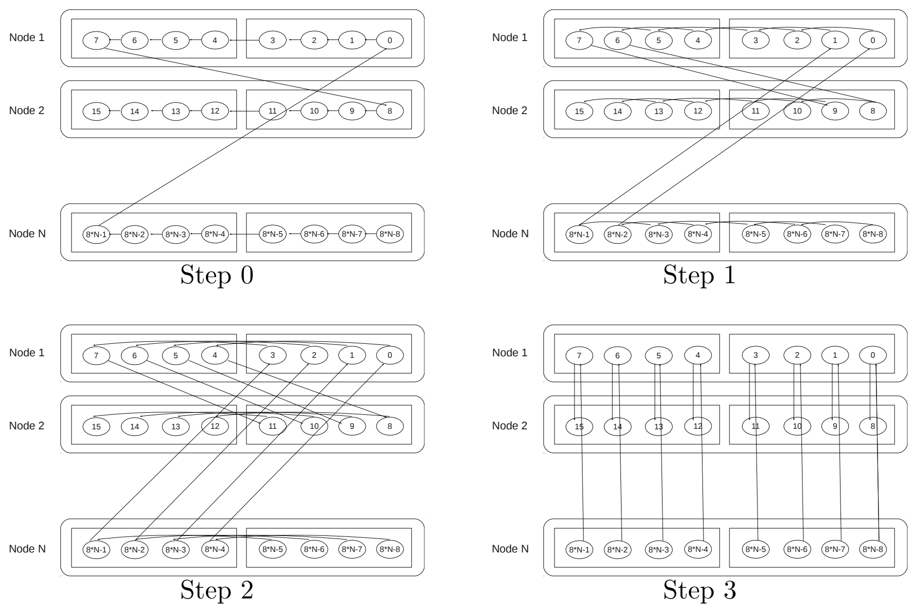

Fig. 2. First four communication steps in the Bruck Index all-to-all algorithm on three nodes with two sockets (shown as rectangles) and eight cores (ovals) each. Internode communications are shown as straight slanted lines across the node boundaries.

"Send-To Receive-From" and Pairwise Exchange. For the block placement of ranks, in each step k $(1 \leq k < p)$ of STRF, a process with rank i sends data to $(i + k) \bmod p$ and receives from $(i - k + p) \bmod p$. Therefore, for the initial and

the final $c-1$ steps, the communications are not purely internode. The PEA uses *exclusive-or* operation to determine the rank of processes for data exchange. It is similar to the BIA in terms of communication phase since after step k where $k = c$, the communication operation remains internode until the end.

Energy Saving Strategy. Because all three algorithms exhibit purely internode communications at a certain step k, the following energy saving strategy may be applied in stages to each of them.

Stage 1 At the start of all-to-all, scale down the frequency of all the cores involved in the communication to the minimum.

Stage 2 During the communication phase, throttle all the cores to the state T_5 in step k if
- ◦ BIA: $2^k \geq c$,
- ◦ STRF: $c \leq k < p - c$.
- ◦ PEA: $k > c$.

Stage 3 For STRF: throttle to state T_0 at the communication step $k = p - c$.

Stage 4 At the end of all-to-all, throttle all the cores to state T_0 (if needed) and restore their operating frequency to the maximum.

Rank Placement Consideration. MVAPICH2 provides two formats of rank placements on multicores, namely *block* and *cyclic*. In the block strategy, ranks are placed such that any node j $(j = 0, 1, \ldots, N-1)$ contains ranks from $c \times j$ to $c \times (j+1) - 1$. In the cyclic strategy, all the ranks i belong to j if $(i \bmod N)$ equals j. The block rank placement calls for only two DVFS and throttling switches in the proposed energy saving strategy, and thus minimizes the switching overhead. In the cyclic rank placement, however, after a fixed number of steps the communication would oscillate between intra- and inter-node, requiring a throttling switch at every such step. Therefore, the block rank placement has been considered for the energy savings application.

2.1 Power Consumption Estimates

Let a multicore compute node has frequencies f_i, $(i = 1, \ldots, m)$, such that $f_1 < \ldots < f_m$, and throttling states T_j, $(j = 0, 1 \ldots, n)$. When all the c cores of the node execute an application at frequency f_i, each core consumes the dynamic power P_i proportional to f_i^3. Let P_{ij} be the dynamic power consumed by the entire node at the frequency f_i and throttling state T_j, P_s be the total static power consumption, and P_d be the dynamic power consumption of the compute node components, such as memory, disk, and NIC, which are different from the processor. Then, the power consumption with no idle cycles (at T_0) may be assumed as $P_{i0} = c \times P_i + P_s + P_d$, so, at T_j, it is

$$P_{ij} = \frac{j \times (P_s + P_d) + (n - j)(P_{i0})}{n} \ . \tag{1}$$

The P_{i0} expression serves just to give an idea of the effect of frequency scaling on power consumption. It may vary with the application characteristics since each application may have a different power consumption pattern depending its utilization of the compute node components.

