Power Consumption Due to Data Movement in Distributed Programming Models

Siddhartha Jana[1], Oscar Hernandez[2], Stephen Poole[2], and Barbara Chapman[1]

[1] Computer Science department,
University of Houston,
Houston, Texas
{sidjana,chapman}@cs.uh.edu
[2] Computer Science and Mathematics Division
Oak Ridge National Laboratory,
Oak Ridge, Tennessee
{oscar,spoole}@ornl.gov

Abstract. The amount of energy consumed due to data movement poses a serious challenge when implementing and using distributed programming models. Message-passing models like MPI provide the user with explicit interfaces to initiate data-transfers among distributed processes. In this work, we establish the notion that from a programmer's standpoint, design decisions like the size of the data-payload to be transferred and the number of explicit MPI calls to service such transfers have a direct impact on the power signatures of communication kernels. Upon closer look, we additionally observe that the choice of the transport layer (along with the associated interconnect) and the design of the data transfer protocol, both contribute to these signatures. This paper presents a fine-grained study on the impact of the power and energy consumption due to data movement in distributed programming models. We hope that results discussed in this work would motivate application and system programmers to include energy consumption as one of the important design factors while targeting HPC systems.

1 Introduction and Related Work

One of the primary challenges on the pathway to Exascale Computing is the 20MW power consumption envelope established by the U.S. Department of Energy's Exascale Initiative Steering Committee [11]. The direct outcome of this has been a rising concern about the energy and power consumption of large-scale applications that rely on various communication libraries for efficient data movement in distributed systems. As part of this work, we establish the notion that the factors responsible for the performance of such libraries, also govern the power-profiles of such applications. Fig. 1 lists many such factors throughout the hardware and software stack.

This work is an extension of our previous experience of studying the impact of one-sided communication in PGAS models (OpenSHMEM) [7]. We had learned that managing small-sized data transfers on RDMA-capable networks are more energy efficent than handling large bulk transfers. In this paper, we present empirical evidence highlighting the contribution of design factors within

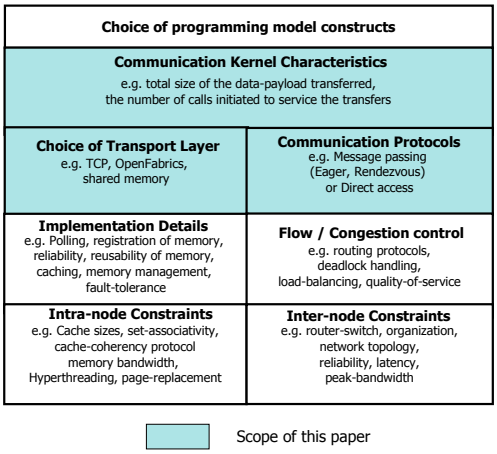


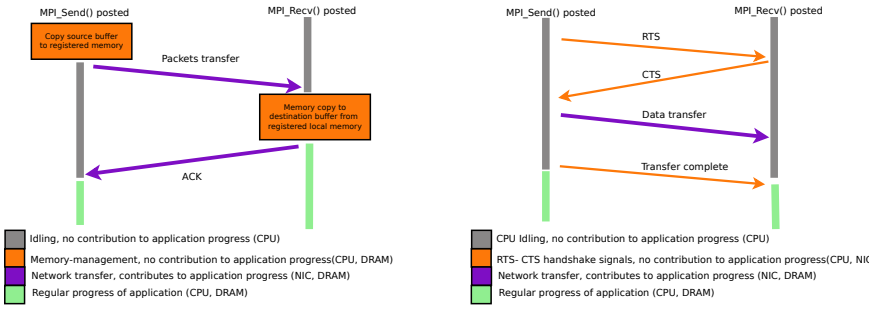
Fig. 1: Factors impacting the energy and power consumption across the hardware and software stack

the software stack to the power consumption by the underlying system. Our takeaway from this study is that the protocols used to implement such interfaces, play a significant role in impacting its power-cost. In addition, since the design of communication libraries are tuned to specific interconnect solutions, the choice of the transport layer adopted for servicing data transfers plays an equally significant role.

