Mapping HALO Exchange onto Toruses and Stuff

Yadu Nand*†‡, Timothy G. Armstrong*
*Dept. of Computer Science, University of Chicago, Chicago, IL, USA
†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA
‡Computation Institute, University of Chicago and Argonne National Laboratory, Chicago, IL, USA

***Abstract*—Abstract != Introduction. [TODO]**

I. Introduction

Halo exchange is a common communication pattern in parallel codes, where each process is assigned an application subdomain and must periodically communicate with other processors that have neighboring subdomains to update information about the state of the boundary between subdomains. A common special case is when a multi-dimensional cartesian space is decomposed into subdomains of equal size. For example, in the three-dimensional case, a 8x8x8 cube might be decomposed into 256 2x1x1 cubes for execution on 256 processors.

This paper explores the problem of mapping such multi-dimensional cartesian halo exchange communications onto parallel computers with hypercube or torus networks.

II. High-Performance Computer Networks

The state of the art in High Performance Computing(HPC) infrastructure, demands high-performance networks to support the movement of data between the nodes as well as to-and-from disk-arrays. HPC systems are increasingly architected with high radix interconnects such as hypercubes and N-dimensional tori. Parallel applications have a wide range of task placements options to exploit the network topology of these HPC systems. These networks have evolved to several different network topologies in order to support different requirements, and data movement patterns. For HPC applications which involve fine-grained communication, high-radix networks provide low latency, smaller diameter, and large bandwidth as multiple links along the multiple dimensions supprted.

We are using Message Passing Interface (MPI) to implement the messaging and synchronization aspects of the HALO exchange code, and hence the performance observed from running the application would be influenced by the behavior of MPI due to it's various protocols on the BlueGene/Q. There are four protocols supported by the MPI implementation used BlueGene/Q. The protocol utilized by MPI is determined by the size of the message that is being sent. The data sizes at which the switch to different protocol occurs is configurable, but for our experiments we are using the defaults on BlueGene/Q.

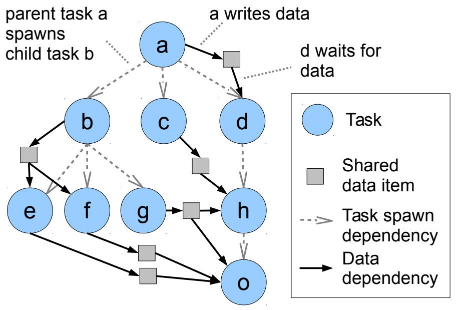

Fig. 1: This is a figure.

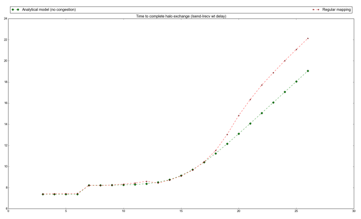

Fig. 2: This is a figure.

Protocol	Min Data Limit	Max Data Limit	Routing
Immediate	0	112	Direct
Short	113	496	Direct
Eager	497	2048	Direct
Rendezvous	2048	unlimited	Adaptive

We're going to cite Swift/T [2] and include an illustration (see Figure 1).

III. High-Performance Computer Networks

A. *Blue Gene/Q 5D torus*

RedBook [1]

B. *Cray Gemini 3D torus*

IV. Models for Network Communication

On an HPC system such as BlueGene/Q or Cray XE6, each node has multiple duplex links to it's neighbors. If the application on every node attempts to exchange messages with every neighbor, we can assume that every link on the network will see similar traffic. Thus we, consider a single link and it's bandwidth to determine t_b the time required to send a Byte along the link. Assume that there are N_{procs} number of identical processes all of which will attempt to utilize the same links. To capture the differences between different mappings we use a simple program to calculare the average distance between neighbors, N_{steps}. The current MPI implementation on leadership class systems like Mira (Bluegene/Q) utilizes shared memory for intranode communication. When a neighbor is present within the same node, the link is weighted as zero, and every network link or hop is weighted as one. Since it is possible to place neighboring tasks from the application domain on the same node there are mappings possible which minimize N_{steps} below one. There are constant costs involved in startup, acquiring a buffer etc, and t_c is an experimentally calibrated constant that is a catch-all for the various constant costs of network communication using MPI. N is the message size in bytes that are exchanged between neighbors.

A simple analytical model determines the time to completion, T of a complete halo exchange operation, from the variables defined above as follows:

$$T = t_c + (N_{steps} * N_{procs} * N * t_b) \tag{1}$$

Since we have a complex set of operations on several hundreds of nodes, there is a cost for synchonisation. We use MPI barriers to synchronise all tasks before and after measurements

V. Experimental Design

VI. Results

VII. Conclusion

References

[1] M. Gilge. *IBM System Blue Gene Solution Blue Gene/Q Application Development, Second Edition*. IBM Redbooks, 2013.

[2] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster. Swift/T: Large-scale application composition via distributed-memory data flow processing. In *Proc. CCGrid '13*.