Designing High Performance and Energy-Efficient MPI Collectives for Next Generation Clusters

Problem Statement: The past decade has seen a renaissance in architecture domain space primarily due to the culmination of Dennard's scaling. Processor architects have since relied on parallelism and reduced clock rates to keep temperature under the required thermal envelope in an attempt to sustain performance scaling. In addition to slower parallel processing units, other specialized many-core devices with simplified logic which lend themselves for scientific computations, have recently emerged and are growing in usage owing to wider applicability. Instances of these include the now mainstream NVIDIA GPGPUs and Intel Many-Integrated-Core (MIC) devices. On the other hand, as these devices are primarily available as PCIe devices (which are co-located with traditional general purpose processors), there are some inherent communication drawbacks. Hence programming models (MPI/PGAS) which enable communication on these processors must distinguish their compute and communication capabilities and ensure that data transfer operations are optimized to ensure high performance, especially long duration collective calls. Traditional collective algorithms developed since the inception of MPI such as Bruck's Alltoall/Allgather algorithms, Rabensiefner's scatter-allgather allreduce algorithm, etc generally made uniformity assumptions of compute and communication costs. As PCIe devices render systems heterogeneous by introducing new communication paths and new memory access costs, new algorithms that are cognizant of these differences are needed to ensure low latency. Figure 1 below, shows an instance of a single heterogeneous node in a GPU cluster that exhibits this nature with peak bandwidth rates of accessing CPU and GPU memories from different entities. Similar trends are seen with MIC-enabled systems as well. From this, it is evident that the use of traditional algorithms, which assume uniformity, routinely take non-optimal path which leads to communication degradation and potentially causes overall application degradation. To overcome these limitations, this work first asks the following questions:

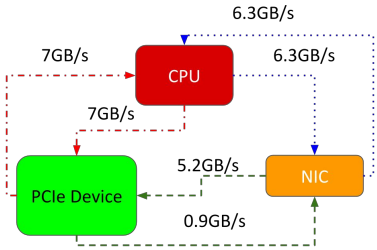


Figure 1

1. *Can variations of popular collective algorithms be proposed that is better suited towards platforms with heterogeneous communication cost paths and compute capacities?*
2. *Can new heuristics be proposed that tries to minimize the communication cost in heterogeneous clusters?*

Orthogonally, novel PCI express peer-to-peer memory access technologies such as NVIDIA's GPUDirect RDMA [GDR] have been introduced for low latency transfers through direct GPU memory access. However, architectural idiosyncrasies limit their potential benefits (read bandwidth < 1 GBps as shown in Figure 1) and alternate strategies are needed for peak performance in the context of throughput-critical streaming applications and in the context of non-blocking GPU collectives. Hence the following question arises:

3. *Can direct-GPU access mechanisms such NVIDIA GPUDirect-RDMA be coupled with existing paradigms such as the hardware multicast feature for throughput oriented applications?*
4. *Can GPUDirect-RDMA and associated CUDA features be combined to realize efficient non-blocking GPU collectives for good overlap and latency?*

While communication latency and throughput can be a key dictator of the application's overall performance, metrics such as energy and power consumption of distributed scientific applications have become critical concerns, particularly with Exascale machines in mind. The majority of works present in

literature up till now present works that 1. View communication as a black box and apply stringent power saving schemes during communication routines that adversely impact their performance 2. Monitor communication behaviour and identify critical path before applying power saving methods appropriately (Jitter, Adagio). These methods have limitations in the general case where not all communication routines present opportunities for power savings or where critical path of the application constantly moves. To address these concerns, this work asks these questions:

5. Can a set of generic rules be proposed for point-to-point and collective routines such that energy savings are made only at opportune moments with little to no performance degradation?

6. Can these rules ensure energy-savings in an application-oblivious manner and not just to well balanced applications?

Major Contributions/Results:

The designs conceived as part of this thesis addresses the above six challenges and has realized them for production systems through the MVAPICH2 MPI project. Figure 2 shows the overall the areas affected by the contributions from this work (with the work's specific designs and future goals highlighted in blue). In this section, the designs contributed has been discussed in more detail.