In Section 2, we discuss the impact of the above factors on the behavior of two-sided communication interfaces within MPI, the de facto standard for distributed memory model. This is an extension of past work on analyzing the impact of data-transfer characteristics on one-sided communication interfaces [7]. We discuss the characteristics of our testbed and our experimental methodology in Section 3. This is followed by a description of our observations of the impact on power consumption by CPU cores and the DRAM while relying on Ethernet (via traditional TCP) and Infiniband (via OFED or OpenFabrics Enterprise Distribution [1]) fabrics (Section 4). All of these are discussed with respect to the implementation of two basic message-passing schemes - the Eager and Rendezvous protocols. Finally in Section 4, we summarize our findings by discussing the total power efficiency achievable for each of the above configurations. We hope this work motivates the practice of taking power-metrics into consideration while designing middleware solutions for Exascale-era machines.

2 Factors affecting Power and Energy profile of remote data transfers

Two-sided data-transfer in distributed-memory models like MPI, sockets, etc. require the active participation of both the sender and the receiver of the data. The impact on the achievable latency and bandwidth of such transfers depend on the design of the transport layer (and the associated interconnect) and the data transfer protocol. As part of this work, we learned that the impact of these factors on the energy metrics is very important.



(a) Eager Protocol (b) Rendezvous Protocol

Fig. 2: Sequence Diagrams for Eager and Rendezvous protocols

2.1 Choice of transport layer and the associated interconnect

If the target platform relies on an OS-based TCP protocol for servicing data transfers, CPU cores undergo multiple switches between user and supervisor operating modes. In addition, relying on Ethernet-based fabric has the potential of degrading the achievable efficiency both in performance and energy consumption (as discussed later). To avoid this, a communication library may exploit kernel-bypass mechanisms and RDMA-based capabilities of the OFED stack on top of modern interconnects like InfiniBand, etc.

2.2 Design of data-transfer protocols

Data transfers within message-passing libraries are based on two well-established paradigms - the eager and rendezvous protocols. The primary phases involved in these protocols are depicted in the line diagrams in Fig. 2.

Rendezvous protocols incorporate RTS-CTS[3] handshaking to ensure that the sender waits for an explicit request from the receiver before servicing the actual transfer. Such an exchange ensures that the receiver's buffer is ready for being overwritten with the incoming payload. This method has proven to be beneficial for large bulk transfers since the overhead of the handshaking operation gets eclipsed by the gain in the throughput of the end-to-end data movement [2]. For small message sizes, however the additional round trip proves expensive.

Eager protocols help mitigate the above overhead by reducing the time and energy spent by the sender waiting for the receiver to post the destination buffer address. The sender may choose to start transfering its data to a pre-allocated buffer without waiting for the receiver to send a CTS signal. This is easily facilitated by an underlying interconnect solution that supports RDMA-based transfers. Once the receiver calls MPI_Recv(), it can copy-out the data from this pre-allocated buffer. Not surprisingly, the impact of latency of such techniques is bounded by the costs of memory registration and the additional in-memory copies both at the sender's and the receiver's end.

2.3 Power Versus Latency

Energy efficient communication depends on a number of factors listed in Fig. 1. Through this text, we hope to establish the difference between optimizing for

[3] Request-To-Send / Clear-To-Send two-sided handshake signal

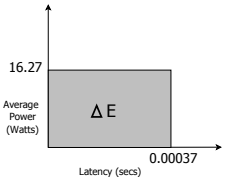
Table 1: Test-Platform characteristics

RAPL monitored Node		PowerPack monitored SystemG Node	
Processor	Intel Xeon CPU E5-2670	Processor	Intel Xeon CPU E5462
Microarchitecture	Intel's Sandy Bridge	Microarchitecture	Intel's Sandy Bridge
Operating Frequency	2.6 GHz	Operating Frequency	2.8 GHz
Maximum Thermal Design Power (TDP)	115 Watts	Maximum Thermal Design Power (TDP)	80 Watts
Hyperthreading support	Disabled	Hyperthreading support	Disabled
Infiniband card	Mellanox MT26428, fw-ver:2.7.0	Infiniband card	Mellanox MT26428, fw-ver:2.5.9
Linux kernel version	2.6.32 x86_64	Linux kernel version	2.6.32 x86_64
Compiler	gcc version 4.4.6	Compiler	gcc version 4.4.4
Compiler flags used	-O3	Compiler flags used	-O3
Energy Sampling rate	1ms	Power Sampling rate	10ms

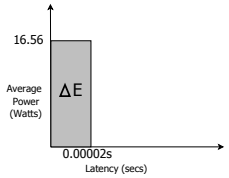
energy versus power. It must be noted that one doesnt always have to sacrifice the lowest possible latency to achieve energy-efficiency. Consider the plots shown in

Fig. 3.

