DOI: 10.14529/jsfi160306

HCA aware Parallel Communication Library: A feasibility study for offloading MPI requirements

Kedar Kulkarni[1], *Shreeya Badhe*[1], *Geetanjali Gadre*[1]

Message Passing Interface (*MPI*) is a standardized message passing system, independent of underlying network, and the most widely used parallel programming paradigm. The communication library should make full use of the Host Channel Adapter (*HCA*) characteristics to maximize performance of the HPC cluster. The communication library may not able to take full advantage of the underlying network adapter, if the library is made generalized. This can have a significant impact on the performance.

Our primary goal is to develop a network dependent message passing library called a Parallel Communication Library (*PCL*) that will exploit C-DAC's proprietary PARAMNet HCA [1] features for efficient message transfer. Using PCL, we intend to observe the feasibility of the network and performance enhancement for additional features. The objective is to carry out different trials by implementing additional features and analyze the implications which would give us more insight towards suitability of transport offload/onload mechanism. This experimentation would give us feedbacks for designing the next generation architecture.

Keywords: MPI, HCA, Communication Pattern Offloading, High Performance Networks, Communication Library.

Introduction

Designing a high performance network is a critical and an arduous task. It involves work at multiple layers. The work is carried out at both software and hardware front. The software stack involves development of driver, communication library and message passing library. The hardware development is mainly focused on low latency and high bandwidth data transfer, and providing support for multiple protocols.

Today, typical High Performance Computing (*HPC*) clusters are commodity based clusters that employ OFED [2] software. OFED software provides interface to many applications regardless of underlying Host Channel Adapter (*HCA*). OFED software offers abstraction to the MPI layer and handles all low level activities such as connection management, basic data structures for connections, etc. Similarly, due to the communication stack hierarchy HCA becomes unaware of end application communication pattern. Typically, proprietary network interconnect developer uses OFED software to run MPI based applications. HCA has to provide various features in order to support OFED, such as unreliable datagram service to use OpenSM. Furthermore, due to the standard and generic structure of the communication stack it is possible that the upper layers may not exploit any additional feature provided by HCA that can improve the performance of the entire HPC cluster.

Message Passing Interface (*MPI*) [3] has become the de facto standard for writing parallel applications in HPC domain. To increase the performance of a HPC cluster, the communication library should exploit all HCA features efficiently. It may not be inappropriate to say that, at least in case of proprietary networks that use standard software stack, the upper layers do not comprehensively use of HCA features. This arises as a result of intricacy in modifying the complete software stack.

[1]Centre For Development of Advanced Computing, Bangalore, India

Our primary goal is to develop a network dependent message passing library called a Parallel Communication Library (*PCL*) for PARAMNet HCA. Using PCL, we intend to observe the feasibility of the network with support of additional features. This would help us to evolve the next generation architecture. Furthermore, any performance improvement observed would help us in contributing to the entire HPC community.

1. Scope

Traditional HPC communication stack is a multi-layer architecture. The multi-layer architecture creates a layer of abstraction between an underlying network and upper communication libraries. One peculiar problem with this approach is the communication library will not be able to use additional features provided by an underlying HCA, even though making use of the additional features could improve the performance. To extract maximum performance out of the underlying network, the HCA and the communication library should work in complementary manner. This paper describes a network dependent library, PCL, and its communication architecture for C-DAC's PARAMNet HCA. Our goal is to develop a library that can provide MPI like interface to the application, while exploring how these calls can be effectively implemented using PARAMNet HCA features. The purpose for developing our own library is to study the feasibility of HCA features that can be made available to the upper layers. Modifying standard communication library is an intricate process. It is easy to develop the library and support only necessary features. Using this approach, we can easily incorporate additional features in HCA and make them visible to an application using PCL.

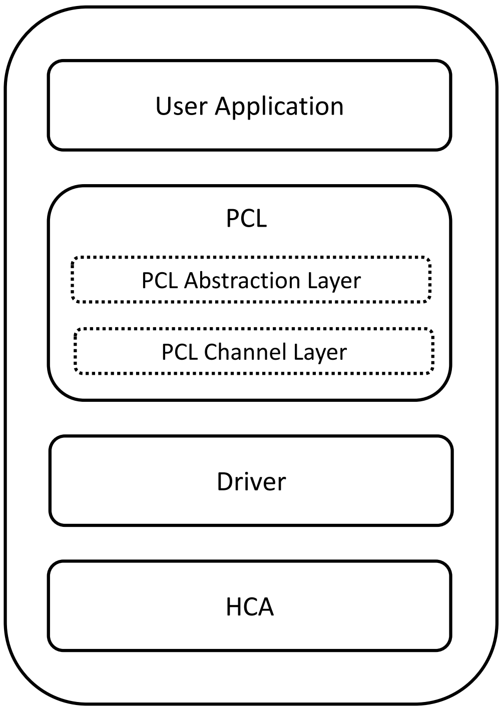

Figure 1. Communication Stack

2. Proposed Framework

The structure of PCL developed for PARAMNet HCA is shown in fig. 1. As shown in the figure PCL is primarily divided into two parts: the PCL abstraction layer and the PCL channel layer. The functionality of main components of PCL is as follows:

