

Designing energy efficient communication runtime systems: a view from PGAS models

Abhinav Vishnu · Shuaiwen Song ·
Andres Marquez · Kevin Barker ·
Darren Kerbyson · Kirk Cameron · Pavan Balaji

© Springer Science+Business Media, LLC 2011

Abstract As the march to the exascale computing gains momentum, energy consumption of supercomputers has emerged to be the critical roadblock. While architectural innovations are imperative in achieving computing of this scale, it is largely dependent on the systems software to leverage the architectural innovations. Parallel applications in many computationally intensive domains have been designed to leverage these supercomputers, with legacy two-sided communication semantics using Message Passing Interface. At the same time, Partitioned Global Address Space Models are being designed which provide global address space abstractions and one-sided communication for exploiting data locality and communication optimizations.

A. Vishnu (✉) · A. Marquez · K. Barker · D. Kerbyson
High Performance Computing Group, Pacific Northwest National Lab, Richland, WA, USA
e-mail: abhinav.vishnu@pnl.gov

A. Marquez
e-mail: andres.marquez@pnl.gov

K. Barker
e-mail: kevin.barker@pnl.gov

D. Kerbyson
e-mail: darren.kerbyson@pnl.gov

S. Song · K. Cameron
Scalable Computing Lab, Virginia Polytechnic Institute, Blacksburg, VA, USA

S. Song
e-mail: s562673@cs.vt.edu

K. Cameron
e-mail: cameron@cs.vt.edu

P. Balaji
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA
e-mail: balaji@mcs.anl.gov

Published online: 06 October 2011

 Springer

PGAS models rely on one-sided communication runtime systems for leveraging high-speed networks to achieve best possible performance.

In this paper, we present a design for Power Aware One-Sided Communication Library – PASCoL. The proposed design detects communication slack, leverages Dynamic Voltage and Frequency Scaling (DVFS), and Interrupt driven execution to exploit the detected slack for energy efficiency. We implement our design and evaluate it using synthetic benchmarks for one-sided communication primitives, *Put*, *Get*, and *Accumulate* and uniformly noncontiguous data transfers. Our performance evaluation indicates that we can achieve significant reduction in energy consumption without performance loss on multiple one-sided communication primitives. The achieved results are close to the theoretical peak available with the experimental test bed.

Keywords Communication runtime system · DVFS · Energy efficiency · InfiniBand

1 Introduction

As we move forward to the next step of exascale computing, energy consumption of systems is expected to be a significant hindrance in naively increasing the computational power by three orders of magnitude from current petascale systems. For example, the US Department of Energy estimates that in order to be able to sustain an exaflop machine, its power consumption cannot be more than ten-fold that of current petaflop machines [1]. That is, we need to achieve a thousand-fold increase in performance, while allowing the power consumption to increase by only ten-fold, and hence energy efficiency must improve by hundred-fold.

Parallel applications in a wide range of scientific domains are being designed to use the proposed exascale systems. Message Passing Interface [2, 3] has become the *de facto* standard for writing these applications. However, several of these scientific domains are a natural fit for Partitioned Global Address Space (PGAS) models [4–7]. These models provide abstractions for distributed data structures (Arrays, Trees, etc.) and primitives for one-sided data transfer to provide load balancing with different execution paradigms. PGAS models use one-sided communication runtime systems to achieve scalability and high performance, while providing abstractions from variability of networks.

As the runtime systems continue to evolve, many architectural innovations for energy efficient computing are being proposed in the literature and becoming available with commodity architectures. User-space abstractions such as Dynamic Voltage and Frequency Scaling (DVFS) have become available, which allow a user process to dynamically change the frequency and voltage of processing elements. High-speed networks such as InfiniBand [8], BlueGene [9], and Quadrics [10] provide methods for interrupt based notification of data transfer—a powerful mechanism which may be used to exploit communication slack.

In this paper, we design a Power Aware One-Sided Communication Library (PASCoL) using Aggregate Remote Memory Copy Interface (ARMCI) [11], which leverages the architectural and network abstractions to exploit the communication

slack and achieve energy efficiency. We lay down the design issues of various one-sided communication primitives and associated communication protocols for different datatypes, specifically focusing on contiguous and uniformly noncontiguous datatypes as a use case from many scientific applications. We implement our design and evaluate it using synthetic benchmarks on an InfiniBand Cluster. Our performance evaluation with benchmarks using various one-sided communication primitives shows that we can achieve significant energy efficiency with negligible performance degradation. The observed energy efficiency is close to the theoretical peak provided by the experimental test bed.

The rest of the article is organized as follows. In Sect. 2, we present the background of our work. In Sect. 3, we present the design of energy efficient communication runtime system—PASCoL using ARMCI. In Sect. 4, we present the performance evaluation of PASCoL using synthetic benchmarks on an InfiniBand cluster. We present the related work in Sect. 5. We conclude and present our future directions in Sect. 6.

2 Background

2.1 One-sided communication runtime systems

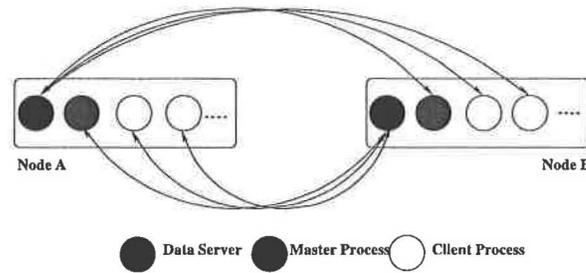
Many one-sided communication runtime systems have emerged to serve the requirements of programming models. MPI-Remote Memory Access, Global Address Space Network (GASNet) [4], Aggregate Remote Memory Copy Interface (ARMCI) [11], Low Level API (LAPI) [12], and Deep Computing Messaging Framework (DCMF) [13], are examples of one-sided communication runtime systems, which provide put, get, and accumulate communication primitives. We specifically focus on ARMCI [11] in this paper.

