AC922 Data Movement for CORAL

Roberts, Steve
IBM
Cognitive Systems
Austin, TX
robers@us.ibm.com

Ramanna, Pradeep
IBM
Cognitive Systems
Austin, TX
pramann2@in.ibm.com

Walthour, John
IBM
Cognitive Systems
Austin, TX
jwalthour@us.ibm.com

***Abstract*—Recent publications have considered the challenge of movement in and out of the high bandwidth memory in attempt to maximize GPU utilization and minimize overall application wall time. This paper builds on previous contributions, [5] [17], which simulate software models, advocated optimization, and suggest design considerations. This contribution characterizes the data movement innovations of the AC922 nodes IBM delivered to Oak Ridge National Labs and Lawrence Livermore National Labs as part of the 2014 Collaboration of Oak Ridge, Argonne, and Livermore (CORAL) joint procurement activity. With a single HPC system able to perform up to 200PF of processing with access to 2.5PB of memory, this architecture motivates a careful look at data movement. The AC922 POWER9 system with NVIDIA V100 GPUs have cache line granularity, more than double the bandwidth of PCIe Gen3, low latency interfaces and are interconnected by dual-rail Mellanox CAPI/EDR HCAs. As such, the bandwidth and latency assumptions from previous simulations should be revisited and compared to characterization results on product hardware. Our characterization approach attempts to leverage existing performance approaches, as applicable, to ease comparison and correlation. The results show that by refocusing our attention on the interconnect between processing elements, it is possible to design efficient logically coherent heterogeneous systems.**

***Index Terms*—MPI, RDMA, Exascale, data sharing, on demand paging, heterogeneous system coherence, POWER9, NVLink, CORAL, CAPI, ATS, GPU Direct**

I. Introduction

The data centric compute vision motivates the elimination or minimization of data movement, enabling processing engines to operate as close as possible to application data. While it's easy to see how this vision minimizes communication, compute density optimization comes from pairing CPU and GPU processing elements together with memories that suite the purpose of each. It is highly desirable for programming models to be unencumbered with managing data movement and ultimately programs that do direct data copies are less portable. Workflow analysis of traditional parallel programming models and emerging cognitive workloads illustrated that moving growing data sets dominate computation time either from parallel file systems to main store or from main store to PCIe attached accelerators.

In preparation for Exascale Systems, the CORAL systems are a critical first step to demonstrate architecture scale and the utility of several data movement innovations. The challenge to scale to over 300PF systems coupled with a 24 MW energy envelope led to a dense heterogeneous node design with CPU and GPU processing elements associated with their respective DRAM or high bandwidth memory (HBM2). Each processor element creates a NUMA domain which in total encompasses over $>$ 2PB worth of total memory (see Table I for total capacity).

TABLE I
CORAL Systems Memory Summary

Lab	Nodes	Sockets	DRAM (TB)	GPUs	HBM2 (TB)
ORNL	4607	9216	2,304	27648	432
LLNL	4320	8640	2,160	17280	270

Efficient programming models call for accessing system memory with as little data replication as possible and with low instruction overhead. The MPI community cited several challenges with computing at this scale including effective synchronization scaling with applications that would otherwise benefit from One-Sided Communication and the necessity of effective MPI_Alltoall communication and has considered new constructs to enable hybrid programming [16]. In consideration of these, IBM's CORAL proposal included two principals that focused on data movement:

1) Minimize data motion
2) Enable compute across the system hierarchy

These principals resulted in hardware assisted data movement innovations focused on reducing overhead and evolving programming models towards a single global address. As a result, the IBM AC922 design aimed at efficient RDMA transfers targeting remote HBM2 and efficient coherent memory transfers between the POWER9 CPU and NVIDIA V100 GPU over NVLink 2.0 .

This contribution will be to benchmark data movement innovations staying as close to industry methods and previous contributions as possible. In cases where able, we demonstrate results with the hardware features active and inactive.

II. Related Work

In this section we concentrate on two macro areas: 1) RDMA accesses between GPUs in different NUMA domains and 2) CPU/GPU memory transfers. We reference existing literature for treatment of these topics and then describe pertinent AC922 design features where description helps to explain

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

our characterization approach. In [9], the AC922 architecture is described with focus around the NVLink 2.0 including the processor element configurations and associated bandwidths.

