Designing Energy Efficient Communication Runtime Systems for Data Centric Programming Models

Abhinav Vishnu [#1], Shuaiwen Song [#2], Andres Marquez [#1],
Kevin Barker [#1], Darren Kerbyson [#1], Kirk Cameron [#2] and Pavan Balaji [#3]

[#1] High Peformance Computing Group,
Pacific Northwest National Lab
902 Battelle Blvd, Richland, WA

[#2] Scalable Performance Lab,
Department of Computer Science,
Virginia Polytechnic Institute,
Blacksburg, VA

[#3] Mathematics and Computer Science Division,
Argonne National Lab, Argonne, IL

***Abstract*—The insatiable demand of high performance computing is being driven by the most computationally intensive applications such as computational chemistry, climate modeling, nuclear physics, etc. The last couple of decades have observed a tremendous rise in supercomputers with architectures ranging from traditional clusters to system-on-a-chip in order to achieve the petaflop computing barrier. However, with advent of petaflop-plus computing, we have ushered in an era where power efficient system software stack is imperative for execution on exascale systems and beyond. At the same time, computationally intensive applications are exploring programming models beyond traditional message passing, as a combination of Partitioned Global Address Space (PGAS) languages and libraries, providing one-sided communication paradigm with put, get and accumulate primitives. To support the PGAS models, it is critical to design power efficient and high performance one-sided communication runtime systems.**

In this paper, we design and implement PASCoL, a high performance power aware one-sided communication library using Aggregate Remote Memory Copy Interface (ARMCI), the communication runtime system of Global Arrays. For various communication primitives provided by ARMCI, we study the impact of Dynamic Voltage/Frequency Scaling (DVFS) and a combination of interrupt (blocking)/polling based mechanisms provided by most modern interconnects. We implement our design and evaluate it with synthetic benchmarks using an InfiniBand cluster. Our results indicate that PASCoL can achieve significant reduction in energy consumed per byte transfer without additional penalty for various one-sided communication primitives and various message sizes and data transfer patterns.

I. Introduction

The last couple of decades have observed a tremendous rise in the design and development of highly scalable applications for a broad spectrum of scientific domains - computational chemistry, weather and ocean modeling, national security to life sciences. The computational requirements of these applications have provided significant momentum to rise of petaflop-plus computing [1]. However, as we move forward to the next step of exascale computing, energy consumptions of systems is expected to be a significant hindrance in naively increasing the computational power by another three orders of magnitude. For example, the U.S. Department of Energy estimates that in order to be able to sustain an exaflop machine, its power consumption cannot be more than 10-fold that of current petaflop machines. That is, we need to achieve a 1000-fold increase in performance, while allowing the power consumption to increase by only 10-fold [2].

Different computational domains have different power efficiency characteristics. For instance, many nuclear physics and computational fluid dynamics applications rely on tightly coupled models based on MPI [3], [4], where processes in a system compute and synchronize with each other every few time steps. Such domains make power efficiency through frequency/voltage scaling of individual processors tricky because of their tightly coupled nature. Some researchers have looked at power efficiency techniques in these domains [5]. On the other hand, application domains such as computational chemistry rely on Partitioned Global Address Space (PGAS) models such as Global Arrays [6] that decouple data management from process synchronization - thus, each process can asynchronously get, put or update data in a globally shared address space.

The primary focus of this paper is to identify the character-

istics of PGAS based applications - specifically in the computational chemistry domain - that provide us with unique capabilities for power and energy efficiency. Hence, we design and implement a "P"ower "A"ware one-"S"ided "Co"mmunication "L"ibrary (PASCoL) based on Aggregate Remote Memory Copy Interface (ARMCI) [7], which is one of the most popular runtime libraries used in various PGAS models including Global Arrays [6] and Co-Array Fortran.

Our implementation is evaluated using synthetic communication benchmarks, which show that we can reduce the overall energy consumption per byte data transfer significantly for various communication primitives and data transfer patterns without incurring significant overhead. The proposed work will be available in an open source manner with ARMCI in upcoming releases.

The rest of the paper is organized as follows. In section II, we present the approaches for power conservation. In section III, we present an overview of computational chemistry with Global Arrays [6] and ARMCI [7]. In section IV, we present the PASCoL design. In section V, we present the performance evaluation of PASCoL using micro-benchmarks and benchmarks. We present the related work in section VI. We conclude and present our future directions in section VII.