The ARMCI [11] communication runtime system provides a general-purpose, efficient, and widely portable one-sided communication operations optimized for contiguous and noncontiguous (strided, scatter/gather, I/O vector) data transfers. In addition, ARMCI includes a set of atomic and mutual exclusion operations. ARMCI exploits native network communication interfaces and system resources (such as shared memory) to achieve the best possible performance of the remote memory access/one-sided communication. Optimized implementations of ARMCI are available for the Cray Portals, Myrinet (GM and MX) [14], Quadrics [10], Gigaset (VIA), and InfiniBand (using OpenFabrics and Mellanox Verbs API) [8, 15–18]. It is also available for leadership class machines including Cray XT4, XT5, XE6, and BlueGene/P [13].

Figure 1 shows the communication structure in ARMCI. The terminology between processes on the same node is differentiated to facilitate the implementation of one-sided communication primitives. The process with lowest rank on a node is called *master* and the rest of the processes on the node are called *clients*. The master process creates a thread, *data server*, which is used as an agent for remote asynchronous progress.

A request for global memory allocation is served by a shared memory segment visible to all processes on a node. One-sided communication occurs only between global address space data, precluding the requirement of client–client communication. The

Fig. 1 Communication Structure in ARMCI



data server is used in designing protocols which may be efficiently implemented using copy based approach. Efficient protocols which require bulk atomic updates (such as accumulates) may also be designed using the data server. Depending on the workload, the network and the communication protocol, the data server *may or may not need to be active all the time*.

2.2 Overview of power conservation approaches for high performance systems

Multiple researchers are exploring smart utilization of power and energy for large scale high performance clusters with parallel applications. Some researchers have applied power efficient strategies at the architectural level to the supercomputer. IBM Blue Gene series [19] and Green Destiny [20] use low frequency processors to build energy efficient systems. However, this approach requires a large amount of low power processors to achieve better energy consumption (as an example, Blue Gene/P consists of 73,728 quad core processors and consumes 2.3 MW of power [19]). For higher energy efficiency, Power Modes [21] techniques using integrated power-aware components have been provided for fine-grained control of high performance systems; these include low-power settings for network cards, spinning down disk drives when they are not in use, making systems sleep or remotely shut down components by smart external power devices like PDUs, etc. The challenge is to balance performance decreases and various low-power operations. While we explore power reduction approaches, high performance is of utmost priority to the HPC community, and we design PASCoL to minimize the performance penalty, while maximizing the energy efficiency.

Using software to dynamically control the power states of system level components has become one of the most popular techniques for power-aware computing. Dynamic voltage/frequency scaling (DVFS) is being widely used for reducing system power consumption during specific phases of parallel applications. Studies like [22–26] have applied DVFS to reduce CPU power consumption and discussed the tradeoffs between performance and energy efficiency. At node level, DVFS enables several levels (P-states) of frequencies that can be switched during runtime. Low and high power states are corresponding to low and high CPU performance utilization.

However, most of these studies have focused on achieving energy efficiency for two-sided communication. One-sided communication exhibits different properties and requires an asynchronous agent for communication. PASCoL, the focus of our

work relies on uniformly non-contiguous communication with support from asynchronous agent heavily for progress. PASCoL combines DVFS methodology and interrupt driven execution to achieve energy efficiency for one-sided communication primitives.

3 Overall design of an energy efficient communication runtime system

In this section, we present the overall design of an energy efficient one-sided communication runtime system. We explore the alternatives for energy efficiency—DVFS and Interrupt driven execution and use them to design energy efficient protocols for one-sided communication primitives.

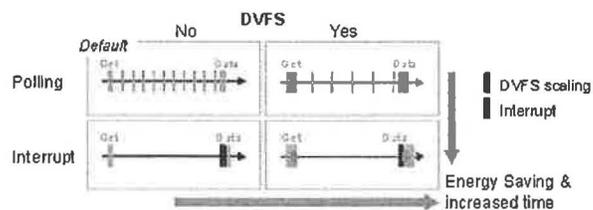
3.1 Mechanisms for energy efficiency

There are multiple mechanisms available for designing energy efficient one-sided communication protocols which are complementary. A combination of these protocols may be used as follows:

- Interrupt based execution allows multiple stages of communication protocols to transition using event driven mechanisms. As an alternative to polling—typically used in high performance computing applications, this method allows much lower CPU utilization to save energy, particularly if enough communication slack is available to be exploited.
- DVFS can improve energy efficiency by reducing the frequency and voltage of processors, typically on a per-core basis (frequency) and on a per-socket basis (voltage). Different communication protocols require varying CPU utilization during different phases. Energy efficiency can be achieved using DVFS, if the communication slack is much higher than the overhead of transitioning between frequency/voltage states.

The use of interrupt based execution and DVFS is shown in Fig. 2 using an example of a one-sided get operation. After the operation is executed, the data becomes available at a later point as shown on the time-line. The default case uses neither the interrupt nor DVFS mechanisms. Polling in this case is indicated by regular activities along the time-line between the get primitive and arrival of data (upper-left in the figure). Combining DVFS and polling (upper-right), the frequency of polling is reduced due to reduced processor frequency but at an expense of transitions between DVFS states. Polling is completely reduced by using interrupts but can increase in latency results due to interrupt handling.

Fig. 2 Mechanisms for energy optimizations



Keeping the mechanisms discussed above in mind, we design and implement an energy efficient one-sided communication runtime system. While our framework allows full DVFS scaling due to the limitations of our experimental setup, we are able to use frequency scaling only.

3.2 Energy efficient communication protocols for one-sided communication

In this section, we discuss the energy efficient communication protocols for one-sided communication primitives. The protocols are classified using datatypes—contiguous and noncontiguous.

3.3 Energy efficient protocols for contiguous data transfer

One-sided communication of contiguous data transfer is the primary case for using the Remote Direct Memory Access (RDMA) mechanism provided by most Interconnects such as InfiniBand [8], Quadrics [10], BlueGene [13], Cray Gemini. RDMA mechanism requires the source and target buffer to be registered for most networks.

Let *source* and *target* represent the source and the target buffers respectively used for the one-sided communication primitive. Let *registered* be the function which checks whether a buffer has been registered. The following algorithm is executed at the client process:

```

if registered(source) && registered(target) then
  Use RDMA method
else
  Copy Data in Intermediate Transmission Buffer
  Send Data
end if

```

When the source and target buffers are registered, the overall communication slack is entirely the network data transfer. A combination of Interrupt based execution and DVFS is used depending on the expected communication slack. Section 4 helps us define the thresholds for using these mechanisms.

