All-to-all Communication Algorithms for Distributed BLAS

Citation

Permanent link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:24829618

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story
The Harvard community has made this article openly available. Please share how this access benefits you. Submit a story.

Accessibility
All-to-all Communication Algorithms for Distributed BLAS

Kapil K. Mathur
S. Lennart Johnsson

TR-07-93
March 1993

Parallel Computing Research Group
Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts

All-to-all Communication Algorithms for Distributed BLAS

Dense Distributed Basic Linear Algebra Subroutine (DBLAS) algorithms based on all-to-all broadcast and all-to-all reduce are presented. For DBLAS, at each all-to-all step, it is necessary to know the data values and the indices of the data values as well. This is in contrast to the more traditional applications of all-to-all broadcast (such as a N-body solver) where the identity of the data values is not of much interest. Detailed schedules for all-to-all broadcast and reduction are given for the data motion of arrays mapped to the processing nodes of binary cube networks using binary encoding and binary-reflected Gray encoding. The algorithms compute the indices for the communicated data locally. No communication bandwidth is consumed for data array indices. For the Connection Machine system CM-200, Hamiltonian cycle based all-to-all communication algorithms improve the performance by a factor of two to ten over a combination of tree, butterfly network, and router based algorithms. The data rate achieved for all-to-all broadcast on a 256 node Connection Machine system CM-200 is 0.3 Gbytes/sec. The data motion rate for all-to-all broadcast, including the time for index computations and local data reordering, is about 2.8 Gbytes/sec for a 2048 node system. Excluding the time for index computation and local memory reordering the measured data motion rate for all-to-all broadcast is 5.6 Gbytes/s. On a Connection Machine system, CM-200, with 2048 processing nodes, the overall performance of the distributed matrix vector multiply (DGEMV) and vector matrix multiply (DGEMV with TRANS) is 10.5 Gflops/s and 13.7 Gflops/s respectively.

Kapil K. Mathur
and
S. Lennart Johnsson
Thinking Machines Corporation
Cambridge, MA 02142
All-to-all Communication Algorithms for Distributed BLAS

On distributed memory architectures, all-to-all broadcast and reduce form fundamental communication primitives. In an all-to-all broadcast, every processing node broadcasts its content to every other processing node in the system. In all-to-all reduce, reduction operations are performed concurrently on different data sets, each distributed over all processing nodes, such that the results of the different reductions are evenly distributed over all processing nodes. An all-to-all broadcast can be accomplished by each processing node sending its data to a dedicated processing node, either one at a time, or all at once, followed by a broadcast of the data from the dedicated processing node to all other processing nodes. All-to-all broadcast can also be realized by shifting data along a Hamiltonian cycle (ring of all processing nodes). For high degree networks, like binary cubes, this idea can be extended to the use of multiple Hamiltonian cycles that balances the communication load and maximizes the bandwidth utilization [2, 6]. All-to-all reduction is in effect the reverse operation of a broadcast, where combiners such as “+”, “max”, or “min” replace the copy operation. Figure 1 shows a simple example of all-to-all reduction. The left part of the figure shows the initial data distribution. After the all-to-all reduction operation, components with the same index are added together. The result consists of eight components distributed evenly across all processing nodes in a consecutive (block) manner. All processing nodes contain initial as well as final data.

Traditionally, all-to-all broadcast has been used to evaluate the pairwise interaction between all particles in direct N-body algorithms [1], [3], and [4]. An important feature to note is that these applications of all-to-all broadcast do not make use of the identity of the data value that is being received at a particular step of the all-to-all algorithm. This article illustrates the use of the all-to-all broadcast and reduce primitives for dense distributed basic linear algebra operations (DBLAS) such as matrix-vector, vector-matrix multiply, and rank-1 updates. These applications require not only the data values but their indices as well. Detailed schedules for all-to-all broadcast and reduction are described for the data motion of arrays mapped to the processing nodes of binary cube networks using binary encoding and binary-reflected Gray encoding. These algorithms compute the indices for the communicated data locally. Thus, no communication bandwidth is consumed for moving data array indices around.

Algorithms for all-to-all broadcast and reduction based on single and multiple Hamiltonian cycles in binary d-cubes and their implementation on a Connection Machine system, CM-200, are described. The performance of different implementations of the Hamiltonian cycles based algorithms is compared with the performance of all-to-all algorithms based on edge-disjoint, multiple spanning trees of minimum height, and the performance of butterfly network based algorithms.

These all-to-all algorithms have been incorporated in the distributed matrix-vector (DGEMV) and vector-matrix multiplication (DGEMV with TRANS) and rank-1 (DGER) update functions available in the Connection Machine Scientific Software Library, CMSSL [7], Version 3.0. A summary of the performance of the matrix-vector (DGEMV) and vector-matrix (DGEMV-TRANS) routines are given in Table 1, and in Figure 2.

<table>
<thead>
<tr>
<th>before reduction</th>
<th>after reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 P1 P2 P3</td>
<td>P0 P1 P2 P3</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 2 4 6</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 3 5 7</td>
</tr>
<tr>
<td>2 2 2 2</td>
<td></td>
</tr>
<tr>
<td>3 3 3 3</td>
<td></td>
</tr>
<tr>
<td>4 4 4 4</td>
<td></td>
</tr>
<tr>
<td>5 5 5 5</td>
<td></td>
</tr>
<tr>
<td>6 6 6 6</td>
<td></td>
</tr>
<tr>
<td>7 7 7 7</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: All-to-all reduction on a four processing node system.
Figure 2: Execution rate in Gflop/s for multiplication of a $P \times P$ matrix by a vector on different Connection Machine system, CM–200, configurations. 64-bit precision.

Table 1: Performance data for matrix–vector and vector–matrix multiplication on different Connection Machine system CM–200 configurations. 64-bit precision.
References


Kapil K. Mathur
and
S. Lennart Johnsson
Thinking Machines Corporation
Cambridge, MA 02142