2.1. PCL Abstraction Layer

The PCL Abstraction layer provides application programming interface (*API*) for PCL. We have designed this layer to provide an interface similar to that of MPI. This layer is mainly responsible for following set of responsibilities:

- Handling the communication Pattern
  This layer handles the communication pattern that decides whether to use standard communication course or to make use of the additional features. For example, this layer can decide for small message whether to make use a low latency communication path or standard eager protocol.
- Managing connection data base
  In PCL application, each process is assigned a rank. PCL abstraction layer creates a data base for each rank. This data base holds all the information required for further data transfer, such as each rank is mapped to which hardware resources (*End-Points, Completion Queues, etc*).
- Buffer Management for Eager Protocol
  PCL supports eager protocol for message transfer. Allocation of buffers and managing these buffers are handled by PCL abstraction layer. The size of Eager buffer is programmable and controlled by *pcl_eager_threshold*.

Like MPI, the PCL works on reliable connection service. Therefore, PCL abstraction layer created all end-points in reliable connection mode. Along with reliable connection end-points, PCL abstraction layer also reserves an end-point per rank in the unreliable connection mode.

2.2. PCL Channel Layer

The PCL Channel layer handles all the communication with PARAMNet HCA. This layer handles responsibility such as posting work requests, creating an end-point. This layer hides low level details such as depth of work-queues, structure of work requests, end-points, completion queues and completion entry from the upper layer. This layer also translates hardware completion format into simpler format.

Along with these two layers a bash script, *pcl_exec*, is developed to work in background to assist to run an application on PCL. It also assists PCL in its working. This bash script will work as a daemon. It will spawn PCL processes, as user requests, on the nodes that are part of the cluster. Furthermore, it will bind each process to distinct core while spawning PCL processes.

Presently, we have implemented following functionalities:

- *pcl_init*
  This module initializes the setup, establishes connection amongst all the ranks, fill the receive work-queues with work requests in order to make every rank ready to accept data. This module also allocates eager buffers and pin those buffers.

- *pcl_finalise*
  This module is called for destroying all the HCA resources reserved during run time. Also, it frees all the eager buffers.
- *pcl_send*
  This module sends user buffer data to the destined remote rank. The internal implementation decides whether to use Eager or Rendezvous mode. Although, Eager mode is operational presently, and Rendezvous mode is under testing phase.
- *pcl_recv*
  This module polls on the receive data and after successful reception of data it copies intended data from eager buffer to application buffer.
- *pcl_barrier*
  The barrier operation is performed across all processes. It blocks until all processes have reached this routine. The detail implementation is explained in the next section.

3. Barrier Optimization

The Barrier function blocks the caller until all group members have called it. The function does not return on any process until all group processes have called the function. Barrier is been used by the applications quite frequently. The data transferred for barrier is very small as MPI transfers just header information. Using PARAMNet HCA feature, we have optimized barrier functionality for PCL. The barrier payload resides in the host memory. In typical implementation of communication stack, the communication library posts barrier call. For the lower layer this call is another data transfer requests, it simply post send work requests, which specify barrier data transfer request, to the HCA. The HCA has to read this data from host memory and then forward it to the destination. Typical PCI Express [4] latency for smaller host memory read requests is quite high due to round trip latency. This increases barrier execution time. PARAMNet HCA supports data in work request (INLINE DATA) feature. As explained earlier, size of data transfer during barrier is very small, we implemented barrier functionality using data in work request feature. In this support, while posting barrier call the payload for barrier is also given to the HCA. In this implementation HCA avoids host memory read to fetch barrier payload. Using data in work request feature, we have observed performance improvement upto 20%.

4. Conclusion and Future Work

This paper describes development of HCA aware Parallel Communication Library. Also, we presented implementation of barrier functionality using feature supported by HCA and we have observed performance improvement. We also realize that in order to meet higher performance, the communication library must know underlying hardware and implement communication calls accordingly.

In future, we plan to optimize collective calls such as Broadcast. We also plan to explore the advantage of supporting vector data type processing. We also plan to comply with OpenFabrics Interface (OFI) [5] which focuses on the development of software interfaces co-designed with fabric hardware.

References

1. C-DAC, PARAMNet-3 Network Interface Card (NIC) based on Gemini Co-processor. `http://www.cdac.in/index.aspx?id=hpc_ss_nic`

2. OpenFabrics Enterprise Distribution (OFED) Software Overview. `https://openfabrics.org/index.php/openfabrics-software.html`

3. Message Passing Interface (MPI) Standards. `https://www.mpi-forum.org/`. Last accessed: 2016-07.

4. PCI Express Specifications. `https://pcisig.com/specifications`

5. OpenFabrics Interfaces Working Group (OFIWG). `https://www.openfabrics.org/index.php/working-groups-overview.html/`