In the above algorithm, if either of the source or the target buffer is not registered, a copy-based communication protocol is used. Since data copy is compute intensive, the DVFS mechanism is used after the copy operation is complete. The interrupt driven execution is used after the data transfer request is completed.

3.4 Handling noncontiguous data transfer

Handling noncontiguous data types for energy efficiency is of critical importance to many application domains. Some of these domains use distributed arrays and perform communication on cartesian blocks of data. This results in uniformly noncontiguous (strided) data transfer. Energy efficient communication protocols for strided data transfer is pivotal for these applications.

Many communication protocols have been proposed for handling strided data communication. Some high-speed networks support strided communication natively

by using scatter/gather mechanisms [8]. However, these mechanisms result in high context memory utilization. Networks which provide high concurrency may also use pipelined data transfer. However, when individual data size is small, such a protocol results in significant overhead.

Let *flatten* be a utility which converts a strided data to a contiguous data by copying it in *Intermediate Transmission Buffers*. A simplified protocol which is applicable to a wide variety of strided communication primitives is presented below:

Flatten the strided buffer
Copy Data in Intermediate Transmission Buffer
Send Data

The algorithm for *flattening the strided buffer* is a recursive operation, resulting in pipelined data transfer. The interrupt driven execution may be used in conjunction with DVFS if the pipelined buffer size is above a particular threshold to exploit communication slack.

3.5 Energy efficient asynchronous agent

To provide asynchronous progress of one-sided communication operations on remote node(s), each node uses an asynchronous agent, such as the data server thread in ARMCI [11]. The asynchronous agent is not involved when RDMA is used for data transfer between user-level buffers. The asynchronous agent is active when a copy based protocol is used for data transfer. Hence, the agent may be active only during these phases of communication. In the PASCoL design, we use a combination of interrupt driven execution and DVFS for the asynchronous agent. The frequency is scaled up after an interrupt has been received and scaled down just before blocking on the interrupt based execution.

3.6 Discussion

In this section, we discuss the issues currently not considered in PASCoL. We specifically focus on atomic memory operations and synchronization methods:

Atomic memory operations Atomic memory operations are widely used in many applications for load balancing, enabling passive synchronization etc. In PASCoL, we have not considered optimizing atomic operations except accumulate operations, which are typically performed on a large data block. Word-based atomic operations are latency sensitive and using DVFS/interrupt based execution may result in significant overhead. We plan to address this limitation in the future that may provide guidelines for leveraging the interrupt based execution with DVFS.

Synchronization methods One-sided communication runtime systems provide active and passive modes of synchronization. With active synchronization, origin and target processes are involved in the synchronization. With passive synchronization, only the origin process is involved. ARMCI supports only active mode of synchronization. The synchronization operation may be optimized by possibly time-stamping

the outstanding requests, providing an estimate of the communication slack. Interrupt based execution with DVFS may be used if the expected communication slack is above a threshold. Currently, PASCoL does not handle energy efficient synchronization. We plan to address this limitation in the near future.

4 Power and performance evaluation of PASCoL

In this section, we present a performance evaluation of PASCoL using synthetic benchmarks designed with ARMCi communication primitives. For one-sided communication primitives—put, get, accumulate, and put strided, we present the relative latency, relative energy consumption per megabyte of transfer, and relative power consumption of a combination of DVFS and Interrupt/polling methodologies. We begin with a description of the Experimental Test bed.

4.1 Experimental test bed

We use the Northwest ICE (NW-ICE) test bed at Pacific Northwest National Lab for power and performance evaluation. The NW-ICE cluster has 192 compute nodes, inter-connected with DDR InfiniBand network adapters and switches. Each NW-ICE node is an Intel Xeon E5345 dual socket quad core CPU with 2.33 GHz frequency. Each node has 16 GBytes of main memory with each core having a 32 kB cache size. Using DVFS, NW-ICE allows frequencies of 2.33 GHz and 1.9 GHz. By default all processes execute at 2.33 GHz frequency. The interface for changing the frequencies is through a memory resident file system.

4.1.1 ESDC monitoring

Real-time data center energy efficiency depends on real-time data streaming from all the power consuming hardware in a data center, as well as data acquisition and reduction software. PNNL has developed a real-time software tool, FRED (Fundamental Research in Energy Efficient Data Centers), to monitor, analyze, and store data from the ESDC-TB facility instrumentation. FRED's underlying technology is derived from PNNL's experience in developing power plant, distribution, and facility monitoring and diagnostic systems for applications ranging from nuclear power generation to building management. We use the real-time software tool of FRED to analyze the energy consumption for various synthetic benchmarks.

FRED consists of the ESDC-TB monitoring system, the Environmental and Molecular Science (EMSL) facility monitoring system, a data collector, a central database, and a web-based graphical user interface (GUI) client. The ESDC-TB monitoring system derives from PNNL's Decision Support for Operations and Maintenance (DSOM) software (R&D100Award), an advanced, flexible diagnostic monitoring application for energy supply and demand systems. The ESDC-TB monitoring system interfaces to auxiliary data acquisition systems that collect data specific to NW-ICE.

4.2 Performance evaluation methodology

In this section, we present the performance methodology for evaluating the power and performance of PASCoL. We design pure communication benchmarks using the one-sided communication primitives—put, get, accumulate, and put strided.

To study the impact of approaches proposed in Sect. 3, we design a shift communication pattern benchmark using each of these communication primitives [27]. Unlike MPI based communication benchmark—which implicitly synchronizes the communicating processes, the shift benchmark designed with one-sided communication primitives synchronizes the memory associated with communication. The following algorithm presents an example of designing the shift benchmark using put primitive:

```
start timer
for  $j = 0$  to iterations do
  for  $i = 0$  to numprocs do
     $dest \leftarrow myid + i$ 
    put(data) to  $dest$ 
    fence to  $dest$ 
  end for
end for
end timer
```

Similarly, the shift benchmark is designed by replacing the corresponding put primitive with get, accumulate, and put strided primitives. A total of four combinations are used for comparison—polling, polling + DVFS, Interrupt, and Interrupt + DVFS. These combinations are used to compare the performance of evaluation metrics discussed below.