The plots (a) and (b) depict the average power consumed by CPU cores (Y-axis) and the corresponding latency (X-axis) incurred while transferring a 32KB payload across the network (MPI Send-Recv over InfiniBand). If this payload is divided into 64 fragments, the energy consumption by the CPU cores is about 6 mJ and the transfer takes about 370 μs to complete. The average power consumption during this transfer is about 16.21 Watts (Fig. 3a). If instead, we chose to split this payload into only 2 fragments (16KB each), the energy consumption drops to 0.33 mJ (by 94.5%) and latency to 20 μs (by 94.6%). However, this comes at the cost of a rise in power consumption to 16.565W, i.e. an increment by 1.8% (Fig. 3b). Thus, despite the higher power consumption, choosing the latter option enables the CPU cores to service the transfer using lesser energy.



(a) Using 64 fragments to transfer data. Energy=6mJ



(b) Using 2 fragments to transfer data. Energy=0.33mJ

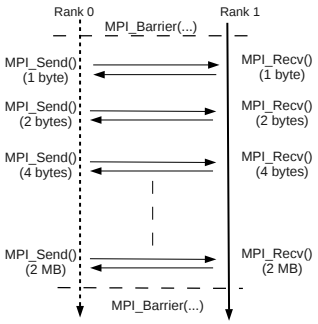
Fig. 3: Power Versus Latency. Use of a 32KB data payload transferred using MPI_Send() over InfiniBand

3 Experimental Setup

Our study was aimed at performing a fine-grained analysis of the impact on different components of a distributed system, namely, the cores, the socket, the motherboard, the memory unit, and the entire compute-node as a whole. The experiments were performed for two different implementations of MPI - Open MPI [4] and MVAPICH2 [9]. We observed similar behavior between the two MPI implementations. Due to space constraints, we discuss the impact of only Open MPI's implementation of data transfers on two major components [5] that contribute to the total power consumption of a system, viz. the CPU cores

Table 2: Synthetic microbenchmark used for evaluating energy and power consumption by varying the total size of data payload and the number of fragments

Sequence Diagram



Code snippet

```
/*MAX_WRK_SIZE: is the maximum data
                payload to be transferred
                within a communication kernel
*/
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
for(j=1;j<=MAX_WRK_SIZE;j*=2)
{
 for(frag_cnt=1; frag_cnt<=j; frag_cnt*=2)
 {
  bytes_per_msg = j / frag_cnt;
  MPI_Barrier();
  // START monitoring
  for (it=0; it<frag_cnt; it++)
   if (rank==0)
    MPI_Send(..., bytes_per_msg, MPI_BYTE, 1, ..);
   else
    MPI_Recv(..., bytes_per_msg, MPI_BYTE, 0, ..);
  // STOP monitoring
 }
}
```

and the memory. While the network card forms an important component of a distributed system, past study indicates that its impact on the total power consumption by a system is about 1% [3]. We therefore omit any further discussion on the impact of NIC from the rest of the text.

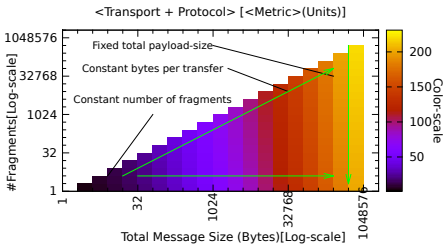


Fig. 4: Layout of the plots in this paper

Note on interpreting the graphical plots We briefly discuss the method of interpreting the plots presented in the following sections. Each plot illustrates empirical results in terms of an energy metric. It corresponds to a specific transport layer and a communication protocol.

The coordinate axes (log-scale) correspond to two controllable factors that define a communication phase in an application - the total size of data transferred during that phase (X-axis) and the number of explicit MPI-calls (Y-axis) used to transfer that payload. Throughout this text, we refer to the latter as the count of *fragments*[4]. The shade of a point in this coordinate space indicates the value of monitored metric that is represented by the color-scale to the right of each plot.

Microbenchmark used for evaluating the impact of data transfers The pseudo-code and the sequence diagram of the synthetic microbenchmark used to study the impact of the explicit data transfers are presented in Table 2. It must be noted that the type of transfers being evaluated use traditional MPI blocking two-sided point-to-point interfaces.

Test Platform for Monitoring Energy and Power Consumption We incorporated two different power-monitoring schemes:

[4] It must be noted that each fragment may further be divided into smaller chunks by the underlying layers based on the middleware design, NIC hardware contraints, etc.