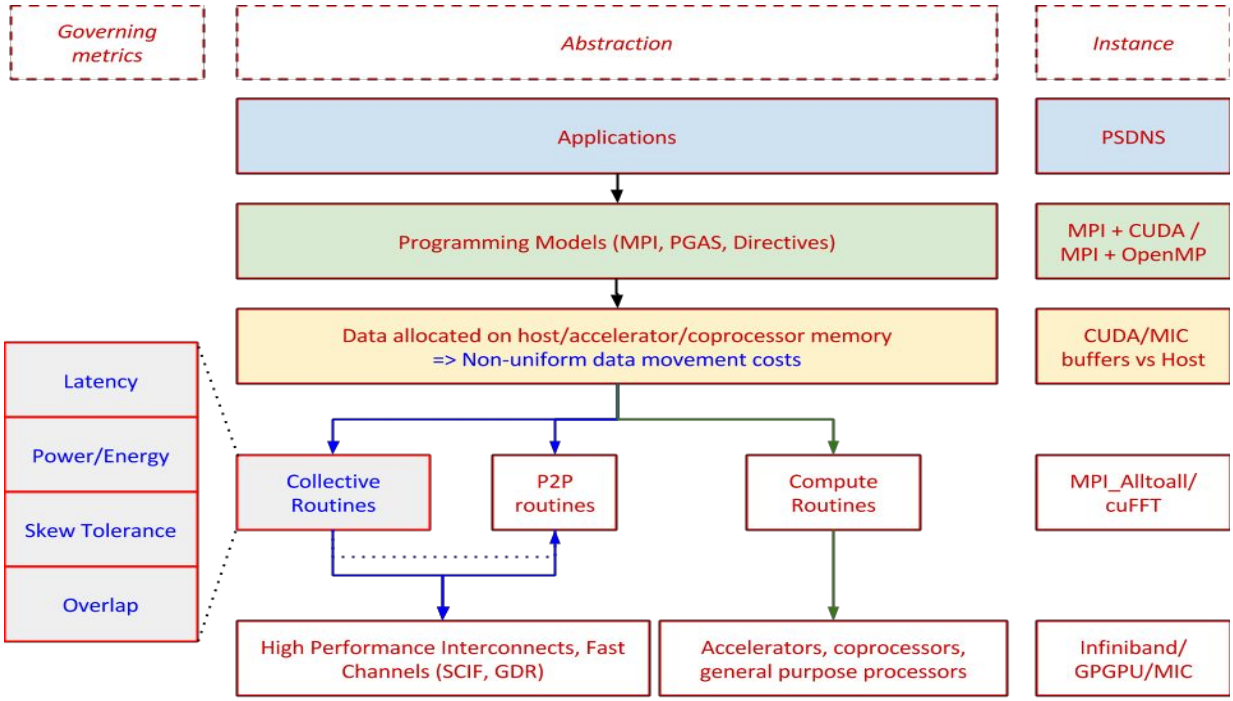


Figure 2

Collectives for Intel-MIC architecture: The current generation Intel Many Integrated Core architecture, known as Knight's Corner, is composed of 60/61 simplified x86 cores. It is available as a PCIe device with limited memory per core (16 or 32 GB in total). MPI jobs that leverage MIC devices generally do so in the 'symmetric mode'. In this mode, each node is composed of both a host processor and a MIC processor; and MPI processes are executed on both the platforms. However there are two primary drawbacks: 1. The simplification of cores (lack of out-of-order execution and simplified branch prediction

logic) renders them less powerful than their peers on the general purpose host cores. 2. Although the processes on the MIC can directly access the network controller (through PCIe peer-to-peer communication), there exists some architectural limitations in reading and writing the MIC's memory (especially the large message range) on the network controller's part (similar to Figure 1). Owing to this, the performance of critical collective operations such as MPI_Bcast, MPI_Allreduce, MPI_Gather, MPI_Alltoall and MPI_Allgather is unfavorable. To address these drawbacks, this work has proposed a set of heuristics to minimize usage of non-optimal communication paths that are either adaptations of state-of-the-art algorithms suited for heterogeneous environments or through delegation mechanisms where beefy cores available on the host processor assist the simplified cores in performing network intensive operations.