II. 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 whole supercomputer. For instance, IBM Blue Gene series [8] and Green Destiny [9] 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 [8]). For saving system power more efficiently and flexibly, Power Modes [10] technique using integrated power-aware components has been provided for fine-grained control for high performance systems, such as low-power setting for network card, spinning down for 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 that performance decreases during the mode switch 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 [11], [12], [13], [14], [15] 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. DVFS can be used to scale CPU frequency down during certain phases like communication to save power where computation is not intensive. It is also possible to apply DVFS to memory modules [16] and network connections [17] but they are not common approaches. PASCoL combines DVFS methodology and improved communication mechanisms to achieve power-aware one-sided communication for various communication patterns.

III. Computational Chemistry with Global Arrays and ARMCI

A vast majority of the computational chemistry applications including the most popular NWChem [18] application are based on the Global Arrays (GA) [6] framework. The GA programming model provides an efficient and portable "shared-memory" programming interface for distributed-memory computers. Each process in a Multiple Instruction Multiple Data (MIMD) parallel program can asynchronously access logical blocks of physically distributed dense multi-dimensional arrays, without need for explicit co-operation by other processes. Unlike other shared-memory environments, the GA model exposes to the programmer the distributed memory characteristics of the high performance computers and acknowledges that access to a remote portion of the shared data is slower than to the local portion. The locality information for the shared data is available, and a direct access to the local portions of shared data is provided. GA internally uses ARMCI [7], as the communication runtime system.

The ARMCI [7] communication runtime system provides a general-purpose, efficient, and widely portable remote memory access (RMA) operations (a.k.a. one-sided communication) optimized for contiguous and non-contiguous (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) [19], Quadrics [20], Giganet (VIA) and InfiniBand (using OpenFabrics and Mellanox Verbs API) [21]. It is also available for leadership class machines including Cray XT4/XT5 and BlueGene/P [22].

While each ARMCI [7] process is an Single Program Multiple Data (SPMD) process, we differentiate in the terminology between processes on the same node to facilitate the implementation of one-sided communication primitives. The process with the lowest rank in the node is called *master* and the rest of the processes on a node are called *clients*. The master process creates a thread, referred to as the *data server*. All processes contribute a part of their memory to a shared-memory pool on each node. The data server exposes

this shared-memory region to other processes in the system as a globally visible address space.

The basic communication model of ARMCI is based on communication between the clients and the data server - there is no direct client-to-client communication. Many modern networks such as InfiniBand, Myrinet, Blue Gene/P and Cray XT provide direct hardware based one sided communication for contiguous messages. In such cases, the client can directly read/write data from the address space exposed by the data server using network hardware primitives and without disturbing the data server at all. However, when atomic updates to the data or non-contiguous communication is required, many network hardware implementations do not provide full hardware support to achieve that. In such cases, the data server needs to actively receive data from the client process and manipulate it as required. **In other words, depending on the communication pattern, the data server might or might not need to be active.**

Another important characteristic of computational chemistry applications such as NWChem is the granularity in which they access data. Since GA explicitly exposes the NUMA model of the distributed memory, most scalable modules within NWChem access data in a coarse-grained nature. That is, they either fetch large amounts of data for computing, or pipeline data with other computation. Thus, these applications are typically not small-message latency bound, but rather rely on asynchronous communication and bulk data transfer capabilities. This characteristic becomes important when seen in the light of network behavior. Specifically, most networks require applications to explicitly initiate communication and then wait for it to complete. The application can wait for completion of the communication in one of two ways - polling or event/interrupt based. In the polling approach, the application would keep querying the network to see if the communication has completed and thus is heavy on CPU usage, but very good for small-message latency. In the event/interrupt based approach, the application asks the network to wake it up when the communication completes and goes to sleep; this approach is typically not very heavy on CPU usage, but not very good for small-message latency, though comparable to the polling based approach for larger messages. **Given that most computational chemistry applications rely on coarse-grained data access we might be able to expect an interrupt-driven approach to give equivalent performance as a polling-based approach, while providing more opportunities for energy efficiency.**

IV. Approaches for Energy Optimization

In this section, we present the PASCoL design. We begin with a discussion on mechanisms for energy efficiency and their applicability to one-sided communication protocols. This is followed by discussion on efficient communication protocols for contiguous and non-contiguous data transfer. We also briefly present the optimizations for data server, which is involved during multiple phases of various communication primitives, as discussed in the background section.