A. RDMA operations between GPUs

The ability to perform RDMA send/recv operations between GPUs in different system nodes without the expense of making intermediate copies to system memory has enabled scaling for GPU workloads. As defined in [12], the features allowing data movement among GPUs and between GPUs and other PCI Express are defined as "GPU Direct". System memory copies and CPU overhead is avoided by copying the data directly to/from from GPU memory. The Ohio State University team behind the OSU benchmarks [4] demonstrate that this savings reduces point-to-point communications times by between 2.5x to 7x in [8] and also demonstrate up to 59% reduction in GPU to GPU latency with "GPU Direct" in [14].

In the AC922 the Mellanox host channel adapter (HCA) is attached directly to the POWER9 processor without going through a PCIe switch. The GPU RDMA transfers, also bypass CPU DRAM eliminating latency bottlenecks using Direct Memory Access. Since GPU memory is used for RDMA, efficient data transfer is achieved by avoid CPU overhead and data replication. RDMA allows HCA DMA engines to saturate IB link bandwidth. Data flow via RDMA cross the NVLink bus to P9 PCIe Host Bridge (PHB) to the HCA. POWER9's NVLink Processing Unit (NPU) bridges the gap between POWER9 internal bus and NVLink 2.0 and also enabled the cache coherency protocol. The POWER9's Memory Management Unit serves the CPU, GPU, and the PHBs supporting Mellanox HCAs.

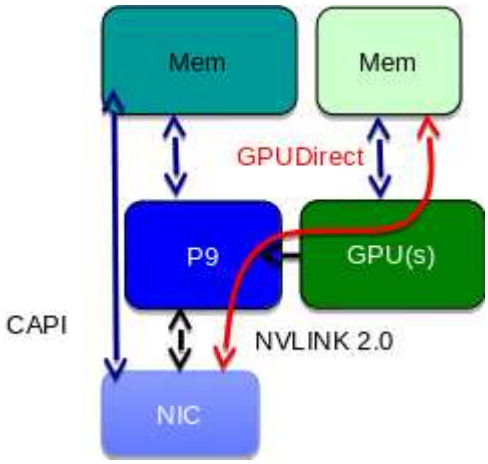

Fig. 1. P9 V100 RDMA Direct

The CORAL system design is interconnected with a dual-rail Infiniband [2] EDR system fabric. The GPU RDMA paths between HCAs travel this fabric and libraries design to optimize transfers through the fabric. For CORAL, IBM leveraged its Blue Gene/Q heritage with PAMI [11] [10]. The PAMI library is able to make sure of Mellanox's On Demand paging as described in Section III-C. This CPU focused work is then extended with the same MPI characterization approach in describing lower latency collective processing in SHARP [7]. From these works we borrow the approach of using point to point communicaiton with MPI_Send and MPI_Recv, MPI_Alltoall, and MPI_allreduce() analysis to characterize latency, but limit the scope to focus on the path within the compute node.

B. CPU/GPU Memory Transfers

Efficient data sharing is cited in [5] [17] as being expensive in terms of performance due to the transfer size of current paging mechanisms and the latency of page fault operations. These works expose a cost 20μs and 50μs for the multiple transfers required to service a page fault. The cost stems from not using 57% of migrated cache blocks due to the access patterns for applications commonly targeted for GPU processing elements. Latency is addressed in [17] by simulating better eviction policies with on demand prefetching and memory oversubscription which are shown to increase the probability of data being local when needed. A different approach lowers transfer latency via a GPU architecture change to enabling GPUs to manage page faults (verse the traditional method of calling back to the runtime) is simulated in [5]. Both of these contributions cited that if fine grained access could be achieved with low latency, the data migration efficiency increases due to less unused transfers and more time to effectively prefetch.

The implementation of cache line transfers between the POWER9 and V100 certainly constitute fine grain sharing. Using vector add kernel, daxpy, the AC922 design team demonstrates 66.2 GB/s bandwidth (>99% efficiency) with GPU resident kernels operating memory owned by the CPU [9]. Also provided in [9] is detailed treatment of the CPU to GPU bandwidths. The peak NVLink 2.0 bandwidth between the CPU and GPUs are listed at 50 GB/s and 75 GB/s bi-directional depending on the 4 or 6 GPU configuration. This bandwidth represents more than double the amount of bandwidth assumed in previous work and the access granularity built in the AC922 equals the lower bound previously assumed for transfer granularity.

