High Performance Middlewares for Next Generation Architectures: Challenges and Solutions

Sourav Chakraborty, Advisor: Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering, The Ohio State University
{chakraborty.52, panda.2}@osu.edu

ABSTRACT

The emergence of modern multi-/many-core architectures and high-performance interconnects have fueled the growth of large-scale supercomputing clusters. Due to this unprecedented growth in scale and compute density, high performance computing (HPC) middlewares now face a plethora of new challenges to solve in order to extract the best performance from such systems. In this work, we study four such challenges — a) launching and bootstrapping jobs on very large scale clusters, b) contention in collective communication, c) point-to-point communication protocols, and d) scalable fault-tolerance and recovery and propose efficient solutions for them. The proposed solutions have been implemented on MVAPICH2, a popular MPI and PGAS runtime used by scientists and HPC clusters around the world.

1 INTRODUCTION

Modern high-performance computing (HPC) systems offer sustained multi peta-flops performance and are enabling scientists to scale their parallel applications to hundreds of thousands of processors. The massive growth in the size and scale of supercomputing systems over the last decade has been driven by the current trends in multi-/many-core architectures such as Xeon Phi and OpenPOWER and the availability of commodity, RDMA-enabled, and high-performance interconnects such as InfiniBand [1] (IB) and Omni-Path [2]. This increase in system size coupled with the diversity of modern architectures have brought forward several challenges in terms of performance, scalability, and resilience. HPC middlewares need to keep up with this growth and address these challenges to provide the best performance to the application developers. In this work, we investigate some of these challenges and propose efficient and scalable solutions to address them. We also evaluate the proposed designs against state-of-the-art solutions at scale and show significant benefits to end-users.

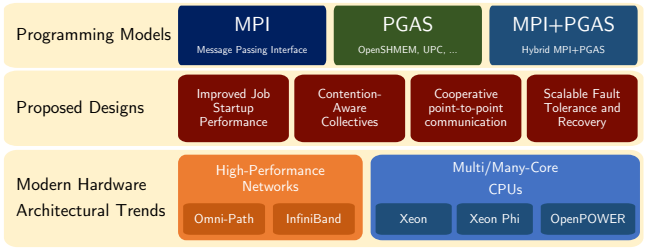

Figure 1: Overview of Proposed Designs

2 RESEARCH FRAMEWORK

Figure 1 illustrates the four major concerns addressed in this work. We focus on Message Passing Interface (MPI) and Partitioned Global Address Space (PGAS) middlewares since these two programming models encompass a large majority of parallel applications. However, many of the same challenges are faced by other high-performance communication runtimes and are applicable to them as well. Broadly, these challenges are:

- **Job startup**: How can we efficiently launch and initialize MPI and PGAS jobs on very large scale clusters with thousands of nodes? How can the memory footprint be kept low?
- **Collective communication**: Kernel assisted communication is used in many parallel programming libraries due to its "single-copy" nature. However, this can suffer from severe lock contention depending on the communication pattern. Can we propose new algorithms or designs to mitigate this?
- **Point-to-point communication**: MPI libraries use a variety of communication protocols based on the message size and other factors. How can we identify the limitations of the existing designs and propose novel "cooperative" protocols to improve the application performance?
- **Fault-tolerance**: Long-running jobs on large number of nodes require efficient fault-tolerance (FT) mechanisms to avoid lost work in case of system or software failures. Can we propose robust and scalable designs to achieve efficient fault tolerance and recovery?

3 PROPOSED SOLUTIONS AND DESIGNS

In this section, we describe the proposed solutions to tackle the aforementioned challenges and highlight some of the key results obtained using these designs.

3.1 Improving Job Startup Performance

The ability to quickly launching and bootstrap parallel jobs on a large number of nodes is important for reducing development and debugging effort, regression testing and benchmarking, and increasing overall system efficiency. RDMA-enabled networks such as InfiniBand and Omni-path require exchanging network endpoint addresses and other information with the peer processes before they can start communication. The Process Management Interface (PMI) was proposed to facilitate this exchange. However, existing exchange mechanisms were not efficient and scalable [8]. We proposed several efficient and non-blocking PMI extensions [5, 8] to a) reduce the overall data movement, b) utilize the high-performance network to exchange most of the data, and c) overlap the exchange phase with other initialization tasks. We also proposed dynamic connection management schemes for OpenSHMEM and other PGAS programming models [7] to further reduce the startup time. Finally, we proposed a shared-memory backed design between the job launcher and the communication runtime to avoid data duplication and reduce memory consumption among the processes on the same node [9]. These improvements resulted in up to 10x and

30x improvement in MPI and OpenSHMEM initialization times on 16,384 processes on 1,024 nodes on the TACC Stampede system (SandyBridge+FDR). On Stampede2 (KNL+Omni-path), the proposed design was able to initialize MPI at full scale (230K processes on 3,584 nodes) under 22 seconds. Compared to the vendor provided Intel MPI 2018, the proposed design performed more than 8x better with 128K processes. The memory consumption was also reduced by an estimated 4GB per node with 64 processes per node and a million processes.