– *Intel's Running Average Power Limiting (RAPL):* The energy consumption by the CPU cores were monitored using the RAPL interface [6] which are exposed as model specific registers on SandyBridge platforms. These were read using VampirTrace [8] via PAPI [10]
– *PowerPack on SystemG:* The power consumption by the memory unit, was measured using the PowerPack 4.0 framework on the SystemG cluster at Virginia Tech. The power consumption is measured directly using the four VDC pins that supply power to the module [5].

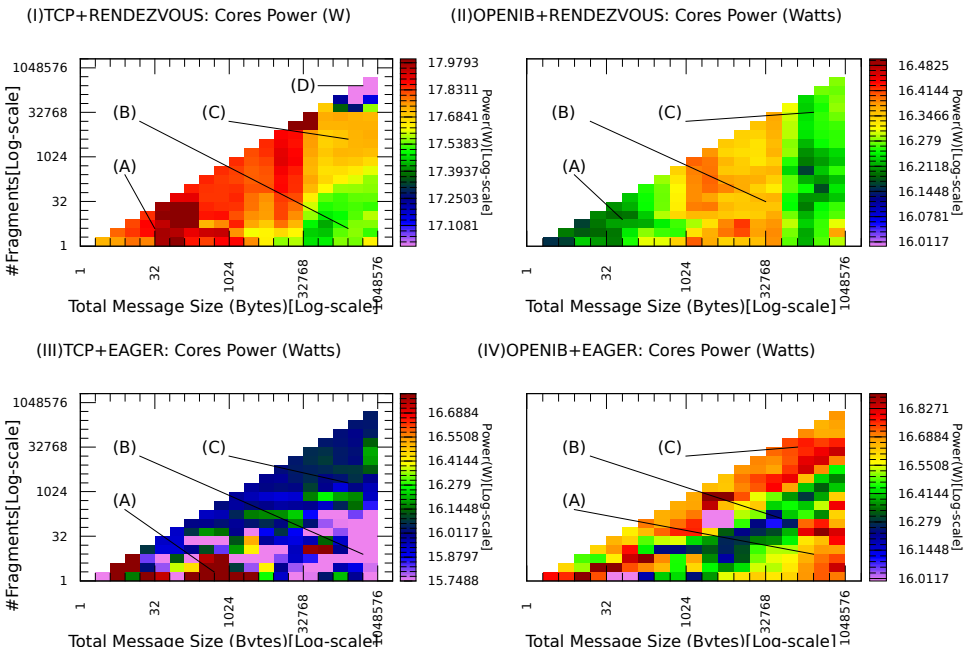
4 Empirical Observation and Analysis

In this section, we present our observations of the impact on the energy and power consumption by the CPU cores and memory due to the factors discussed in the previous section.