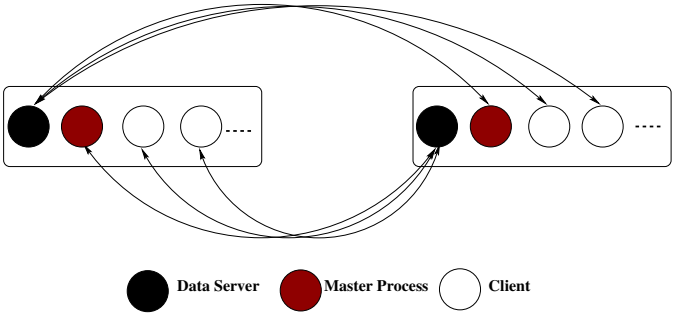

Fig. 1. Communication Structure in ARMCI

A. Mechanisms for Energy Efficiency

There are multiple complimentary mechanisms available for designing energy efficient one-sided communication protocols:

- Interrupt based execution allows multiple stages of communication protocols to transition using event driven mechanisms. As an alternative to polling, which is typically used in high performance computing applications, this method allows much better CPU utilization, particularly, if the overhead of interrupts can be controlled.
- DVFS provides energy efficiency by reducing the frequency and/or voltage. Different communication protocols require varying CPU utilization during different stages of communication primitives. Energy conservation can be achieved using DVFS, if the overhead of voltage and frequency scaling can be controlled.

Keeping the mechanisms discussed above in mind, we design and implement efficient one-sided communication protocols. While our framework allows voltage scaling, due to the limitations of our setup, we are able to use frequency scaling only. For the rest of the article, we would use dynamic frequency scaling (DFS).

B. Contiguous Data Transfer

As presented in section III, most modern network provide RDMA and send/receive primitives to allow zero copy data transfer. The primary restriction for RDMA is that the source and destination buffer should be contiguous. Most networks also require that the buffer be pinned (so that the network transfers do not result in page fault(s)) and registered (for access privileges). One-sided communication libraries including ARMCI [7], pin and register the user buffers and exchange the access keys (for InfiniBand [21]) during the memory allocation phase. As a result, during the data transfer phase, the contiguous data transfer typically results in using network descriptor(s). Hence, for these protocols, the primary overhead is posting a descriptor for data transfer request. Since most of the modern networks, including InfiniBand, use Programmed I/O (PIO) for small messages, the CPU needs to be involved during the phase of posting the descriptor.

For contiguous data transfer with small messages, we use DFS (scale down) after the descriptor has been posted. The frequency scale up is performed when the data transfer has

been completed. For large messages, we use DFS at the point of data transfer request, since doorbell mechanism is used when posting the descriptor. The doorbell mechanism does not require CPU involvement.

For blocking data transfer, the interrupt based mechanism is used immediately after the descriptor has been posted. With non-blocking data transfer, the mechanism is invoked when the application requests *Wait* on the previously posted descriptors. This ensures that the non-blocking data transfer does not incur any additional overhead.

C. Non-Contiguous Data Transfer

ARMCI provides primitives for uniformly non-contiguous (strided) and non-uniform non-contiguous (vector) data transfer patterns. There are multiple protocols for communication with these patterns. The simplest method is to post individual descriptors for each data segment. However, it results in significant overhead with inefficient utilization of network concurrency. Networks like InfiniBand [21] also provide mechanisms for list based data transfer, in which a list of scatter/gather entries can be provided. However, this mechanism results in higher context memory utilization [23].

Hence, ARMCI primarily uses copy based communication protocol for non-contiguous data transfer. In this approach, the user data is copied to a preregistered buffer and the descriptor(s) are posted. Since the copy phase of the protocol is CPU intensive, we use DFS (scale down)/interrupts or both after the desciptor(s) have been posted.

D. Data Server

As presented in section III, the processes communicate to the data server for different communication protocols, as needed. The copy based protocols require communication to the data server, while contiguous data transfer (such as put/get) uses Remote Direct Memory Access (RDMA), bypassing the data server completely. The data server might need to be active, depending on whether different processes are in computation/communication phases. They may also not need to communicate to the data server at all. Given that most computational chemistry applications rely on the coarse grained data accesses, the data server may need to be active infrequently.

To optimize the data server for energy efficiency, we use a combination of polling/interrupts and DFS. We scale up the frequency when an interrupt is received (in case of interrupt driven execution) and scale down the frequency before blocking on an event from the network. In case of polling, we scale down the frequency before polling on a completion queue and scale up the frequency when a message is received on the completion queue.

V. Power & Performance Evaluation of PASCoL

In this section, we evaluate the performance of PASCoL with synthetic communication benchmarks designed and implemented at the ARMCI [7] layer. We begin with description of Experimental Testbed. This is followed by a discussion on the communication patterns of different benchmarks and associated metrics of interest. We present the evaluation of these benchmarks using an InfiniBand [21] cluster. The description of the experimental testbed is presented below.