Consider the scenario of a 2-node MPI job for an MPI_Bcast operation where, on each node, there exists 2 MPI processes on the host and 2 MPI processes on the MIC. If the root of the broadcast is located on the MIC, then the root is responsible for performing sends to all other processes. This leads to a net cost of ($1 * t_{intra\text{-}mic} + 2 * t_{intra\text{-}node\text{-}from\text{-}mic} + 4 * t_{inter\text{-}node\text{-}from\text{-}mic}$). The bottleneck in this pattern is the last component ($4 * t_{inter\text{-}node\text{-}from\text{-}mic}$) owing to the slow reads of data from the MIC's memory by the NIC in addition to the fact that 7 communication steps are driven by the process on the weak MIC core. The work proposes alleviation of this cost by requiring a delegation process on the host to carry over the task of disseminating the data to the rest of the processes in the communicator once it has received a copy. The critical path in the broadcast now switches to host-to-host memory inter-node transfer which can be a few factors less than inter-node transfer from MIC memory. Extensions such as multi-level hierarchical dissemination, use of multiple delegation processes and use of overlapped-pipelined transfers (where non-blocking sends/recvs are used) have also been proposed for operations such as MPI_Allreduce and MPI_Gather in the related works published [11, 17]. Figure 3 shows examples of the proposed schemes with rank 7 as root with 60%/70% latency reduction for Bcast/Gather.

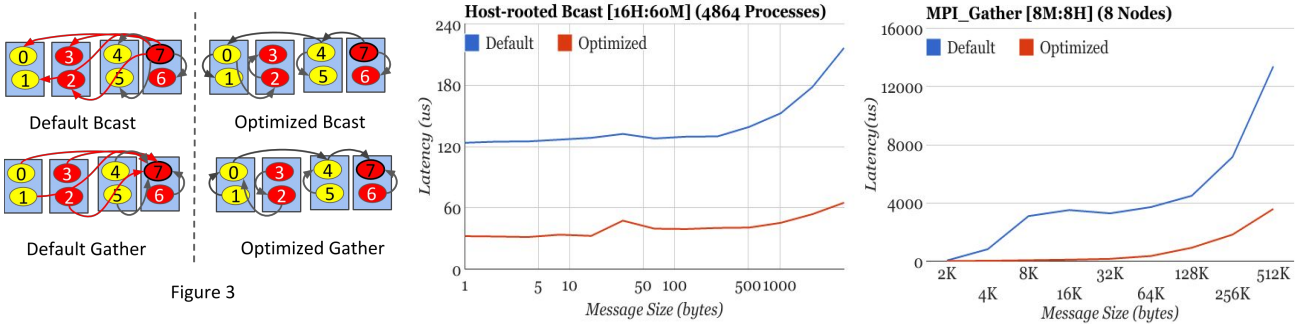


Figure 3

For dense collective operations such as MPI_Allgather and MPI_Alltoall, the work identifies suboptimal paths used in state-of-the-art and proposed optimizations to the popular ring and recursive-doubling algorithms for allgather operations; and to bruck's and pairwise algorithms for alltoall operations. The key idea with both these sets of algorithms was to reorder schedules to minimize transfer of large data on slower links. For instance, in the recursive doubling algorithm, the size of the message doubles in each step and the use of the regular algorithm, being unaware of slower PCIe paths, resulted in the largest transfer over the slowest paths. To avoid this, the contributed work [8] reorders the steps such that the smallest messages go over the weaker links. For the ring algorithm, where performance is dictated by the

slowest link, the work identifies the slowest path (generally between a mic process on one node and a mic/host process on another node) and reorders ranks such that the ring is composed of connections between just host processes over the network. This is possible as long as there is at least one host process on each node. The schemes are shown in Figure 4 with corresponding results (60% latency savings in allgather/alltoall).