III. Background: AC922 Data Movement

The AC922 data movement features come in a few categories:

1) CPU to GPU communications described in [9]
2) Features implemented as part of OpenCAPI 3.0
3) Reductions in logical and physical data paths

Design elements involving CPU and GPU communications are described in Section III-A and Section III-B. In Section III-C we introduce design features for the OpenCAPI 3.0 standard and its application to On Demand Paging and PAMI.

A. Heterogeneous Coherent Memory

The AC922 implements 128B cache line coherency between the POWER9 processor elements and the NVIDIA V100 (Volta) processing elements. This coherency comes with an

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

instruction set of atomic operations and shared coherency protocol. A shared address space is maintained in the operating system that encapsulates the CPU's memory (sysmem) and the GPU's memory (vidmem). These share a simply coherency protocol containing only invalid and owned states. The GPU reads system memory into its L1 cache (software managed) and writes back through its L2 which then updates the system coherency tables via the NVLink Processing Unit (NPU). The GPUs keep track of lines loaned to the CPU in a hardware table so that these can be efficiently pulled back when needed.

B. Low Latency GPU RDMAs

The NPU unit acts as a proxy to the system page table and provides an Address Translation Sevvice (ATS). The NPU accepts requests from the GPU, facilitates translation, creates the request in the POWER9 Memory Management Unit and maintains translations context for outstanding requests. When the request is serviced, the hardware table is updated. An undisclosed hardware algorithm selects the invalidation routine. This mechanism is similar to what is described in [5] but has the ability to deal with GPU faults without interaction with the CUDA runtime.

C. On Demand Paging with CAPI ATS

CAPI attach IO [15] as implemented with the POWER9 ATS provides the following latency features:

1) A path from Infiniband to application memory that eliminates kernel and device driver software overhead (around 500 instructions verse 15000 instructions required for PCIe)
2) The ability to pin translations in the host ERAT (effective to real address translation table) and use an address tag for subsequent references versus an effective address
3) Translate touch to prefetch address translation caches
4) Wake-up host thread, which allows a low latency mechanism in lieu of either interrupts or host processor polling mechanism of memory
5) Posted writes for improved streaming performance

Without CAPI/ATS, when a On Demand Page (ODP) packet arrives the Mellanox HCA will try to fetch the real address from an on-chip translation cache. If found in cache, it will normally process the packet. If not in cache, then it will interrupt the system and drop the packet. The IB driver interrupt handler will use the kernel call get_user_pages to do the translation and move that translation into cache. When the packet is retransmitted, it will have real address available and packet will be processed and acknowledged.

The path is almost the same with CAPI/ATS. However if it has a cache miss, it will use the CAPI address translation services to get the real address. This is targeted to service a page in under 3 $/mu$s, unless there is a page fault (in which case there will also be an interrupt). With the current latency, the packet could be serviced instead of getting dropped.

The impact of this would be seen with a program that generates translation cache misses at the adapter level, but not page faults and there has to be a large enough percentage of these misses to make a performance difference. If the application is linearly marching through memory, there may not be enough translation misses to show a performance difference.

IV. Approach

A. Hardware Coherent Memory

While it would seem logical to consider performance counter equivalents to a miss status handling register (MSHR) and track the misses as the cost of coherency [13], we take a different approach postulating the bottleneck is the number of wires on the NVLink thereby instead measuring the ability to sustain full bandwidth traffic as the measure of successful coherency directory design. In the AC922, this means the NPU would need to have the resources required to keep the NVLink saturated in both directions with coherent memory access as demonstrated by Bordawekar et al [3]. For more coverage of these mechanisms in conjunction with system atomics and CUDA's unified memory, we leverage the approach provided in [5] and employ CHAI [6].

B. Low Latency RDMA

Using OSU benchmarks, which inculcates non-blocking collectives, we intend to achieve overlap between computation and communication. RDMA data transfers between GPUs and other PCIe devices will be initiated thereby hiding CPU to GPU bandwidth and latency bottlenecks. This will demonstrate a significant improvement in MPI send and receive efficiency between GPUs and other remote nodes. With OSU point-to-point workloads we intend to show the data movement in terms of latency using blocking MPI_Send and MPI_Recv implementation and bandwidth measurements which should reflect maximum sustained data transfer rate achieved using non-blocking MPI_Send and MPI_Recv implementations.