Bruck Index Algorithm. Let t_{intra} and t_{inter} be the time to transfer a singe byte of data in an intra node and inter node message transfer respectively. Let $C_{net}(x)$ and $C_{mem}(y)$ be the parameters to consider the effect of network and memory contention where x and y processes are involved in inter node and intra node message transfer within a node respectively. Also, consider O_{dvfs} and $O_{throttle}$ be the overheads associated with a frequency scaling and throttling switch. So for a cluster having N nodes, each node having c cores, an all-to-allmessage exchange of total M bytes will undergo communication in two stages. In the first stage message transfers are both inter and intra node in nature. The total communication time for the first stage, can be written as,

$$T_{B1} = O_{dvfs} + \sum_{i=1}^{t} max(t_{intra} C_{mem}(c - 2^{i-1})\frac{M}{2}, t_{inter} C_{net}(2^{i-1})\frac{M}{2}), \quad (2)$$

where t=$\lceil \log_2 c \rceil$. In the second stage of the all-to-alloperation, all the message transfers are inter node in nature and the time spent in this phase can be expressed as,

$$T_{B2} = 2 \times O_{throttle} + (r - t)(t_{inter} + O_5(\frac{M}{2}))C_{net}(c)\frac{M}{2} + O_{dvfs}, \quad (3)$$

where r=$\lceil \log_2 p \rceil$ and $O_j(\frac{M}{2})$ is the increase in the inter node transfer time for a message size $\frac{M}{2}$, due to throttling state T_j.

The power consumption during the first stage of the communication is P_{10} across a node, since the execution is at the minimum frequency and no throttling is applied. Whereas in the second stage, the power consumption is P_{15} as throttling level T_5 is applied. Therefore the average power consumption for the proposed algorithm can be expressed as ,

$$\bar{P}_B = \frac{P_{10} T_{B1} + P_{15} T_{B2}}{T_{B1} + T_{B2}}. \quad (4)$$

As the number of nodes increase in a cluster, the communication time for the second stage dominates and hence the average power consumption approximately becomes P_{15}.

STRF. For this algorithm, communication takes place in three stages where in the the first and third stage, intra node and inter node message transfers take place and in second stage, message transfers are purely inter node.The communication time for the first and third stage can be expressed as,

$$T_{S1} = T_{S3} = O_{dvfs} + \sum_{i=1}^{c-1} max(t_{intra} C_{mem}(c - i)\frac{M}{p}, t_{inter} C_{net}(i)\frac{M}{p}). \quad (5)$$

The communication time for stage 2 can be expressed as,

$$T_{S2} = 2 \times O_{throttle} + (p + 1 - 2 \times c)(t_{inter} + O_5(\frac{M}{p}))C_{net}(c)\frac{M}{p}. \qquad (6)$$

So, the average power consumption for the proposed energy saving algorithm when STRF algorithm is used for all-to-alloperation is,

$$\bar{P}_S = \frac{2 \times P_{10}T_{S1} + P_{15}T_{S2}}{2 \times T_{S1} + T_{S2}}. \qquad (7)$$

Similar to the BIA, the execution time for the second stage dominates the total execution time for all-to-alloperation in the STRF algorithm and therefore, the average power consumption is close to P_{15}.

3 Experimental Results

The computing platform used comprises ten Infiniband-connected compute nodes, each of which has 16 GB of main memory and two Intel Xeon E5450 Quad core processors arranged as two sockets with the operating frequency ranging from 2.0 GHz to 3.0 GHz and the eight levels of throttling from T_0 to T_7. For measuring the node power and energy consumption, a Wattsup[5] power meter is used with a sampling rate of 1 Hz. Due to such a low measuring resolution, a large number of all-to-all operations have to be performed.

OSU MPI Benchmarks. This set of benchmarks[6] are used here to determine the change in execution time and power consumption of "stand alone" all-to-all operations. From Fig. 3(right), it can be observed that the execution time for all-to-all has very low performance penalty when the proposed energy savings are used. The average performance loss observed for various message sizes was just 0.97% of that for the *Full power* case. While somewhat higher than in the *DVFS only* case, which was 0.5%, it is quite acceptable taking into the consideration large reductions in the power consumption achieved (Fig. 3(left)) with the *Proposed* strategy. Note, however that, in all the cases, the power consumption increases with the message size since the memory dynamic power consumption increases because of message copying [13]. Similar power reductions have been obtained for the all-to-all-vector operation.