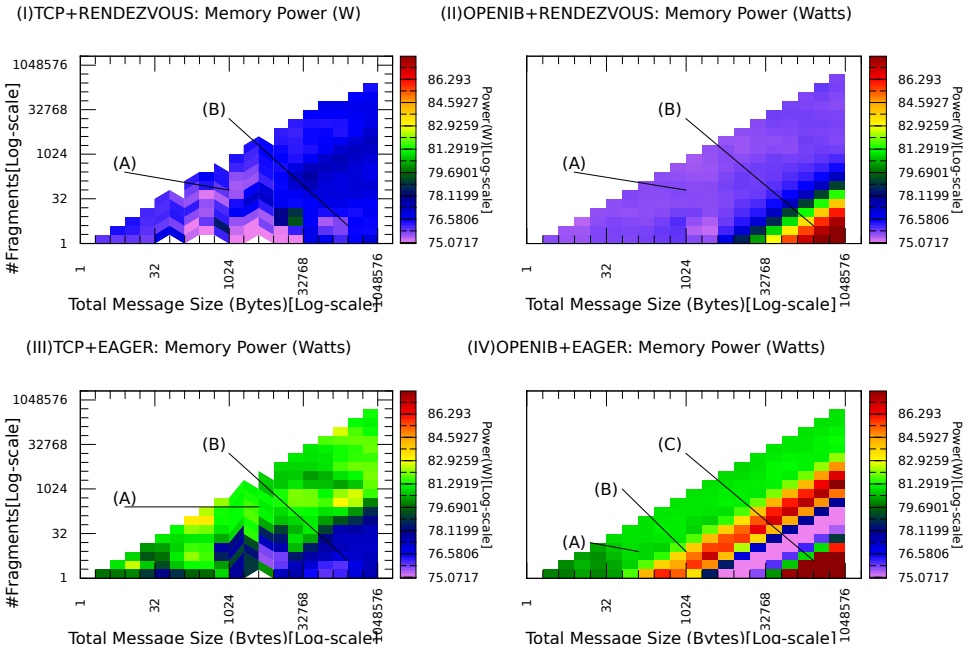
4.1 Using TCP over Ethernet

Using Rendezvous protocol Consider the power consumption by the CPU cores servicing the sender process (Fig. 5a(I)). While handling small data payloads ($< 1KB$) the CPU cores suffer a high power cost (region A). The reason for this may be attributed to the very low latency of the operation, the frequent context switches between the operating modes (see Section 2) and the high overhead incurred during the handshake operations. This cost reduces for large bulk transfers ($> 32KB$) due to a rise in the latency of the data transfer and a drop in the rate of active participation by the CPU cores (region B). Dividing such bulk buffers into smaller fragments again leads to a rise in the cost (region-C). However, this rise in power-cost in limited due to high latency that arises with heavy fragmentation. Due to this, the inverse relation between the increase in latency and the drop in the average power consumption can be observed in region-D. On the receiver's end (Fig. 6a(I)), the power consumed primarily depends on the size of the data being transferred (regions A,B). It must also be noted that the passive participation by the receiver (when compared to the sender), leads to a lower range of power-consumption (15.5-16.2W as compared to 17-18W). From the memory unit's perspective (Fig. 5b(I)) the power cost incurred by the sender process while servicing small transfers (region A) is lesser than that while servicing large transfers (region B).

Using Eager protocol Switching from a rendezvous protocol to an eager protocol definitely reduces the operating power-cost incurred by the CPU cores while servicing large data transfers by the sender and the receiver processes (Figs. 5a(III), 6a(III)(regions A-B)). The negative impact of fragmentation can be observed in terms of the rise in the power-cost incurred by the memory modules, both by the sender as well as the receiver (Figs. 5b(III), 6b(III)(regions A-B)). Implementing an eager protocol using a non-RDMA based fabric like Ethernet leads to a significant rise in power-consumption at the receiver's end (Fig. 6b(III)(region A)). A rise in the number of bytes transferred per fragment leads to a rise in the energy consumed by the memory. However, we see from region B in Figs. 5b(III), 6b(III) that the power consumption by the memory