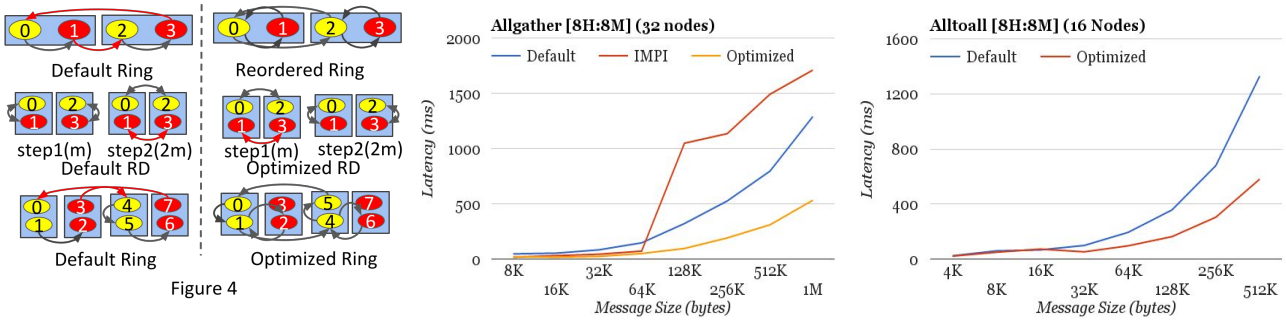


Figure 4

Throughput-oriented Broadcast for GPU-clusters with GPUDirect RDMA and hardware multicast: Streaming class applications are generally characterized by a live source disseminating data to processing sites equipped with GPUs. As a result, these rely heavily on the throughput of broadcast operations directly to GPU memories. The work proposed the amalgamation of Infiniband hardware multicast feature with NVIDIA's GPUDirect RDMA feature to achieve this[6]. The idea is to leverage Infiniband's scatter-gather list abstraction to conserve the PCI bandwidth and avoid expensive cuda memory copy operations. Figure 5 shows the scheme and results which show over 50% latency-reduction.

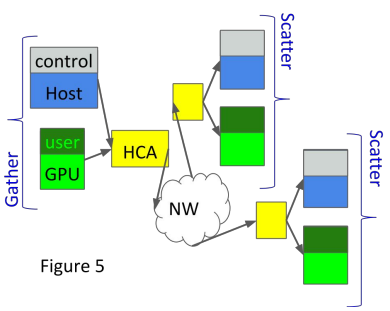
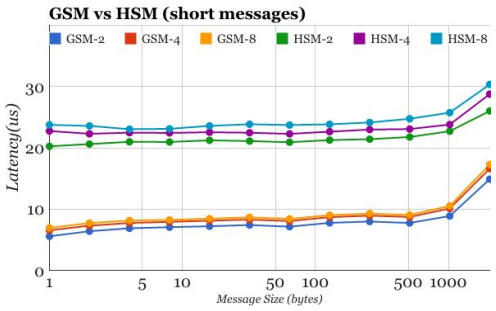


Figure 5



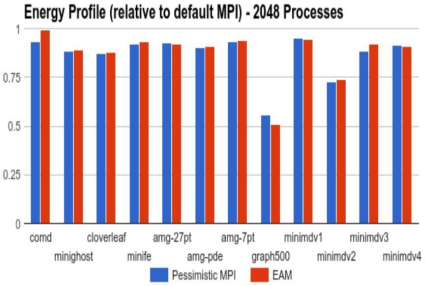


Figure 6

Energy-aware MPI Runtime: The majority of energy-aware solutions hitherto aims to conserve energy for iterative and predictable applications. The designs proposed [1] leverages on intimate knowledge of MPI point-to-point protocols such as Eager protocol, Rendezvous RDMA-Read/Write protocol; and collective algorithm knowledge (k-nomial trees, ring pattern, etc) to conserve energy for both collectives and the whole application during communication phases. It does so by applying energy-levers in a timely manner without relying on application knowledge and hence achieves energy savings for a broad class of applications including non-iterative and non-temporal classes without degradation. Figure 6 summarizes savings in applications where energy savings in the range of 5% - 28% can be seen with as many as 4K processes.