4.2.1 Evaluation metrics

There are three evaluation metrics which are used for evaluation of PASCoL and comparing the performance of various approaches presented above. The fundamental metric is the latency observed by each of the approaches. Another metric of interest is the power consumption of the approaches. However, each of the above metrics may not be individually sufficient. We propose a derived metric—Energy consumed relative to volume of data transfer (Energy/MByte). We specifically focus on the thresholds beyond which the power and energy/mbyte may be improved without an increase in latency.

4.3 Results

In this section, we present the evaluation of PASCoL for each of the communication primitives using the metrics presented above, while comparing the performance of the approaches—Polling, Polling + DVFS, Interrupt, Interrupt + DVFS. The performance results are normalized with the polling approach—the default methodology for most one-sided communication runtime systems.

Fig. 3 ARMCI put performance, latency

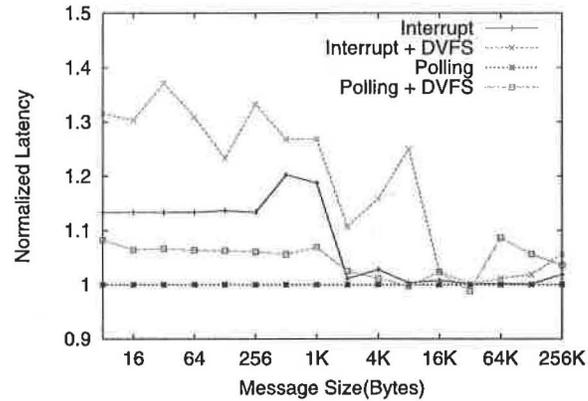
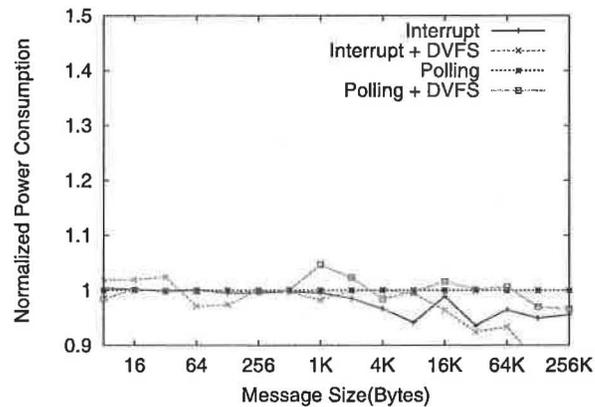


Fig. 4 ARMCI put performance, normalized power consumption



Figures 3, 4, and 5 show the normalized latency, power consumption and Energy/Mbyte for the shift communication benchmark using the put one-sided primitive on 64 processes, respectively. The polling approach outperforms other approaches for small and medium size messages, due to their sensitivity to latency and significant overhead of using the Interrupt and DVFS mechanisms. We observe spikes for Interrupt and DVFS based approaches, due to the limitations of our current test bed, as the sampling is available once every 5 seconds.

With increasing message size, the latency for multiple approaches converges significantly (less than 5% difference). At 16 KBytes message size, we observe that the latency for all approaches (with a slight exception to Polling + DVFS), converges. A similar trend is observed in the relative power consumption of these approaches. For messages at 256 KBytes, the Interrupt + DVFS approach provides an improvement of 12% in power consumption in comparison to the polling approach. Smaller improvements are also observed in the power consumption of other approaches.

At the same time, Energy/Mbyte consumption for the Interrupts + DVFS approach improves significantly compared to the other approaches. Overall, we observe an improvement of 8% in the Energy/Mbytes using Interrupts with DVFS compared to the

Fig. 5 ARMCI put performance, energy/MBytes

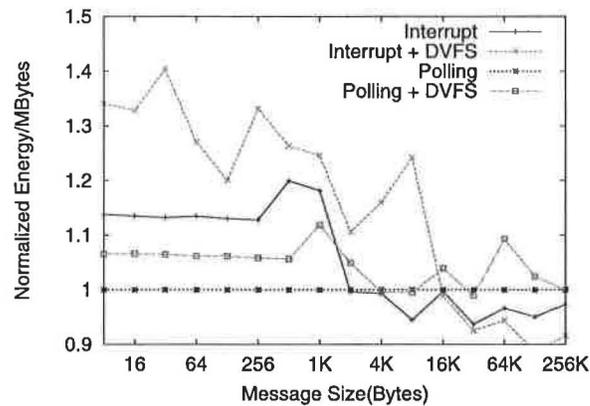
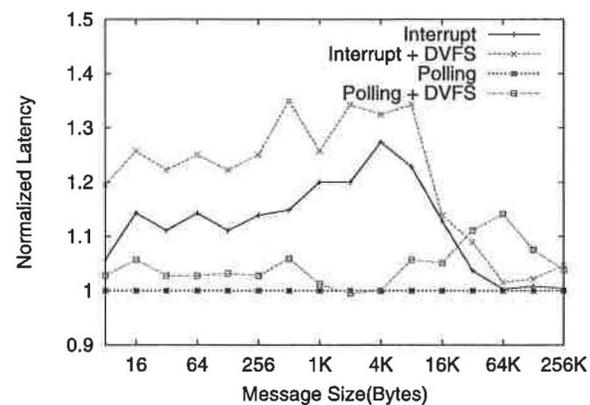


Fig. 6 ARMCI get performance, latency



default polling case, while an improvement of 5% is observed compared to the Interrupts scheme. For large messages, the overhead incurred by interrupts is amortized by the overall time of data transfer. We observe that a threshold of 16 KBytes can be used for using Interrupts with DVFS without significant performance degradation.

Figures 6, 7, and 8 show the normalized latency, Power consumption and Energy/Mbyte for the shift communication benchmark using the get one-sided primitive on 64 processes, respectively. We observe the trends in the performance similar to the shift communication benchmark designed using the put communication primitive. The polling approach outperforms other approaches for all the evaluation metrics and small messages. The limitations of sampling rate produces spikes, which are prominent for smaller messages and less for the larger messages. An improvement of about 11% is observed in the relative power consumption by Interrupt + DVFS approach, while much lesser improvements are observed for Interrupt and DVFS only approaches. The Interrupt approach produces an outlier at 4 KBytes message, which is still under observation. Similar trends are observed for energy/mbytes metric, where interrupts + DVFS approach outperforms other approaches for larger messages, but incurs significant overhead for small messages.