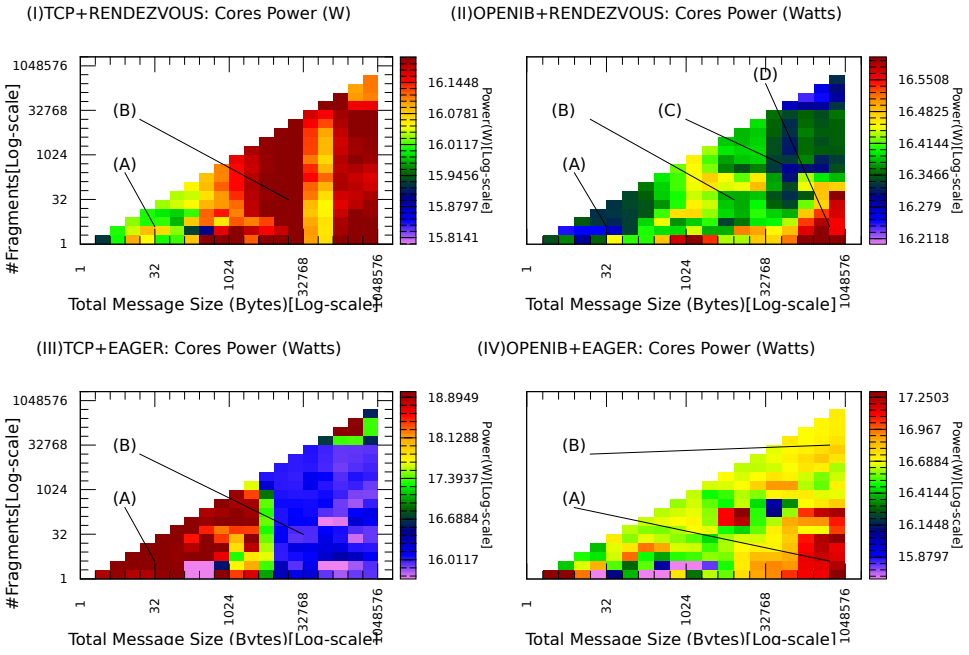


(a) Power consumed by the CPU cores

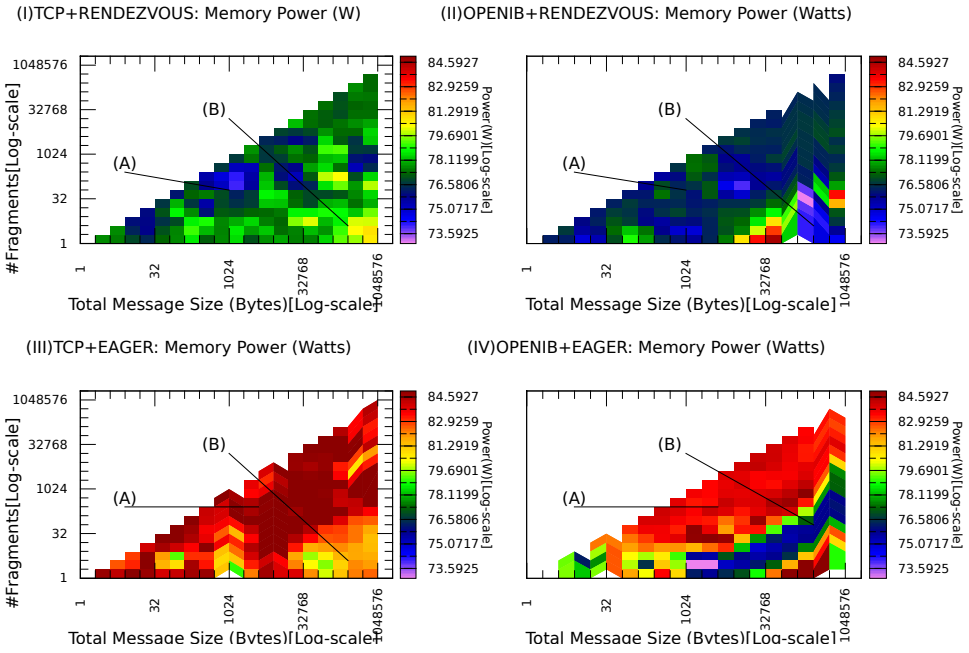


(b) Power consumed by the DRAM

Fig. 5: Power consumed by the CPU cores and the DRAM while servicing remote data transfers by the sender process



(a) Power consumed by the CPU cores



(b) Power consumed by the DRAM

Fig. 6: Power consumed by the CPU cores and the DRAM while servicing remote data transfers by the receiver process

module drops. This can be attributed to the rise in the latency in completion of the transfer of the entire data payload.

4.2 Using OpenIB/OFED stack over InfiniBand

Using Rendezvous protocol: At first glance, Figs. 5a(II), 6a(II) (regions A,B,C), depict that the power consumed by the CPU cores is dependent on the total size of the data payload and not so much on the degree of fragmentation. However, one must take into account that using the rendezvous protocol over the OFED stack leads to a combination of two different types of overhead. The first is the power-penalty of using either memory-pinning or local memcpy operations (as explained in the next subsection). The second is the overhead due to the handshaking operations (as explained in the previous sub-sections).

With regards to the power consumed by the memory at the sender's side, the cost increases monotonically with a rise in the size per fragment. As discussed in the following bullet point below, using the OFED stack is accompanied with the power-penalty of either memory-pinning or local memcpy operations. This cost varies with the number of bytes transferred with each fragment (Fig.5b(III)).

Using Eager protocol: Parallel diagonally-colored bands in Figs. 5a(IV), 5b(IV), 6a(IV), 6b(IV) show that the power consumed by the cores and the memory unit, both depend on the number of bytes transferred within each fragment. As discussed before, either the memory space containing these fragments are dynamically pinned-down (registered) with the NIC or its contents are copied over to some pre-registered buffer. The performance penalty of dynamic registration of small buffers is expensive. Thus, a runtime implementation would typically perform a local copy of the contents into a pre-registered buffer. Our experience shows that the power cost of this memory copy increases with rise in the fragment size (i.e. bytes/fragment). This can be observed in region-C. As the size of each fragment increases, an implementation would typically start dynamically registering user buffers with the NIC. Either way, this keeps the CPU cores active. It is during this inflection point that we observe a slight drop in the cores power consumption. Further increase in the size of fragment again leads to a rise in this cost (region A).

Complementary to the CPU power consumption, the power-cost incurred by the memory rises with the size per fragment (region A). It too hits a cool spot (region B) and then rises up monotonically with a rise in the achievable bandwidth on the NIC.