A. Experimental Testbed

For the purpose of experimental evaluation, we use a cluster - NW-ICE, hosted at Pacific Northwest National Lab. The NW-ICE cluster has 192 compute nodes, connected with DDR InfiniBand network adapters and switches. Each NW-ICE node is a dual socket quad core with 2.33 GHz frequency. Each node has 16GBytes of main memory with each core having a 32KB cache size.

With DVFS, NW-ICE allows frequencies of 2.33 GHz and 1.9 GHz. By default all processes start at 2.33 GHz. The dynamic scale down reduces the frequency to 1.9 GHz and scale up increases the frequency to 2.33 GHz. The interface for changing the frequencies is through a memory resident file system. We expect much higher improvements in performance more frequency settings.

B. Performance Evaluation Methodology

In this section, we present the performance of PASCoL with multiple synthetic communication benchmarks. The benchmarks perform communication to all processes with displaced ring communication [23]. The displaced ring communication is defined with MPI two-sided semantics. The one-sided version uses Put, Get and Accumulate primitives provided by ARMCI [7] to achieve a similar objective.

However, unlike MPI All-to-all personalized exchange, this benchmark does not necessary maintain stepwise data exchange between various communication phases. Hence, the communication pattern achieves a similar objective as MPI All-to-all personalized exchange, however, the characteristics during different phases is different altogether. The pseudo code for the benchmark using put primitive is presented here:

$start\ timer$
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
$end\ timer$

The benchmarks for Accumulate and Put Strided are implemented by replacing the appropriate put calls with respective primitive calls. For Get primitive, the fence is not needed, since return of the Get primitive indicates the data completion as well.

1) Evaluation Metrics: As presented above, we have considered primarily communication centric benchmarks for performance evaluation of PASCoL. To capture the efficacy of our proposed approaches, we measure the total energy consumed for each message size of the benchmark. We normalize the energy consumed with the overall message size, to provide an