Fig. 7 ARMC1 get performance, normalized power consumption

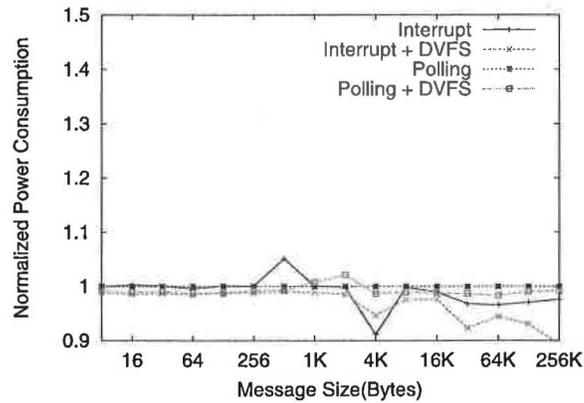
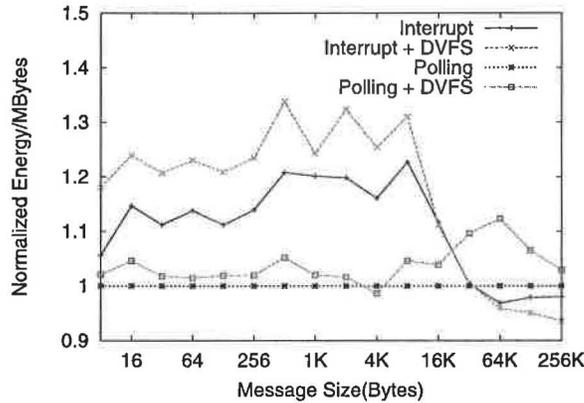


Fig. 8 ARMC1 get performance, energy/MBytes



Figures 9, 10, and 11 show the normalized latency, Power consumption and Energy/Mbyte for the shift communication benchmark using the accumulate one-sided primitive on 64 processes, respectively. The shift communication benchmarks using put, and get primitives use RDMA and the associated communication protocol does not involve the asynchronous agent. The accumulate one-sided communication primitive uses the pipelined data transfer, and involves the asynchronous agent for remote progress.

As presented in Sect. 3, the pipelined communication protocol flattens the buffer and uses the copy based approach. The copy phase and the atomic update phases of the protocol are CPU intensive. Hence, we do not use DVFS during this phase. As a result, the overall improvement in relative power consumption and energy/mbyte is reduced in comparison to the contiguous data transfer. The improvement in relative power consumption is 8%, while the improvement in Energy/Mbyte is about 6%.

Figures 12, 13, and 14 show the normalized latency, power consumption and energy/Mbyte for the shift communication benchmark using the put strided one-sided primitive on 64 processes, respectively. The strided communication primitives use copy based approach for data transfer. Similar to the shift communication bench-

Fig. 9 ARMCI accumulate performance, latency

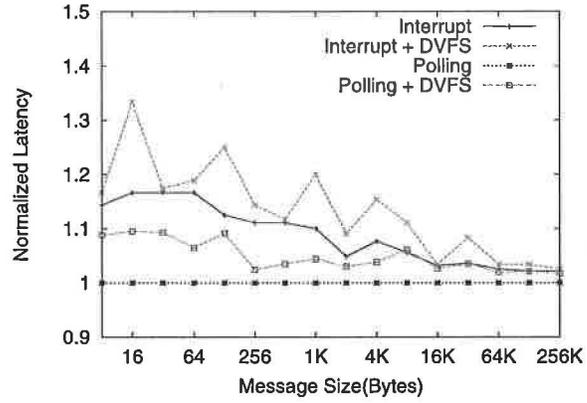


Fig. 10 ARMCI accumulate performance, normalized power consumption

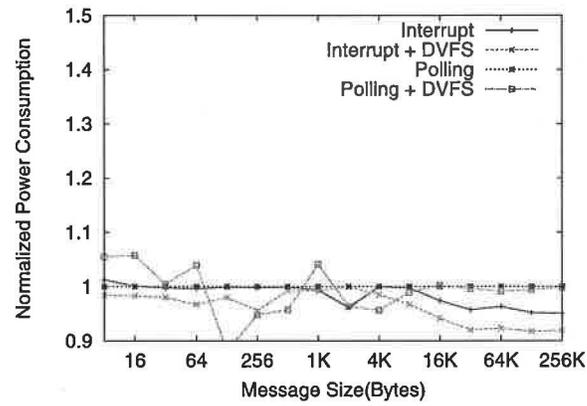
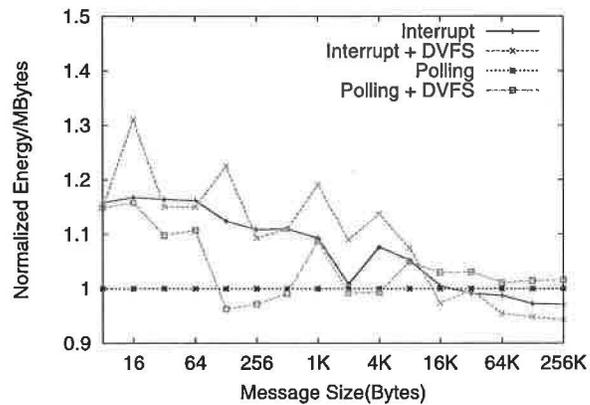


Fig. 11 ARMCI accumulate performance, energy/MBytes



mark using accumulate primitive, the relative latencies converge for messages beyond 32 Kbytes and significant improvements in the relative power consumption and energy/mbyte are also observed.

