A Model for Communication in Clusters of Multi-core Machines

Christine Task
Arun Chauhan

Computer Science Department
Indiana University
{ctask, achauhan} @cs.indiana.edu

Introduction

A common paradigm for scientific computing is distributed message-passing systems, and a common approach to these systems is to implement them across clusters of high-performance workstations. As multi-core architectures become increasingly mainstream, these clusters are very likely to include multi-core machines. However, the theoretical models which are currently used to develop communication algorithms across these systems do not take into account the unique properties of processes running on shared-memory architectures, including shared external network connections and communication via shared memory locations. Because of this, existing algorithms are far from optimal for modern clusters. Additionally, recent attempts to adapt these algorithms to multicore systems have proceeded without the introduction of a more accurate formal model and have generally neglected to capitalize on the full power these systems offer. We propose a new model which simply and effectively captures the strengths of multi-core machines in collective communications patterns and suggest how it can be used to properly optimize these patterns.

Collective Communications and Communications Models

Distributed systems computations typically follow the SPMD programming model. Problems in this paradigm are solved by parallel processes which interact frequently, completing small pieces of the problem and then exchanging information with other processes before proceeding. When these communications involve a large number of processes, a collective communication algorithm is used to optimize the communication pattern. Examples of collective communications are: broadcast, in which one process shares a piece of information with all other processes; gather, in which one process gathers a piece of information from every other process; and all-to-all, in which every process simultaneously broadcasts to all other processes. To perform any of these operations optimally in an arbitrary network is NP-complete.

Models have been created which abstract message costs and network behavior in order to provide a theoretical framework for developing communication algorithms. The simplest of these models is sometimes referred to as the “telephone model”. In this model, processes and network connections are represented by nodes and edges, respectively, in an undirected graph. Communication proceeds in discrete “rounds” with nodes able complete one message transfer across one network connection each round. Algorithm complexity is measured in the number of rounds to completion, and time estimates for real systems are attained by assigning a round duration which reflects the processing speed of the nodes and the latency of the network. As no more than two messages can be on any network link simultaneously, the telephone model is effective under very conservative bandwidth limits.

However, a commonly noted shortcoming of this model is that its assumptions are too conservative. Later models have eliminated discrete rounds and introduced parameters for message send cost, message receive cost, and network latency. Thus, if the time taken to send a message is less than the latency of the network, the sending node may proceed to send additional messages before its original message is received. One very popular model, LogP model, introduces two significant features. It neglects the underlying topology of the network, assuming each process may communicate with any other process over a connection with latency $L$. It also introduces a fourth parameter $g$ which represents the minimum gap between messages on the network, thus limiting the bandwidth of network connections to $1/g$.

Issues in Modeling Clusters with Shared Memory Machines

Existing models cannot represent the behavior of clusters which include multiple processes executing on the same machine. Consider a broadcast algorithm developed for processes on distinct machines, but applied to a cluster of multi-core machines. Broadcast to $n$ processes traditionally requires at best $O(log(n))$ messages. However, Open MPI is optimized to broadcast to processes executing on the same machine by placing messages in a shared memory location—only a single message is required. Similarly, a gather algorithm run on a graph model which believes the network latency to be the same for all edges has no preference for communicating between processes on the same machine over sending messages across the network to external processes. This could result in extremely inefficient
communication patterns. Additionally, processes running on the same machine must share their computer’s external network connections. It is not possible to represent this in either the LogP model or as a traditional undirected graph.

As described in [3], previous approaches to these problems have focused on hierarchical systems. In these systems, multi-core computers are considered to be single nodes in global communication patterns, and separate internal algorithms complete the communication among their processes. Thus, a broadcast would be performed exactly as though each machine represented a single node in the graph. However, this too overlooks an important feature of multi-core nodes: the ability to assemble and send messages out in parallel. Open MPI is optimized such that a multi-core machine with $n$ network devices and at least $n$ processes can assemble and send messages out on all $n$ connections simultaneously [2]. Treating multi-core computers as simple nodes overlooks the significant ability of individual processes within the machine to contribute to the global communication pattern.

Our Solution

The three points described above—the ability to write a message to processes in a shared memory machine in constant time, the difference between internal and external communication, and the ability of processes on multi-core machines to communicate in parallel with the outside world—have a significant impact on the development of efficient algorithms and are not representable in current models. Kumar, et. al. have proposed and tested a simple all-to-all algorithm which took several of these issues into account [3]. They achieved a performance improvement of 55% over commonly used algorithms. An accurate formal model of multi-core clusters will aid the development of communication algorithms which are carefully optimized and proven to run efficiently on modern systems.

Our model introduces three new rules to reflect the behavior of processes on shared memory machines:

- **Read Is Not Write:** A value can be written to any subset of processes on the same machine in constant time—in writing, a multi-core machine acts as a node. However, reading from these processes requires the time necessary to assemble the message at each process—in reading, a multi-core machine acts as a clique in which nodes share access to external network connections.

- **Local Edges Are Short, Global Edges Are Long:** Communication between processes on the same machine is considerably more efficient than communication with external processes. In cost models which assign latencies to edges, internal edges should be assigned a weight separate from external edges. In simplified, round-based models, we’ll assume any number of internal edges may be traversed during a single round and include this extra cost in our round length estimate.

- **Parallel Communication:** Processes on a multi-core machine may use their machine’s external network connections in parallel.

Current and Future Work

The proposed rules may be adapted to many different cost models, as they primarily affect the dynamics of the message passing system. For our initial work, we are focusing on the simple round-based telephone cost model. This model is conscious of network topology, limits bandwidth, and minimizes the number of network edges traversed. Algorithms which are efficient under these strict conditions will do well in general, and the simplicity of this model provides a good framework for exploring the implications of the proposed rules.

Our work to date has focused on the analysis of the broadcast and gather problems in multi-core clusters, including the performance of existing algorithms in our model and the development of algorithms better suited to this new environment. Certain interesting results are immediately apparent. We define a machine with $n$ network connections and at least $n$ processes to have *degree* $n$. Traditionally, optimal gather trees are the inverse of optimal broadcast trees, but this is not necessarily the case with multi-core clusters. A machine with degree $n$ can broadcast efficiently to its $n$ neighbors, but it is unable to simultaneously gather data from both them and its own $n$ processes. Additionally, “fastest node first” is a popular heuristic for broadcast across heterogeneous clusters. However, the similar “highest degree node first” is a poor heuristic for broadcast on non-sparse multi-core clusters. In these networks, nearby nodes with high degree are likely to have a large intersection of neighbors, and thus blindly prioritizing high degree nodes may not result in efficient coverage.

In the future, we intend to adapt this work to more realistic cost models and examine more complex communication problems including gossip and all-to-all.

References

[1] D. Culler, R. Karp, R. Subramonian, and T.V. Eicken. Logp: Towards a realistic model of parallel computation. In *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–12, 1993.

[2] R.L. Graham, T.S. Woodall, and J.M. Squyres. Open MPI: A flexible high performance MPI. In *Proceedings, 6th Annual International Conference on Parallel Processing and Applied Mathematics*, Poznan, Poland, September 2005.

[3] R. Kumar, A. Mamidala, and D.K. Panda. Scaling alltoall collective on multi-core systems. In *Parallel and Distributed Processing, 2008*, pages 1–8, 2008.