C. CAPI ATS On Demand Paging

CAPI ATS and On Demand Paging performance will be demonstrated using MPI application benchmark called AMG (Algebraic MultiGrid Solver). AMG supports a large memory footprint and is a proxy for sparse memory access pattern workloads in CORAL. Running with and without CAPI ODP, we will demonstrate paging effectiveness through better execution times and understand this contributes to scale out performance. Performance improvement for one-sided MPI applications with CAPI ATS with no remote host involvement will be justified using RDMA access support to entire host address space.

V. Evaluation

The experiments are performed on the IBM AC922 (8335-GTW) equipped with a 2 POWER9 22-core processors running at 3.45GHz, 256GB of memory, 6 NVIDIA V100 GPUs with 16GB of memory, and a Mellanox CX5 CAPI/EDR host bus adaptor with 2 physical and 2 virtual links in the shared bifurcated x16 slot. For the CAPI/ATS results in this paper, the minimum level skiboot open firmware is 6.0.4 and the RHEL

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

kernel version greater or equal to 4.14.0-49.9.1. The NVIDIA V100 driver 396.37 driver was used with CUDA 9.2 runtime libraries. Our fabric stack is built on Mellanox MLNX_OFED version 4.3-3.0.9.1.

A. RDMA Performance

The dataset shown in Figure 2 was created with the Infiniband Verbs Performance Tests, *perftest* [1]. It shows comparison RDMA bandwidth for cases Host-to-Device, Host-to-Host, Device-to-Device and Device-to-Host. GPUs are designated as Device in this comparison. For all these cases discussed below, RDMA is performed inter-nodes and confined to single processor socket. These results peak at 11.8 GB/s (limited by single EDR link) whereas the OSU Bandwidth Test results peak at 12.3 GB/s (limited by PCIe Gen4 x8). In our evaluation, we found that both share the same shape, but due to the buffering in the OSU test, tuning at the device driver level is more straight forward with *perftest*. At this bandwidth, the GPUDirect function is 90% the peak of PCI Gen4 bandwidth and higher that what is previously published doing through PCIe-based approach.With OSU benchmarks this increases to 93%.Further improvement is planned by increasing number of DMA Read engines to reduce the GPU memory latency which will produce bandwidth of 13.1 GB/s.

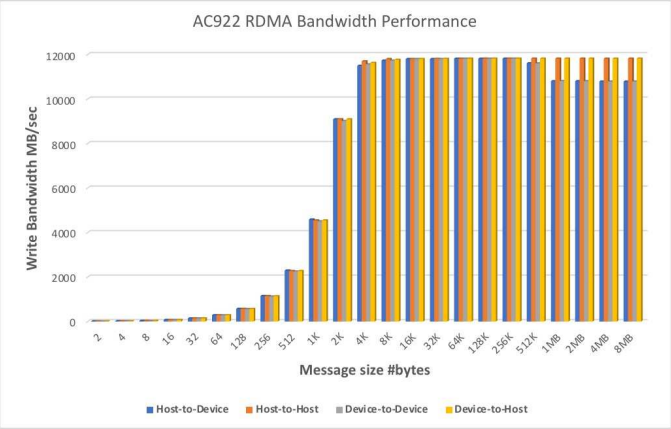

Fig. 2. RDMA Bandwidth Performance

The results of the RDMA based OSU latency tests are shown in Figure 3. For smaller message sizes, GPU to GPU latency is 7-8 μs as compared to Host-to-Host latency which is 2 μs. GPU to GPU latency becomes comparable to Host-to-Host for message size 64KB and above. These results show a that the latency for small page size until 64KB is 17 μs for a GPU initiated transfer to the host. As mentioned above, these results are expected to improve with increase of DMA read engines for GPU memory.

B. CHAI

The CHAI benchmarks are implemented in two versions, the 'D' version uses traditional approaches such as double allocation of buffers, explicit copies between devices, and kernel launch/termination for synchronization [6]. The 'U' version

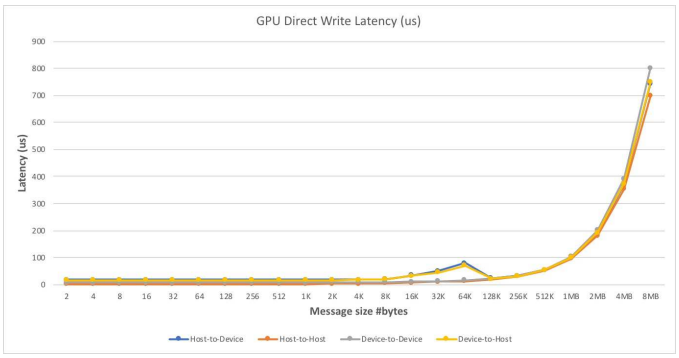

Fig. 3. RDMA Latency Performance

employs unified virtual memory, relies on system coherency and system atomics. With these, a general comparison can be made.