Fig. 12 ARMCI put strided performance, latency

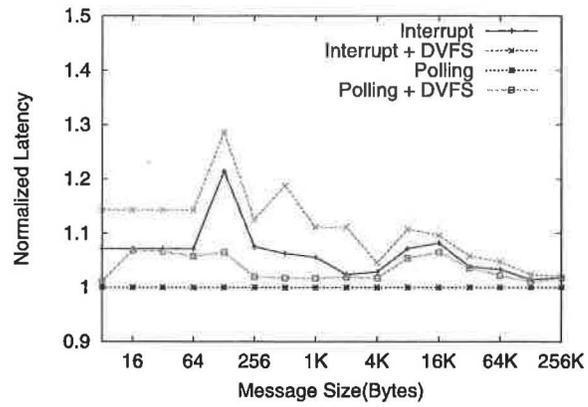


Fig. 13 ARMCI put strided performance, normalized power consumption

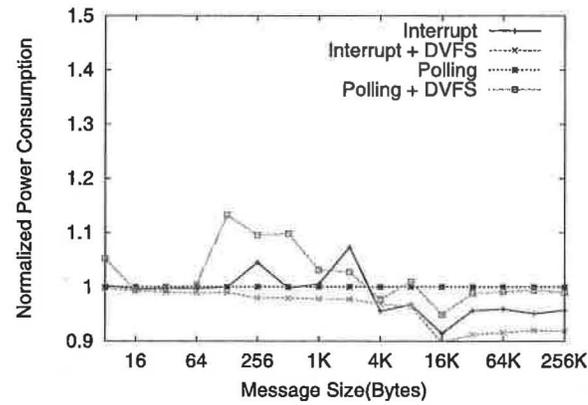


Fig. 14 ARMCI put strided performance, energy/MBytes

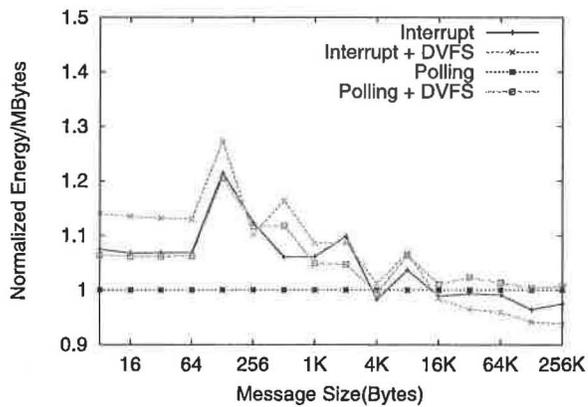


Table 1 Energy efficiency advantages for different communication semantics

Primitive	Min message size (KB)	Increase in latency (%)	Decrease in energy (%)
Get	32	5	6
Put	16	5	6
Accumulate	16	3	5
Strided Put	16	2	6

4.4 Discussion

As presented in the previous section, a combination of interrupt based execution and DVFS mechanisms provides the maximum energy efficiency for large messages with each of the communication primitives. A summary of the results is presented in the Table 1 for each of the put, get, accumulate, and strided put primitives. In this table, we also indicate the increase in latency resulting from the DVFS transitions and the interrupt handling, and also the overall decrease in energy consumption. These metrics are presented relative to the default case of no DVFS and use of polling. Also listed in Table 1 is the message size at which increased energy efficiency is observed.

As we described in the previous section, the NW-ICE test bed used for the experimentation has its limitations. In particular, there are only two processor-core frequency levels available—1.9 GHz and 2.33 GHz. This limits the potential energy efficiency that may be observed. Assuming that the processor core power consumption is directly proportional to the frequency then the difference in these two states represents a power difference of 22% and affects just the power used by the processors. Measurements made on NW-ICE on a rack basis indicate that an idle rack consumes 7.8 kW, and a rack containing all processor-cores performing the all-to-all benchmark consumes 9.4 kW resulting in a 16% difference. These two observations are inline with each other as the rack-based measurements include everything in the rack and not just the processors.

The energy improvements observed, as listed in Table 1 shows that a significant portion of the 16% maximum savings is being realized when using the Interrupt and DVFS energy saving mechanisms. Current state of the art processors including the Intel Nehalem series, and the AMD Magny-Cours have a greater number of DVFS states in comparison to our experimental test bed. In addition, the overhead of transitioning between DVFS states is expected to reduce in future processor generations. These two factors impact expected energy savings in two ways:

- The greater number of DVFS states has a greater potential for energy savings for large-messages when using the combination of Interrupt and DVFS mechanisms for one-sided communications.
- The decrease in transition overheads should reduce the message size at which energy savings will occur.

5 Related work

Multiple researchers have focused on exploring accurate component and system level power/energy profiling approaches. Other researchers have designed and developed techniques to efficiently reduce the total power consumption without incurring performance penalty. State of the art methodologies focus on measuring the aggregate power consumption of entire system or building level power [28] through proprietary hardware [29], power panels, or empirical estimations by rules-of-thumb [30]. Many studies, including both simulations and empirical analysis, have also explored evaluation of individual system components such as processor [31, 32], memory [31], disk [33–35], motherboard [36], CPU and system fan control [37], and interconnection networks [38]. Due to a high demand for fine-grained system-wide component level power/energy profiling tools, Ge et al., have designed and developed a power/energy/performance profiling infrastructure—PowerPack [36] to evaluate energy efficiency and power-aware techniques for parallel applications. Song et al. have used PowerPack to study the power characteristics of multiple suites in HPC benchmark [39] at a high granularity [40]. Most of the studies mentioned above have considered evaluation of workloads in context of single node, considering mechanisms such as DVFS. Recently, Kandalla et al., have presented a design for power efficient collective communication algorithms [41]. However, the design is not applicable for one-sided communication primitives which do not exhibit regular communication structure as collective communication primitives. To facilitate this, we have designed and implemented PASCoL, which serves this purpose.

Multiple researchers have also focused on reducing total power consumption during runtime without incurring performance penalty. One of the most common approaches to achieve this is to save CPU power during communication phases by applying Dynamic Voltage/Frequency Scaling (DVFS), since CPU consumes most power in system-wide for most current architectures [36, 40]. Many researchers have discussed the tradeoff between performance and energy consumption for scientific applications such as NAS Parallel Benchmark [23, 42–45]. They have pointed out the importance of efficient detection of communication regions during runtime [42, 44]. In [44], researchers also combine DVFS with concurrency throttling technique on multicore systems to explore the right combination of “switches” (frequency level and number of cores being utilized) for saving power. Instead of locating communication phases, work such as [46] monitors system performance counters to estimate workload in order to predict the proper frequency for next time interval on a single node. Researchers in [47] and [41] propose energy saving approaches using DVFS and CPU throttling for collective communication primitives. Liu et al. have provided a detailed empirical study of the benefits of power efficiency of RDMA compared to the traditional communication protocols such as TCP/IP [48]. However, this work has been done using verbs level interface, and does not provide guidance for higher level communication protocols for implementation.