3.2 Contention-aware Kernel-assisted Collectives

Due to the increased core count on modern multi-/many-core architectures, intra-node communication makes up a significant portion of the overall communication done by parallel applications. Kernel-assisted mechanisms such as Cross-memory-attach (CMA) allows processes to directly read or write the data from another process on the same node by issuing a system call. This technique is typically used by middlewares for medium and large messages to avoid two copies required by POSIX shared memory designs. However, these kernel-assisted mechanisms need to lock the page table of the source/target process which results in significant lock-contention and performance degradation with one-to-all and all-to-one communication patterns. We analyzed the cost of this contention and developed a model to predict the cost of different communication patterns and message sizes. Based on this, we proposed contention-aware kernel-assisted designs for several collectives including Broadcast, Scatter, Gather, Allgather, and Alltoall [6]. We compared the proposed algorithms against state-of-the-art MPI libraries including Intel MPI 2017 and Open MPI v2.1.0 on three modern architectures - Broadwell, KNL, and OpenPOWER and showed up to 4x, 5x, and 14x improvements.

3.3 Cooperative Rendezvous Protocols

High-performance MPI/PGAS middlewares broadly use two protocols - "Eager" and "Rendezvous" to perform point-to-point communication. Rendezvous protocols utilize one or more control messages between the sender and the receiver to initiate the data transfer notify the peer upon completion. Existing rendezvous protocols only utilize the sender or the receiver CPU to perform the data movement and thus do not take advantage of all the available resources. Furthermore, they do not take into account the overall communication pattern and can lead to load-imbalance on the CPU cores. They also do not provide the best overlap of intra-node and inter-node communication. To mitigate these issues, we proposed a family of "cooperative" rendezvous protocols that a) utilize all the available resources, b) dynamically adapt to the communication pattern to improve load balance, and c) improve overlap of intra-node and inter-node communication [3]. Compared to existing designs, the proposed designs showed up to 19%, 16%, and 10% improvement with Graph500, CoMD, and MiniGhost with 896 processes on a Broadwell+EDR cluster.

3.4 Scalable Fault-tolerance and Recovery

Due to the tremendous growth in system sizes and long-running jobs, resilience and fault-tolerance of parallel applications have become important. However, proposed FT solutions for MPI are not scalable and difficult to adopt into bulk-synchronous processing (BSP) applications. We proposed and implemented a novel FT interface named "EReinit" (Efficient Reinitialization) for such applications. One of the novel contributions of this work was co-designing the resource manager and the MPI library to achieve high-performance and scalable failure detection, propagation, and recovery [4]. The proposed design can recover from checkpoints stored in parallel file-system as well as in-memory checkpoints. The proposed design was able to recover from single process failures within 2.25 seconds with 4,096 processes on a SandyBridge+QDR system. Application level studies with Enzo and CoMD showed that the recovery time with the proposed designs were up to 4 times faster than the existing designs.

4 FUTURE WORK AND BROADER IMPACT

We plan to continue our research in improving the performance and scalability of HPC middlewares on modern architectures. Some of the planned and in-progress research activities include studying the impact of proposed designs on accelerators, hybrid programming models such as MPI+threads, highly dynamic and adaptive MPI runtimes, and larger scale application-level studies.

The proposed designs are publicly available in the latest releases of MVAPICH2 and MVAPICH2-X and are being used in many large-scale production HPC clusters including several systems in the TOP500 list. This software is being used by more than 2,925 organizations in 86 countries worldwide to extract the potential of these emerging networking technologies for modern systems. As of Aug '18, more than 485,000 downloads have taken place from this project's site. The authors have also contributed some of the proposed designs to other public open-source software such as Slurm, a widely used resource manager on HPC systems.

REFERENCES

[1] 2017. Infiniband Trade Association. (2017). http://www.infinibandta.org/

[2] M S Birrittella, M Debbage, R Huggahalli, J Kunz, T Lovett, T Rimmer, K D Underwood, and Robert C Zak. 2015. Intel® Omni-path Architecture: Enabling Scalable, High Performance Fabrics. In *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*. IEEE, 1–9.

[3] Sourav Chakraborty, Mohammadreza Bayatpour, Jahanzeb Maqbool Hashmi, Hari Subramoni, and Dhabaleswar K Panda. 2018. Cooperative Rendezvous Protocols for Improved Performance and Overlap. In *Supercomputing, 2018. SC'18. Proceedings of the 2018 ACM/IEEE Conference on*. IEEE, (In press).

[4] Sourav Chakraborty, Ignacio Laguna, Murali Emani, Kathryn Mohror, DK Panda, Martin Schulz, and Hari Subramoni. 2017. EReinit: Scalable and Efficient Fault-Tolerance for Bulk-Synchronous MPI Applications.

[5] S Chakraborty, H Subramoni, A Moody, A Venkatesh, J Perkins, and D K Panda. 2015. Non-Blocking PMI Extensions for Fast MPI Startup. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE, 131–140.

[6] Sourav Chakraborty, Hari Subramoni, and Dhabaleswar K Panda. 2017. Contention-Aware Kernel-Assisted MPI Collectives for Multi-/Many-Core Systems. In *Cluster Computing (CLUSTER), 2017 IEEE International Conference on*. IEEE, 13–24.

[7] S Chakraborty, H Subramoni, J Perkins, A A Awan, and D K Panda. 2015. On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2015 IEEE International*. IEEE.

[8] S. Chakraborty, H. Subramoni, J. Perkins, A. Moody, M. Arnold, and D. K. Panda. 2014. PMI Extensions for Scalable MPI Startup. In *Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/ASIA '14)*.

[9] S Chakraborty, H Subramoni, J Perkins, and D K Panda. 2016. SHMEMPMI: Shared Memory Based PMI for Improved Performance and Scalability. In *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*. IEEE.