For the evaluation with CHAI, we increased the default number of warm-up runs to 10 and took the average of 100 runs. This was done somewhat arbitrary, but our initial results were subject to OS jitter as many of those processes were on the lower CPUs and these have affinity to GPU0 which was the default target in the test. Note that note all the CHAI tests ran correctly; the benchmark with created on a different CPU/GPU architecture and some of the synchronization barriers require some tuning for more general application. Due to time constraints we focused on those working out without modification. Luckily, this set includes at least one test from each collaboration pattern.

The 'D' and 'U' results are represented in Figure 4 where we chart the execution times as measured but the tooling of the CHAI benchmark set. Since the allocation times were small, generally less than 2% of the total, we summed these with the any explicit copies and summed them into the category of data movement (DM). The data is normalized to the 'D' results to better illustrate the comparison between the two implementations. The results shows that the 'U' cases have faster execution times largely due to the reduction in explicit data movement. The kernel execution time in the 'U' cases is generally larger; we suspect that is because the kernels are stalled at some points waiting for data.

The PAD testcase is interesting outlier. It was the only algorithm that did better with explicit memory copies to and from the GPU than relying on run-time to facilitate the CPU/GPU copies. We took the liberty of adding a cudaMemAdvise routine for the data array in the benchmark code and the kernel time within a few usecs of the CUDA-D version. Since the focus is on the underlying hardware functions, continued with this change. Otherwise, the PAD kernel execution time would have been 3x the CUDA-D version (larger than any other benchmark in the suite).

VI. Conclusion

With the hardware support in the AC922, the measured RDMA performance supports the most optimistic premise on

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

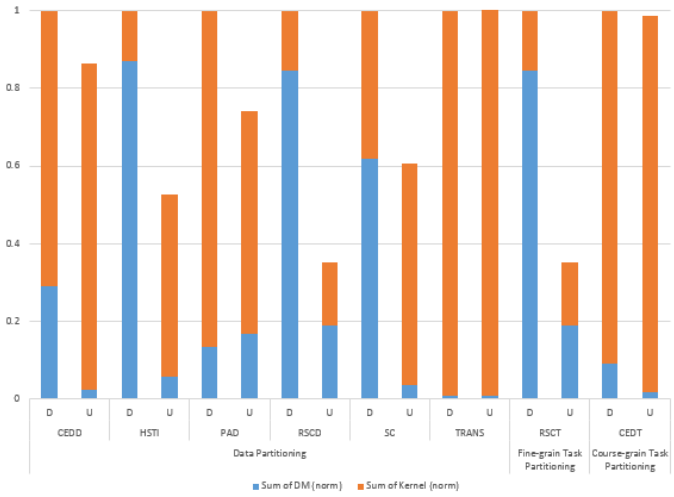

Fig. 4. Evaluation of CHAI Workloads

software controlled page fault latency; a 4 KB page takes 17 μs to complete where by the optimistic lower bound for software control page fault resolution was 20 μs.

Using the CHAI benchmark, we have demonstrated that the data movement features between the CPU and GPU in the AC922 allow for naive data or task partitioning schemes to perform comparable to fix partitioning. The allocation time in previous reported results is significantly reduced on the AC922 making the tests employing unified virtual memory, coherence, and system atomics execution times faster than the more traditional offload approaches.

The low latency RDMA results combined with reduction in data movement in the CHAI benchmarks shows promise for successful implementations of on-demand-paging and memory prefetch algorithms to move data to the targeted compute engines.

VII. Future Work

Results for workload level CAPI ATS evaluation with ODP will be future work.

Acknowledgment

We would like to thank Robert Blackmore for his guidance and education on the inter-workings of the PAMI software and interplay of these libraries with On-Demand Paging. In addition, we thank John Irish and Brian Rodgers from the NPU design team who provided insights and test cases to help us understand the design choices made in hardware and how they manifest as faster execution times.

References

[1] https://github.com/linux-rdma/perftest. Accessed: 2018-07-05.

[3] Rajesh Bordawekar and Pidad Gasfer D'Souza. Evaluation of hybrid cache-coherent concurrent hash table on ibm power9 ac922 system with nvlink2. http://on-demand.gputechconf.com/gtc/2018/video/S8172, 2018. Accessed: 2018-07-05.