None of the studies mentioned above have explored design challenges for one-sided communication runtime systems, while recent work has focused on designing energy efficient collective communication primitives. To address this limitation of state of the art research, we present PASCoL, which provide power efficient and high performance communication runtime system for one-sided primitives.

6 Conclusions and future work

In this paper, we have designed a Power Aware One-Sided Communication Library (PASCoL) using Aggregate Remote Memory Copy Interface (ARMCI) [11], which leverages the architectural and network abstractions to exploit the communication slack for energy efficiency. We have laid down the issues involving various one-sided communication primitives and associated communication protocols for different datatypes, specifically focusing on contiguous and uniformly noncontiguous datatypes as a use case from many scientific applications. We have implemented our design and evaluated it using synthetic benchmarks on an InfiniBand Cluster. Our performance evaluation with benchmarks using various one-sided communication primitives has demonstrated that we can achieve significant energy efficiency with negligible performance degradation. The observed energy efficiency is close to the theoretical peak provided by the experimental test bed.

We plan to continue design and development of energy efficient one-sided communication protocols for different platform and high speed communication networks. We also plan to evaluate the efficacy of these designs on large scale systems using scientific applications such as NWChem [49] and Subsurface Transport over Multiple Phases (STOMP) [50].

Acknowledgements This work was supported in part by the National Science Foundation Grant #0702182 and by Office of Advanced Scientific Computing Research, Office of Science, US Department of Energy, under Contract DE-AC02-06CH11357.

References

1. Crosscutting Technologies for Computing at the Exascale. <http://extremecomputing.labworks.org> (2010)
2. Gropp W, Lusk E, Doss N, Skjellum A (1996) A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Comput* 22(6):789–828
3. Geist A, Gropp W, Huss-Lederman S, Lumsdaine A, Lusk EL, Saphir W, Skjellum T, Snir M (1996) MPI-2: Extending the message-passing interface. In: *Euro-Par*, vol I, pp 128–135
4. Husbands P, Jancu C, Yelick KA (2003) A performance analysis of the Berkeley UPC compiler. In: *International conference on supercomputing*, pp 63–73
5. Nieplocha J, Harrison RJ, Littlefield RJ (1996) Global arrays: a nonuniform memory access programming model for high-performance computers. *J Supercomput* 10(2):169–189
6. Charles P, Grothoff C, Saraswat V, Donawa C, Kielstra A, Ebcioğlu K, von Praun C, Sarkar V (2005) X10: an object-oriented approach to non-uniform cluster computing. In: *OOPSLA '05: Proceedings of the 20th annual ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications*. ACM, New York, pp 519–538
7. Chamberlain BL, Callahan D, Zima HP (2007) Parallel programmability and the Chapel language. *Int J High Perform Comput Appl* 21(3):291–312
8. InfiniBand Trade Association (2004) *InfiniBand Architecture Specification*, Release 1.2, October 2004
9. Yu H, Chung I-H, Moreira J (2006) Blue gene system software—topology mapping for blue gene/l supercomputer. In: *SC '06: Proceedings of the 2006 ACM/IEEE conference on supercomputing*. ACM, New York, p 116
10. Pettini F, Feng W, Hoisie A, Coll S, Frachtenberg E (2002) The quadrics network: high-performance clustering technology. *IEEE MICRO* 22(1):46–57
11. Krishnan M, Vishnu A, Palmer B (2010) Aggregate remote memory copy interface

