Accelerating MPI Collective Communications through Hierarchical Algorithms with Flexible Communication and Imbalance Awareness

Benjamin S. Parsons (bsparson@purdue.edu) - Advised by Vijay S. Pai
Purdue University - School of Electrical and Computer Engineering

Abstract

This work investigates collective communication algorithms on a shared memory system, and develops the universal hierarchical algorithm. This algorithm can pair arbitrary hierarchy unaware inter-node communication algorithms with shared memory intra-node communication. In addition to flexible inter-node communication, this algorithm works with all collectives, including those incompatible with past works, like alballo. The universal algorithm shows impressive performance results, improving upon the MPICH algorithms as well as the Cray MPT algorithms. Speedups average 15x - 30x for most collectives with improved scalability up to 8k cores.

The second part of this work creates new hierarchical collective algorithms designed to tolerate process imbalance. The process imbalance of benchmarks is thoroughly evaluated, and is used to design collective algorithms that minimize the synchronization delay observed by early arriving processes. Preliminary results for a reduction show speed-ups reaching 80x over a binomial tree algorithm in the presence of high, but not unreasonable, imbalance.

Universal Hierarchical Algorithm

Pseudocode for the universal hierarchical algorithm is given below. The CopyIn and CopyOut functions move data from the private buffers to shared memory. For collectives without a globally known communication pattern the ExchangeCnData function shares each processes count data with all processes colocated on a node. SetSplitter calculates the inter-node communication arguments used when the leader of each node calls the inter-node MPI collective.

```java
function 
  if (UnknownGlobalPattern) then 
    ExchangeCnData(SharedCn, *Cn)
  else 
    Barrier(Comm)
  end if 
  if (rank == Leader) then 
    MPI_Collective(SharedBuff, *Cn, Leader Comm)
  else 
    SendBuff(InterCn, SharedCn)
  end if 
  end function 
```

Results

The following example demonstrates the communication pattern of the original AltoAvl versus when used in the Universal Hierarchical Algorithm. Solid lines are inter-node communication, dotted lines are intra-node communication.

### AltoAvl Example

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
</tr>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td>C0</td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
</tr>
</tbody>
</table>

### Hierarchical

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
</tr>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td>C0</td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
</tr>
</tbody>
</table>

### Process Imbalance Terminology

To characterize process imbalance the following terms are used:
- **Worst Case Imbalance** – This is the time difference of the first and last arriving process.
- **Average Imbalance** – This is the average distance of the process arrival times relative to the mean arrival time.
- **Average and Worst Case Factor** – The average and worst case imbalances divided by the minimal latency message on a given system.

The terminology and micro benchmark were introduced by Faraj et al. in “A study of process arrival patterns for MPI collective operations,” Int. J. of Parallel Programming, 2008.

### Imbalance Micro-benchmark

![Diagram of inter-node and intra-node communication patterns]

- **Setup Inter-Node**
- **Execution Inter-Node**
- **Execution Intra-Node**
- **Execution MPI**
- **Execution Binomial Tree**
- **Execution Universal**

### Imbalance Results

The average and worst case imbalance times for the micro benchmark on the Cray XE6 running 8192 cores.

![Heat map of normalized arrival times on a Cray XE6 running 2048 cores. Notice that specific cores to not consistently arrive late.]

### Future Work

This work presents the several optimized algorithms for MPI collective communications that improve both performance and flexibility. Future work aims to expand the imbalance aware algorithms to include a wider variety of collectives, as well as observe the effect these algorithms have when used in macro-benchmarks.