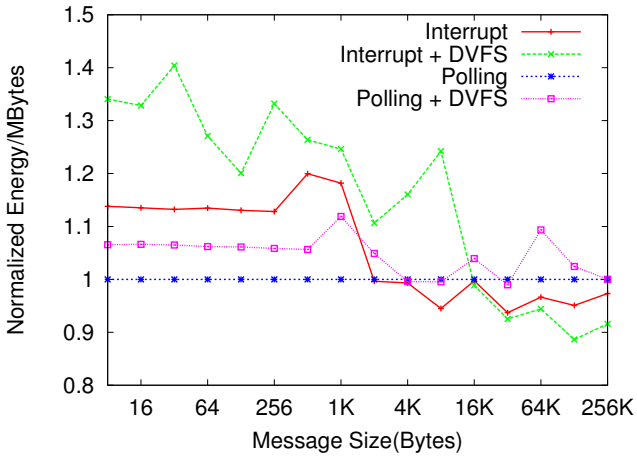

Fig. 2. ARMCI Put Performance, Energy/MBytes

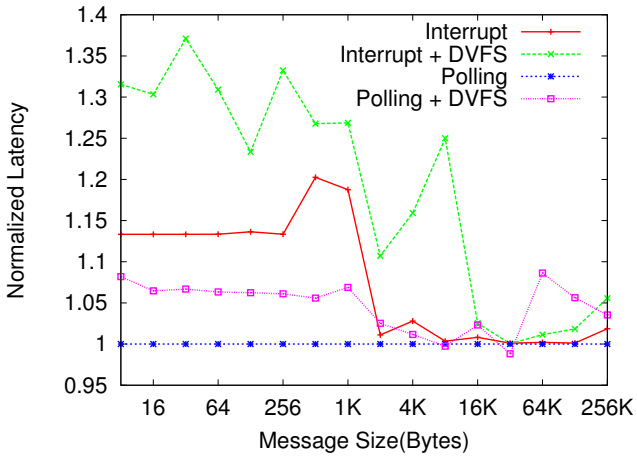

Fig. 3. ARMCI Put Performance, Latency

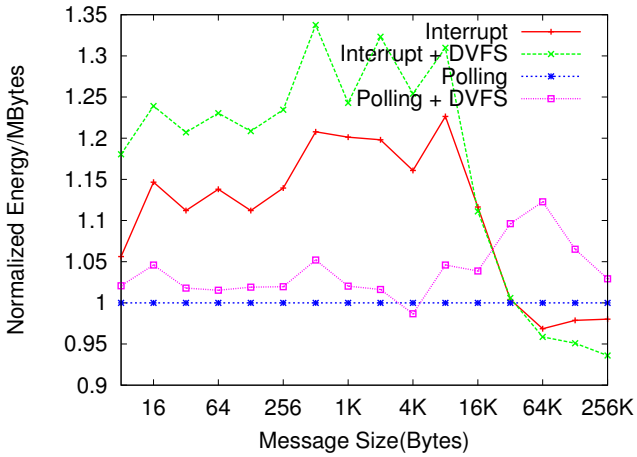

Fig. 4. ARMCI Get Performance, Energy/MBytes

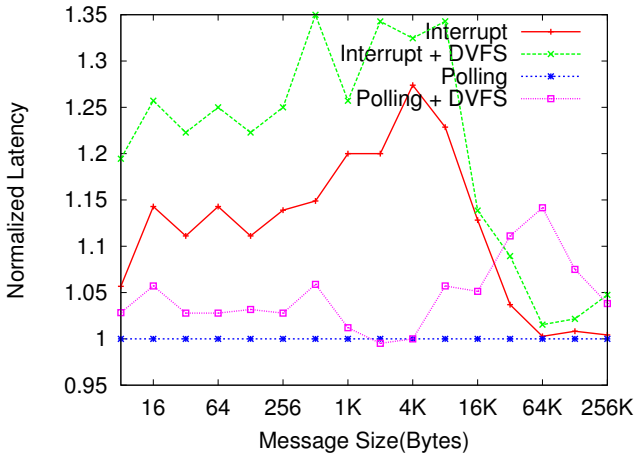

Fig. 5. ARMCI Get Performance, Latency

asymptotic value beyond which one approach performs better than the other. This metric is referred to as *Energy/Mbytes* for the rest of the section. Since the overall execution time is critical to an end user, we also measure these values and compare the approaches. For each of these metrics, we normalize the values to the default case of polling.

C. Results

Figures 2 and 3 show the results for displaced ring communication with ARMCI Put benchmark on 64 processes using normalized values with the polling implementation. We compare the performance of polling with interrupts only, interrupts with DVFS and polling with DVFS. For small messages, we observe that polling has the best Energy/Mbytes and latency. Since small messages are highly sensitive to latency, using either interrupt(s) or DVFS results in significant overhead. As a result, polling outperforms other implementations for the small messages. We observe spikes for different implementations, because a small change in timing for DVFS/interrupts can significantly impact the overall overhead observed by the implementation. Polling with DVFS performs worse than polling only, while interrupts only performs worse than polling with DVFS. The overhead incurred by interrupts with DVFS is the highest among all approaches for small messages.

With increasing message size, the latency for multiple approaches converges significantly (less than 5% difference). At 16KBytes message size, we observe that the latency for all schemes (with a slight exception to polling with DVFS), converges. However, we observe that Energy/Mbytes consumption for the interrupts with DVFS improves significantly compared to the other schemes. Overall we observe an 8% performance improvement in Energy/Mbytes using interrupts with DVFS compared the default polling case, while an improvement of 5% is observed compared to the interrupts only scheme. For large messages, the overhead incurred by interrupts is amortized by the overall time of data transfer. However, with the increased data transfer time, the time for completion increases providing higher energy efficiency for interrupts and DVFS approach. We observe that a threshold of 16KBytes can be used for using interrupts with DVFS without significant performance degradation.

Figures 4 and 5 show the performance of Energy/Mbytes and latency comparison using ARMCI get primitive. We observe trends similar to ARMCI put communication primitive presented above. For small messages, interrupt with DVFS performs the worst with energy consumed per Mbyte and latency, since the overhead incurred per message transfer is significant for interrupts and DVFS. The interrupts only approach performs slightly better, with polling and DVFS approach slightly worse than the polling only approach for small messages in terms of Energy/Mbytes and observed latency. We observe that the latency for different communication primitives converges at 32KBytes. We observe an improvement of 7% at 256Kbytes, however we observe a slight degradation in performance compared to the polling at this point. We are still investigating this observation.

Figures 6 and 7 show the performance for ARMCI Accumulate primitive, with Energy/Mbytes and latency, respectively. The accumulate communication primitive involves data server at the remote node to perform local accumulates. We observe that around 32KBytes, the latency for different approaches converges, while the Energy/Mbytes performance improves for larger messages. Overall, a performance improvement of 7% is observed using interrupts with DVFS in comparison to the polling scheme.

Figures 8 and 9 show the performance of ARMCI Put strided communication primitive with increasing message size for normalized Energy/Mbytes and latency, respectively. As presented in the design section, the strided communication primitives are implemented using a copy based approach. Hence, the communication protocol has a copy and data transfer phase at the client and the data server side, each. We observe that the threshold for using interrupt with DVFS is around 64Kbytes, which is higher in comparison to the contiguous data transfer. We observe an improvement of 7% for large messages using interrupts with DVFS in comparison to the polling scheme. With interrupts only scheme, the overall improvement in Energy/Mbytes is 3%, while the normalized latency converges.