4.3 Summary: Achievable Energy-efficiency during data transfers:

To study the net impact of the choice of the communication protocols and the transport layer, we evaluated the power efficiency using a metric tuned towards communication-intensive kernels. The energy-efficiency of a compute-intensive application kernel is given by the total number of machine/floating-point operations per second per watt of power consumed (MOPS/Watt or FLOPS/Watt). To evaluate the cost of data transfer operations, we use a similar metric - the net bandwidth achievable per watt of power consumed by the participating processes; in other words - the number of bytes that can be transferred across the

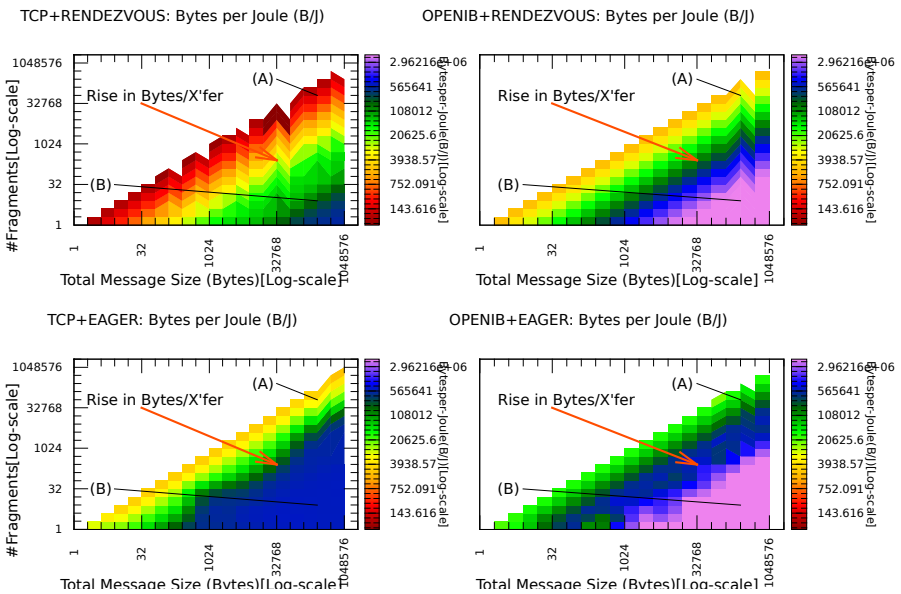


Fig. 7: A summary of the total bytes transferred per Joule of energy consumed by the sender and the receiver while participating in remote data transfers.

network for each joule of energy consumed by the sender and receiver. For a point-to-point communication model like MPI, this may be represented by the equation below:

$$\frac{Bw}{P_{net}} = \frac{Bw}{(P_{s,cpu}+P_{s,mem}+P_{r,cpu}+P_{r,mem})} = \frac{B_{payload}}{\Delta E_s+\Delta E_r} \quad (\frac{Bytes}{Joule}) \tag{1}$$

The symbols used in this equation are listed in Table 3. The net impact on this metric is discussed in Fig: 7. The primary observations are:

Table 3: Symbols in Eqn.1

Symbols	Metric
Bw	Achievable bandwidth (bytes/sec)
P_{net}	Net average power consumed (W)
$B_{payload}$	Total number of bytes transmitted
ΔE_s	Energy consumption by sender (J)
ΔE_r	Energy consumption by receiver (J)
$P_{s,cpu}$	Cores power consumption at sender (W)
$P_{r,cpu}$	Cores power consumption at receiver (W)
$P_{s,mem}$	Memory power consumption at sender (W)
$P_{r,mem}$	Memory power consumption at receiver (W)

- The net bandwidth achievable using an interconnect directly impacts the maximum value of energy-efficiency. Thus the peak bytes transmitted per joule is an order of a magnitude higher when using the OpenIB over InfiniBand as compared to TCP over Ethernet. Moreover, irrespective of the type of transport adopted, energy-efficient communication can be achieved using an eager-based protocol.
- *Impact of number of bytes packed per transfer:*
  - In the figure, the arrow points towards the direction of the increase in the number of bytes transferred per call. For TCP+Rendezvous config-

uration (Fig. 7(I)), we see that the peak energy efficiency during a data transfer (about 0.565MB/Joule) may be attained only when the total message size per transfer is higher than 128KB. With the TCP+Eager protocol however, this peak is attained for message sizes beyond 1KB in size.

- The highest power efficiency among all the configurations is achievable while using an eager-based protocol over the OpenIB stack - A maximum of 3MB of data is transferred for every joule of energy consumed.

5 Conclusion

Data movement across large-scale systems has the potential of impacting not only the performance of distributed programming models, but also the power-signatures. In this paper, we established the notion that the choice of the transport layer and the design of communication protocols play a significant role in terms of the energy and power consumption. The empirical results discussed in this paper highlighted the behavior of this impact on the CPU cores and the memory. It was observed that the power consumption by CPU cores and the memory bandwidth is not only impacted by the latency of the remote transfers, but also the memory bandwidth between the CPU cores and the memory.