[2] InfiniBand Trade Association. *InfiniBand architecture specification: release 1.0*. Number v. 1 in InfiniBand architecture specification: release 1.0. InfiniBand Trade Association, 2000.

[4] D. Bureddy, H. Wang, A. Venkatesh, S. Potluri, and D. K. Panda. Omb-gpu: A micro-benchmark suite for evaluating mpi libraries on gpu clusters. In *Proceedings of the 19th European Conference on Recent Advances in the Message Passing Interface*, EuroMPI'12, pages 110–120, Berlin, Heidelberg, 2012.

[5] V. García-Flores, E. Ayguade, and A. J. Peña. Efficient data sharing on heterogeneous systems. In *2017 46th International Conference on Parallel Processing (ICPP)*, pages 121–130, Aug 2017.

[6] Juan Gómez-Luna, Izzat El Hajj, Victor Chang, Li-Wen Garcia-Flores, Simon Garcia de Gonzalo, Thomas Jablin, Antonio J Pena, and Wen-mei Hwu. Chai: Collaborative heterogeneous applications for integrated-architectures. In *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*. IEEE, 2017.

[7] Richard L. Graham, Devendar Bureddy, Pak Lui, Hal Rosenstock, Gilad Shainer, Gil Bloch, Dror Goldenerg, Mike Dubman, Sasha Kotchubievsky, Vladimir Koushnir, Lion Levi, Alex Margolin, Tamir Ronen, Alexander Shpiner, Oded Wertheim, and Eitan Zahavi. Scalable hierarchical aggregation protocol (sharp): A hardware architecture for efficient data reduction. In *Proceedings of the First Workshop on Optimization of Communication in HPC*, COM-HPC '16, pages 1–10, Piscataway, NJ, USA, 2016. IEEE Press.

[8] K. Hamidouche, A. Venkatesh, A. A. Awan, H. Subramoni, C. H. Chu, and D. K. Panda. Exploiting gpudirect rdma in designing high performance openshmem for nvidia gpu clusters. In *2015 IEEE International Conference on Cluster Computing*, pages 78–87, Sept 2015.

[9] IBM. Functionality and performance of nvlink with power9 processors. *IBM Journal of Research and Development*, page to appear in, July 2018.

[10] S. Kumar, A. R. Mamidala, D. A. Faraj, B. Smith, M. Blocksome, B. Cernohous, D. Miller, J. Parker, J. Ratterman, P. Heidelberger, D. Chen, and B. Steinmacher-Burrow. Pami: A parallel active message interface for the blue gene/q supercomputer. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pages 763–773, May 2012.

[11] Sameer Kumar, Robert Blackmore, Sameh Sharkawi, Nysal Jan K. A., Amith Mamidala, and T. J. Chris Ward. Optimization of message passing services on power8 infiniband clusters. In *Proceedings of the 23rd European MPI Users' Group Meeting*, EuroMPI 2016, pages 158–166, New York, NY, USA, 2016. ACM.

[12] NVIDIA. Gpu direct. https://developer.nvidia.com/gpudirect. Accessed: 2017-03-23.

[13] J. Power, A. Basu, J. Gu, S. Puthoor, B. M. Beckmann, M. D. Hill, S. K. Reinhardt, and D. A. Wood. Heterogeneous system coherence for integrated cpu-gpu systems. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 457–467, Dec 2013.

[14] R. Shi, S. Potluri, K. Hamidouche, J. Perkins, M. Li, D. Rossetti, and D. K. D. K. Panda. Designing efficient small message transfer mechanism for inter-node mpi communication on infiniband gpu clusters. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10, Dec 2014.

[15] J. Stuecheli, B. Blaner, C. R. Johns, and M. S. Siegel. Capi: A coherent accelerator processor interface. *IBM Journal of Research and Development*, 59(1):7:1–7:7, Jan 2015.

[16] Rajeev Thakur, Pavan Balaji, Darius Buntinas, David Goodell, William Gropp, Torsten Hoefler, Sameer Kumar, Ewing Lusk, and Jesper Traff. Mpi at exascale. http://aegjcef.unixer.de/publications/img/mpi_exascale.pdf. Accessed: 2017-05-11.

[17] T. Zheng, D. Nellans, A. Zulfiqar, M. Stephenson, and S. W. Keckler. Towards high performance paged memory for gpus. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 345–357, March 2016.