For each of communication primitives discussed above, we observe that a reduction in Energy/Mbyte can be observed for large messages, without incurring significant overhead on latency. As presented in section III, most computational chemistry applications rely on coarse grained data accesses. Hence, there is a potential for computational chemistry applications to benefit from the proposed design choices with PASCoL.

VI. 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. Above mentioned methodologies focus on measuring the aggregate power consumption of entire system or building level power [24] through proprietary hardware [25], power panels, or empirical estimations by rules-of-thumb [26]. Many studies, including both simulations and empirical analysis, have also explored evaluation of individual system components such as processor [27], [28], memory [27], disk [29], [30], [31], motherboard [32], CPU and system fan control [33] and interconnection networks [34]. 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 [32] to evaluate energy efficiency and power-aware techniques for parallel applications. Song et al., [35] have used PowerPack to study the power characteristics of multiple suites in HPCC benchmark [36] at a high granularity. 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. 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 [32], [35]. Many researchers have discussed the tradeoff between performance and energy consumption for scientific applications such as NAS Parallel Benchmark [12], [37], [38] [39], [40]. They have pointed out the importance of efficient detection of communication regions during runtime [39], [37]. In [39], researchers also combine DVFS with concurrency throttling technique on multi-core 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 [41] monitors system performance counters to estimate workload in order to predict the proper frequency for next time interval on a single node. Researchers in [42] and [43] 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 [44]. 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.