While using traditional TCP over Ethernet, energy savings can be obtained by choosing an eager-based protocol over a rendevous-based one. While using an eager protocol, an efficiency of upto 600bytes/joule may be obtained. Despite these savings, it must be noted that mapping an eager protocol over a non-RDMA based fabric leads to high power consumption by the memory. While using an RDMA-capable network like InfiniBand, the use of eager-based protocol lends itself naturally to the semantics of the transport layer (OpenFabrics OFED, in our case).

Irrespective of the type of transport and protocol, higher efficiency (bytes transferred per joule) can be achieved by aggregating user buffers into contiguous larger fragments before servicing the transfer. In addition, the net bandwidth achievable during a transfer impacts this efficiency. We hope that results of energy-efficiency as well as a detailed study of the impact on the various sub-components of the system would motivate the design of "power-aware" middleware for use with HPC applications.

As the next step, we plan to extend this study to evaluate the impact on large-scale multi-node systems. It is equally essential to study the contribution of communication kernels to the energy profiles of large scale real world HPC applications.

6 Acknowledgments

This work is supported by the U.S. Dept. of Defense via the Extreme Scale Systems Center and used resources at the Oak Ridge Leadership Computing Facility, which is supported by the Office of Science (U.S. Dept. of Energy) under Contract No. DE-AC05-00OR22725. Thanks are due to the support teams of

VampirTrace from the Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH, Technische Universität Dresden, Germany), PowerPack from the Scalable Performance Lab (Virginia Tech, U.S.A.) and PAPI from the Innovative Research Lab (University of Tennessee at Knoxville, U.S.A.).

References

1. OpenFabrics Alliance: The Case for Open Source - RDMA (August 2011), https://www.openfabrics.org/index.php/ofa-documents/presentations/doc_download/228-the-case-for-open-source-rdma-.html
2. Barrett, B.: OpenMPI Data Transfer. http://www.open-mpi.org/video/internals/Sandia_BrianBarrett-1up.pdf (December 2012), detailed overview of the OpenMPI data transfer system.
3. Feng, X., Ge, R., Cameron, K.: Power and energy profiling of scientific applications on distributed systems. In: Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International. pp. 34–34 (April 2005)
4. Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Dongarra, J.J., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L., Woodall, T.S.: Open MPI: Goals, concept, and design of a next generation MPI implementation. In: Proceedings, 11th European PVM/MPI Users' Group Meeting. pp. 97–104. Budapest, Hungary (September 2004)
5. Ge, R., Feng, X., Song, S., Chang, H.C., Li, D., Cameron, K.W.: Powerpack: Energy profiling and analysis of high-performance systems and applications. IEEE Transactions on Parallel and Distributed Systems 21(5), 658–671 (2010)
6. Intel Corporation: Intel(R) 64 and IA-32 Architectures Software Developer's Manual Vol. 3B: System Programming Guide, Part-2 (February 2014), http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.html
7. Jana, S., Hernandez, O., Poole, S., Hsu, C.H., Chapman, B.: Analyzing the energy and power consumption of remote memory accesses in the openshmem model. In: Poole, S., Hernandez, O., Shamis, P. (eds.) OpenSHMEM and Related Technologies. Experiences, Implementations, and Tools, Lecture Notes in Computer Science, vol. 8356, pp. 59–73. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-05215-1_5
8. Knüpfer, A., Brunst, H., Doleschal, J., Jurenz, M., Lieber, M., Mickler, H., Mller, M., Nagel, W.: The vampir performance analysis tool-set. In: Resch, M., Keller, R., Himmler, V., Krammer, B., Schulz, A. (eds.) Tools for High Performance Computing, pp. 139–155. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-68564-7_9
9. Liu, J., Wu, J., Panda, D.: High performance rdma-based mpi implementation over infiniband. International Journal of Parallel Programming 32(3), 167–198 (2004), http://dx.doi.org/10.1023/B%3AIJPP.0000029272.69895.c1
10. Mucci, P.J., Browne, S., Deane, C., Ho, G.: PAPI: A portable interface to hardware performance counters. In: In Proceedings of the Department of Defense HPCMP Users Group Conference. pp. 7–10 (1999)
11. Shalf, J., Dosanjh, S., Morrison, J.: Exascale Computing Technology Challenges pp. 1–25 (2011)