12. Shah G, Nieplocha J, Mirza JH, Kim C, Harrison RJ, Govindaraju R, Gildea KJ, DiNicola P, Bender CA (1998) Performance and experience with LAPI—a new high-performance communication library for the IBM RS/6000 SP. In: IPPS/SPDP, pp 260–266
13. Kumar S, Dozsa G, Almasi G, Heidelberger P, Chen D, Giampapa ME, Blocksome M, Faraj A, Parker J, Ratterman J, Smith B, Archer CJ (2008) The deep computing messaging framework: generalized scalable message passing on the Blue Gene/P supercomputer. In: ICS '08: Proceedings of the 22nd annual international conference on supercomputing, pp 94–103
14. Boden NJ, Cohen D, Felderman RE, Kulawik AE, Seitz CL, Seizovic JN, Su W (1995) Myrinet: a gigabit-per-second local area network. *IEEE MICRO* 15(1):29–36
15. Vishnu A, Mamidala A, Narravula S, Panda DK (2007) Automatic path migration over InfiniBand: early experiences. In: Proceedings of third international workshop on system management techniques, processes, and services, held in conjunction with IPDPS'07, March 2007
16. Vishnu A, Mamidala AR, Jin H-W, Panda DK (2005) Performance modeling of subnet management on fat tree InfiniBand networks using OpenSM. In: Proceedings of first international workshop on system management techniques, processes, and services, held in conjunction with IPDPS'07
17. Narravula S, Mamidala A, Vishnu A, Vaidyanathan K, Panda DK (2007) High performance distributed lock management services using network-based remote atomic operations. In: CCGRID, pp 583–590
18. Narravula S, Mamidala A, Vishnu A, Santhanaraman G, Panda DK (2007) High performance MPI over iWARP: early experiences. In: International conference on parallel processing
19. IBM BlueGene Team (2008) Overview of the IBM Blue Gene/P project. *IBM J Res Dev* 52(1/2):199–220
20. Feng W, Warren M, Weigle E (2002) The bladed beowulf: A cost-effective alternative to traditional beowulfs. In: IEEE international conference on cluster computing, p 245
21. Cameron KW, Ge R, Feng X (2005) High-performance, power-aware distributed computing for scientific applications. *Computer* 38(11):40–47
22. Rountree B, Lowenthal DK, Funk S, Freeh VW, de Supinski BR, Schulz M (2007) Bounding energy consumption in large-scale mpi programs. In: SC '07: Proceedings of the ACM/IEEE conference on supercomputing. ACM, New York, pp 1–9
23. Feng X, Ge R, Cameron KW (2005) Power and energy profiling of scientific applications on distributed systems. In: IPDPS '05: Proceedings of the 19th IEEE international parallel and distributed processing symposium (IPDPS'05)—papers. IEEE Computer Society, Washington, p 34
24. Hsu C-H, Kremer U, Hsiao M (2001) Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors. In: ISLPED '01: Proceedings of the international symposium on low power electronics and design. ACM, New York, pp 275–278
25. Burd TD, Brodersen RW (2000) Design issues for dynamic voltage scaling. In: ISLPED '00: Proceedings of the 2000 international symposium on low power electronics and design. ACM, New York, pp 9–14
26. Benini L, de Micheli G (2000) System-level power optimization: techniques and tools. *ACM Trans Des Autom Electron Syst* 5(2):115–192
27. Vishnu A, Koop MJ, Moody A, Mamidala AR, Narravula S, Panda DK (2007) Hot-spot avoidance with multi-pathing over InfiniBand: an MPI perspective. In: Cluster computing and grid, pp 479–486
28. LBNL (2003) Data Center Energy Benchmarking Case Study: Data Center Facility 5
29. IBM (2007) PowerExecutive
30. Bailey AM (2002) Accelerated strategic computing initiative (asci): Driving the need for the terascale simulation facility (tsf). In: Energy 2002 workshop and exposition. IEEE Computer Society, Los Alamitos
31. Ye W, Vijaykrishnan N, Kandemir M, Irwin MJ (2000) The design and use of simple power: A cycle-accurate energy estimation tool, pp 340–345
32. Brooks D, Tiwari V, Martonosi M (2000) Wattch: a framework for architectural-level power analysis and optimizations. In: ISCA '00: proceedings of the 27th annual international symposium on computer architecture. ACM, New York, pp 83–94
33. Zedlewski J, Sobti S, Garg N, Zheng F, Krishnamurthy A, Wang R (2003) Modeling hard-disk power consumption. In: FAST '03: proceedings of the 2nd USENIX conference on file and storage technologies. USENIX Association, Berkeley, pp 217–230
34. Helmbold DP, Long DDE, Sherrod B (1996) A dynamic disk spin-down technique for mobile computing. In: MobiCom '96: Proceedings of the 2nd annual international conference on mobile computing and networking. ACM, New York, pp 130–142

35. Douglis F, Krishnan P, Bershada BN (1995) Adaptive disk spin-down policies for mobile computers. In: MLICS '95: Proceedings of the 2nd symposium on mobile and location-independent computing. USENIX Association, Berkeley, pp 121–137
36. Ge R, Feng X, Song S, Chang H-C, Li D, Cameron KW (2009) Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Trans Parallel Distrib Syst* 99:658–671
37. Moore J, Chase J, Ranganathan P, Sharma R (2005) Making scheduling “cool”: temperature-aware workload placement in data centers. In: ATEC '05: Proceedings of the annual conference on USENIX annual technical conference, USENIX Association, Berkeley, p 5
38. Xinping H-SW, Wang HS, Zhu X, Peh LS, Malik S (2002) Orion: A power-performance simulator for interconnection networks, pp 294–305
39. Luszczek PR, Bailey DH, Dongarra JJ, Kepner J, Lucas RF, Rabenseifner R, Takahashi D (2006) The hpc challenge (hpc) benchmark suite. In: SC '06: Proceedings of the 2006 ACM/IEEE conference on supercomputing. ACM, New York, p 213
40. Song S, Ge R, Feng X, Cameron KW (2009) Energy profiling and analysis of the hpc challenge benchmarks. *Int J High Perform Comput Appl* 23(3):265–276
41. Kandalla SSK, Mancini EP, Panda DK (2010) Designing power-aware collective communication algorithms for InfiniBand clusters. Technical Report, June 2010
42. Freeh VW, Lowenthal DK, Pan F, Kappiah N, Springer R, Rountree BL, Femal ME (2007) Analyzing the energy-time trade-off in high-performance computing applications. *IEEE Trans Parallel Distrib Syst* 18(6):835–848
43. Freeh VW, Pan F, Kappiah N, Lowenthal DK, Springer R (2005) Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In: IPDPS '05: Proceedings of the 19th IEEE international parallel and distributed processing symposium (IPDPS'05)—papers. IEEE Computer Society, Washington, DC, p 4.1
44. Curtis-Maury M, Shah A, Blagojevic F, Nikolopoulos DS, de Supinski BR, Schulz M (2008) Prediction models for multi-dimensional power-performance optimization on many cores. In: PACT '08: Proceedings of the 17th international conference on parallel architectures and compilation techniques. ACM, New York, pp 250–259
45. NAS (2010) NAS Parallel Benchmark
46. Ge R, Feng X, Feng W-C, Cameron KW (2007) Cpu miser: A performance-directed, run-time system for power-aware clusters. In: ICPP '07: Proceedings of the 2007 international conference on parallel processing. IEEE Computer Society, Washington, DC, p 18
47. Zamani R, Afsahi A, Qian Y, Hamacher C (2007) A feasibility analysis of power-awareness and energy minimization in modern interconnects for high-performance computing. In: CLUSTER '07: Proceedings of the 2007 IEEE international conference on cluster computing. IEEE Computer Society, Washington, DC, pp 118–128
48. Liu J, Poff D, Abali B (2009) Evaluating high performance communication: a power perspective. In: ICS '09: Proceedings of the 23rd international conference on supercomputing. ACM, New York, pp 326–337
49. Kendall RA, Aprà E, Bernholdt DE, Bylaska EJ, Dupuis M, Fann GI, Harrison RJ, Ju J, Nichols JA, Nieplocha J, Straatsma TP, Windus TL, Wong AT (2000) High performance computational chemistry: an overview of NWChem, a distributed parallel application. *Comput Phys Commun* 128(1–2):260–283
50. Subsurface Transport over Multiple Phases. STOMP. <http://stomp.pnl.gov/>

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.