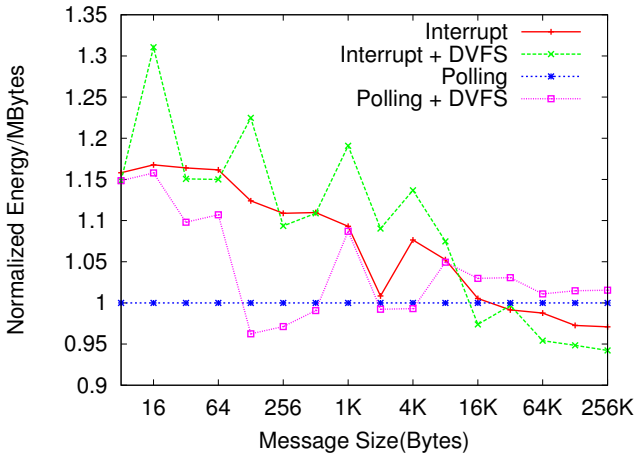

Fig. 6. ARMCI Accumulate Performance, Energy/MBytes

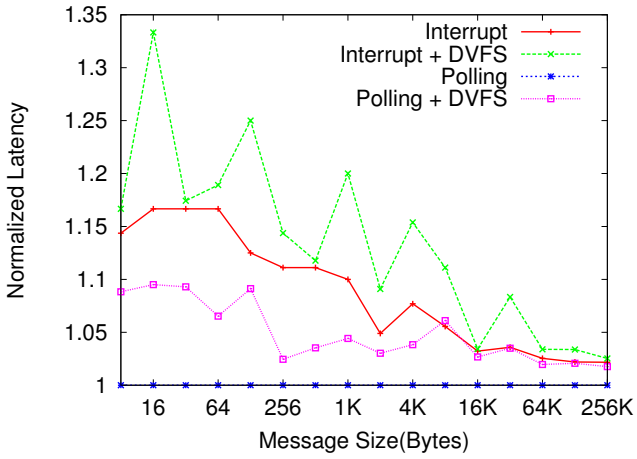

Fig. 7. ARMCI Accumulate Performance, Latency

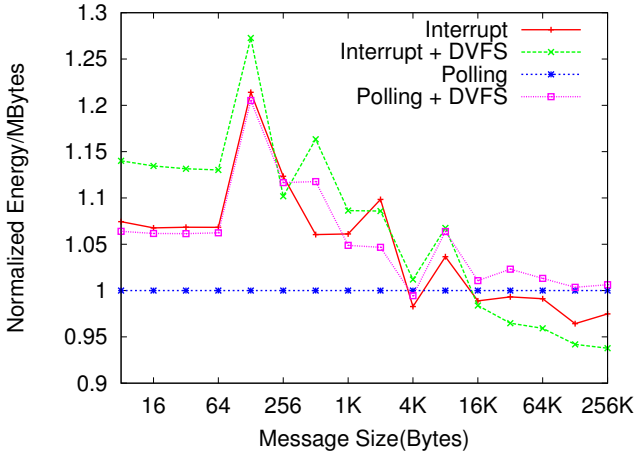

Fig. 8. ARMCI Put Strided Performance, Energy/MBytes

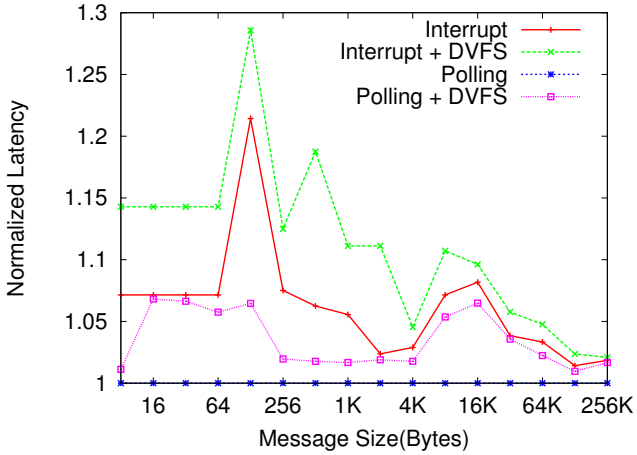

Fig. 9. ARMCI Put Strided Performance, Latency

VII. Conclusions and Future Work

The primary focus of this paper is to identify the characteristics of PGAS based applications - specifically in the computational chemistry domain - that provide us with unique capabilities for power and energy efficiency. To supplicate this, we have designed and implemented a power-aware one-sided library based on Aggregate Remote Memory Copy Interface (ARMCI) [7], which is one of the most popular runtime libraries used in various PGAS models including Global Arrays and Co-Array Fortran. Our implementation is also evaluated using synthetic communication benchmarks, which show that we can reduce the overall energy consumption per byte data transfer significantly for various communication primitives and data transfer patterns without incurring significant overhead. The proposed work will be available in an open source manner with ARMCI in upcoming releases.

We plan to continue design and development of power aware 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 [18] and Subsurface Transport over Multiple Phases (STOMP) [45].

References

[1] "TOP500 Supercomputing Sites," http://www.top500.org.
[2] "Crosscutting Technologies for Computing at the Exascale," in *http://extremecomputing.labworks.org*, 2010.
[3] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard," *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996.
[4] A. Geist, W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. L. Lusk, W. Saphir, T. Skjellum, and M. Snir, "MPI-2: Extending the message-passing interface," in *Euro-Par, Vol. I*, 1996, pp. 128–135.
[5] C.-H. Hsu and W. chun Feng, "A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters," in *International Conference on Cluster Computing*, 2005, pp. 1–10.
[6] J. Nieplocha, R. J. Harrison, and R. J. Littlefield, "Global Arrays: A Nonuniform Memory Access Programming Model for High-Performance Computers," *Journal of Supercomputing*, vol. 10, no. 2, pp. 169–189, 1996.
[7] J. Nieplocha and B. Carpenter, "ARMCI: A Portable Remote Memory Copy Library for Distributed Array Libraries and Compiler Run-Time Systems," in *Lecture Notes in Computer Science*. Springer-Verlag, 1999, pp. 533–546.
[8] I. B. Team, "Overview of the IBM Blue Gene/P project," *IBM J. Res. Dev.*, vol. 52, no. 1/2, pp. 199–220, 2008.

[9] W. Feng, M. Warren, and E. Weigle, "The bladed beowulf: A cost-effective alternative to traditional beowulfs," *Cluster Computing, IEEE International Conference on*, vol. 0, p. 245, 2002.

[10] K. W. Cameron, R. Ge, and X. Feng, "High-performance, power-aware distributed computing for scientific applications," *Computer*, vol. 38, no. 11, pp. 40–47, 2005.

[11] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz, "Bounding energy consumption in large-scale mpi programs," in *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 2007, pp. 1–9.

[12] X. Feng, R. Ge, and K. W. Cameron, "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*. Washington, DC, USA: IEEE Computer Society, 2005, p. 34.

[13] C.-H. Hsu, U. Kremer, and M. Hsiao, "Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors," in *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2001, pp. 275–278.

[14] T. D. Burd and R. W. Brodersen, "Design issues for dynamic voltage scaling," in *ISLPED '00: Proceedings of the 2000 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2000, pp. 9–14.

[15] L. Benini and G. d. Micheli, "System-level power optimization: techniques and tools," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 2, pp. 115–192, 2000.

[16] H. B. Fradj, C. Belleudy, and M. Auguin, "Multi-bank main memory architecture with dynamic voltage frequency scaling for system energy optimization," *Digital Systems Design, Euromicro Symposium on*, vol. 0, pp. 89–96, 2006.