The static power consumption P_s of a node was around 150 watts as measured by the Wattsup power meter. The most significant contribution to the dynamic power consumption P_d of the components in a node apart from the processor, comes from the memory. For measuring memory dynamic power consumption, an extrapolation method discussed in [7] was used since each node has four modules of 4 GB each. So, the memory dynamic power consumption, P_d was

[5]https://www.wattsupmeters.com

[6]OSU MPI Benchmarks: http://mvapich.cse.ohio-state.edu

determined to be around 20 watts. When the all-to-all message exchange is for 1 MB, P_{15} can be calculated as,

$$P_{15} = \frac{5 \times (150 + 20) + 3 \times 222}{8} = 189.5. \tag{8}$$

It can be observed from Fig. 3(right) that the value of P_{15} calculated in (8) is close to the value of average power consumption for an all-to-all message exchange of 1 MB.

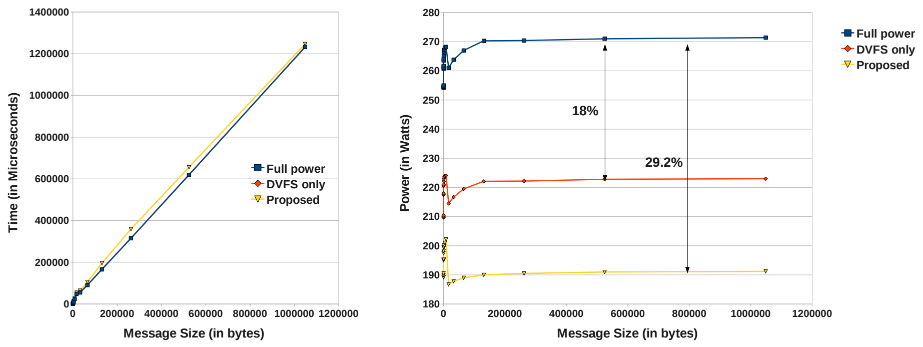

Fig. 3. The all-to-all execution time on 80 processes (left) and the power consumption across a compute node (right) for the three cases: Executing at the highest frequency and no throttling (`Full power`); Only frequency scaling without throttling (`DVFS only`); and Using the proposed energy saving strategies (`Proposed`).

CPMD and NAS Application Benchmarks CPMD (CarParrinello Molecular Dynamics)[7] is a ab-initio quantum mechanical molecular dynamics technique using pseudopotentials and a plane wave basis set. Eleven input sets are used here. MPI_Alltoall is the key collective operation in CPMD. Since most messages have the sizes in the range of 128 B to 8 KB, the BIA is used. From the NAS benchmarks [1], FT and IS Class C benchmarks are chosen because they use the all-to-all operation. Fig. 4 shows the execution time and energy consumption of CPMD and NAS benchmarks on 80 and 64 processes, respectively, normalized to the `Full power` case. For the CPMD with the `Proposed` strategies, the performance loss ranges from 0.4% to 4.3% averaging 2.78% leading to the energy savings in the range of 9.8% to 15.7% (13.4% on average). For the NAS, the performance loss ranges from 1.1% to 4.5% and the average energy savings are about 10%. Hence, the benchmark applications tested suffer from little performance loss and have significant energy savings.

[7]CPMD Consortium: http://www.cpmd.org

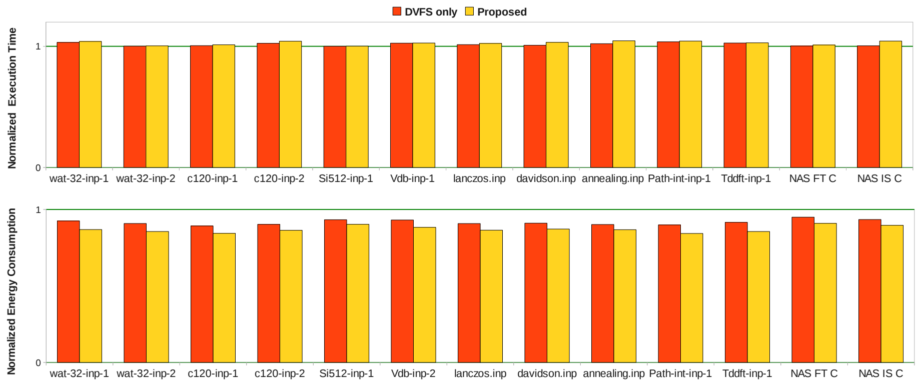

Fig. 4. Execution time (up) and energy consumption (down) of 11 CPMD inputs on 80 processors and of NAS benchmarks on 64 processes for the `DVFS only` and `Proposed` cases normalized to the `Full power`.