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

[18] R. A. Kendall, E. Aprà, D. E. Bernholdt, E. J. Bylaska, M. Dupuis, G. I. Fann, R. J. Harrison, J. Ju, J. A. Nichols, J. Nieplocha, T. P. Straatsma, T. L. Windus, and A. T. Wong, "High Performance Computational Chemistry: An Overview of NWChem, A Distributed Parallel Application," *Computer Physics Communications*, vol. 128, no. 1-2, pp. 260–283, June 2000.

[19] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, and W. Su, "Myrinet: A Gigabit-per-second Local Area Network," *IEEE Micro*, vol. 15, no. 1, pp. 29–36, February 1995.

[20] F. Petrini, W. Feng, A. Hoisie, S. Coll, and E. Frachtenberg, "The Quadrics Network: High-Performance Clustering Technology," *IEEE Micro*, vol. 22, no. 1, pp. 46–57, 2002.

[21] InfiniBand Trade Association, "InfiniBand Architecture Specification, Release 1.2," October 2004.

[22] S. Kumar, G. Dozsa, G. Almasi, P. Heidelberger, D. Chen, M. E. Giampapa, M. Blocksome, A. Faraj, J. Parker, J. Ratterman, B. Smith, and C. J. Archer, "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*, 2008, pp. 94–103.

[23] A. Vishnu, M. J. Koop, A. Moody, A. R. Mamidala, S. Narravula, and D. K. Panda, "Hot-Spot Avoidance With Multi-Pathing Over InfiniBand: An MPI Perspective," in *Cluster Computing and Grid*, 2007, pp. 479–486.

[24] LBNL, "Data Center Energy Benchmarking Case Study: Data Center Facility 5," 2003.

[25] IBM, "PowerExecutive," 2007. [Online]. Available: http://www-03.ibm.com/systems/management/director/extensions/powerexec.html.

[26] A. Bailey, "Accelerated strategic computing initiative (asci): Driving the need for the terascale simulation facility(tsf)," in *Energy 2002 Workshop and Exposition*. IEEE Computer Society, 2002.

[27] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: A cycle-accurate energy estimation tool," 2000, pp. 340–345.

[28] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA '00: Proceedings of the 27th annual international symposium on Computer architecture*. New York, NY, USA: ACM, 2000, pp. 83–94.

[29] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, "Modeling hard-disk power consumption," in *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2003, pp. 217–230.

[30] D. P. Helmbold, D. D. E. Long, and B. Sherrod, "A dynamic disk spin-down technique for mobile computing," in *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1996, pp. 130–142.

[31] F. Douglis, P. Krishnan, and B. N. Bershad, "Adaptive disk spin-down policies for mobile computers," in *MLICS '95: Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*. Berkeley, CA, USA: USENIX Association, 1995, pp. 121–137.

[32] 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," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. RapidPosts, pp. 658–671, 2009.

[33] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": temperature-aware workload placement in data centers," in *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 5–5.

[34] H.-S. W. Xinping, H. sheng Wang, X. Zhu, L. shiuan Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," 2002, pp. 294–305.

[35] S. Song, R. Ge, X. Feng, and K. W. Cameron, "Energy profiling and analysis of the hpc challenge benchmarks," *Int. J. High Perform. Comput. Appl.*, vol. 23, no. 3, pp. 265–276, 2009.

[36] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The hpc challenge (hpcc) benchmark suite," in *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 2006, p. 213.

[37] 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 Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 835–848, 2007.

[38] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer, "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*. Washington, DC, USA: IEEE Computer Society, 2005, p. 4.1.

[39] M. Curtis-Maury, A. Shah, F. Blagojevic, D. S. Nikolopoulos, B. R. de Supinski, and M. Schulz, "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*. New York, NY, USA: ACM, 2008, pp. 250–259.

[40] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS Parallel Benchmarks," in *The International Journal of Supercomputer Applications*, no. 3, 1991, pp. 63–73. [Online]. Available: citeseer.ist.psu.edu/bailey95nas.html

[41] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, "Cpu miser: A performance-directed, run-time system for power-aware clusters," in *ICPP '07: Proceedings of the 2007 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 2007, p. 18.

[42] R. Zamani, A. Afsahi, Y. Qian, and C. Hamacher, "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*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 118–128.

[43] S. S. Krishna Kandalla, Emilio P. Mancini and D. K. Panda, "Designing Power-Aware Collective Communication Algorithms for InfiniBand Clusters," *Technical Report*, June 2010.

[44] J. Liu, D. Poff, and B. Abali, "Evaluating high performance communication: a power perspective," in *ICS '09: Proceedings of the 23rd international conference on Supercomputing*. New York, NY, USA: ACM, 2009, pp. 326–337.

[45] Subsurface Transport over Multiple Phases, "STOMP," http://stomp.pnl.gov/.