4 Related Work

In [13], authors study the power efficiency and communication performance of RDMA over TCP/IP and conclude that RDMA performs better for the both cases. The energy efficiency delivered by the modern interconnects in high performance clusters is discussed in [4]. The communication phase characterization to obtain energy savings by using DVFS is done in [3][5][6]. Performance counter based algorithms for determining stall cycles to save energy are used in [11][10][8]. In [9], authors have developed a tool which estimates power consumption characteristics of a parallel application in terms of various CPU components. In [12], algorithms to save energy in the collectives, such MPI_Alltoall and MPI_Broadcast, are proposed. However, they differ significantly with the approach presented in this paper. Specifically, [12] assumes that throttling has a negative effect on internode communication. Thus, the authors of [12] redesign the all-to-all operation, such that a set of sockets does not take part in communication a some point of time, and thus, may throttled. But as the number of cores within a node keep on increasing, this approach of forcing the sockets to remain idle during communication, can introduce significant performance overheads. The power saving achieved in [12] are be equivalent to executing the two sockets in a node at the minimum frequency and throttling state T_4, whereas the approach proposed here achieves better power saving by keeping both sockets at minimum frequency and throttling to a higher state T_5. The detailed experimental comparison of the two algorithms is left as future work.

5 Conclusions

Energy-saving strategies have been proposed for the all-to-all operation and implemented as the MPI_Alltoall collective in MVAPICH2 *without* modifying the standard algorithms used to perform this operation. The sensitivity of inter- and intranode message transfers to CPU throttling has been assessed, and it was observed that throttling has almost no negative effect on the performance of internode communications. Thus, both DVFS and CPU throttling were applied in the appropriate communication steps within three different all-to-all algorithms used by MVAPICH2. The experiments demonstrate that the proposed strategies can deliver up to 15.7% energy savings, without introducing significant performance overhead for the CPMD and NAS application benchmarks, which reflect the potential beneficial effect on scientific applications in general.

Similar energy saving strategies may be extended to other collectives including reduction operations. Furthermore, as the number of cores within a node keeps increasing, the opportunity of applying throttling in intranode communication must be also explored. The point-to-point operations should also be studied for energy savings since many modern architectures may support socket level DVFS and CPU throttling.

References

1. D. H. Bailey et al. The nas parallel benchmarks-summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, Supercomputing '91, pages 158–165, New York, NY, USA, 1991. ACM.
2. J. Park et al. Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 419–424, 2010.
3. M. Y. Lim et al. Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.
4. R. Zamani et al. A feasibility analysis of power-awareness and energy minimization in modern interconnects for high-performance computing. In *Proceedings of the 2007 IEEE International Conference on Cluster Computing*, CLUSTER '07, pages 118–128, Washington, DC, USA, 2007. IEEE Computer Society.
5. V. W. Freeh et al. Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01*, IPDPS '05, pages 4.1–, Washington, DC, USA, 2005. IEEE Computer Society.
6. V. W. Freeh et al. Using multiple energy gears in mpi programs on a power-scalable cluster. In *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '05, pages 164–173, New York, NY, USA, 2005. ACM.
7. Xizhou Feng, Rong Ge, and Kirk W. Cameron. Power and energy profiling of scientific applications on distributed systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01*, IPDPS '05, pages 34–, Washington, DC, USA, 2005. IEEE Computer Society.
8. R. Ge, X. Feng, W. Feng, and K.W. Cameron. CPU MISER: A performance-directed, run-time system for power-aware clusters. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, page 18, 2007.

9. R. Ge, X. Feng, S. Song, H. C. Chang, D. Li, and K.W. Cameron. PowerPack: Energy profiling and analysis of high-performance systems and applications. *Parallel and Distributed Systems, IEEE Transactions on*, 21(5):658–671, May 2010.
10. C. H. Hsu and W. Feng. A power-aware run-time system for high-performance computing. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, page 1, nov 2005.
11. S. Huang and W. Feng. Energy-efficient cluster computing via accurate workload characterization. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 68–75, May 2009.
12. K. Kandalla, E.P. Mancini, S. Sur, and D.K. Panda. Designing power-aware collective communication algorithms for infiniband clusters. In *Parallel Processing (ICPP), 2010 39th International Conference on*, pages 218–227, sept. 2010.
13. J. Liu, D. Poff, and B. Abali. Evaluating high performance communication: a power perspective. In *Proceedings of the 23rd international conference on Supercomputing*, ICS '09, pages 326–337, New York, NY, USA, 2009. ACM.
14. R. Thakur and R. Rabenseifner. Optimization of collective communication operations in mpich. *International Journal of High Performance Computing Applications*, 19:49–66, 2005.