A Technique for Improving Performance of Global Collective Operations on Cluster of SMPs

Benny N. Cheng
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, U.S.A. 91109

Abstract ***We describe a technique for speeding up the performance of global collective operations on a cluster of symmetric multiprocessor (SMP) parallel computer. Global collective operations are inherently faster within an SMP computer than between such computers. This algorithm takes advantage of this fact and performs the global collective operations first within the SMP machine, and then completes the operations between the machines. This results in significant improvement in global collective performance timing, almost twice as fast as conventional MPI global reduction calls in some cases.***

Keywords: SMP cluster,cluster computing,parallel programming

1 Introduction

When running parallel code on cluster of SMPs, traditional global collective or reduction operations such as global sums and multiply invariably runs into performance bottlenecks, due to the simple fact that the connection mechanism between SMP nodes are usually order of magnitude slower than those connecting processors within an SMP node. The standard binary tree reduction algorithms employed by popular communication architecture such as MPI are certainly not optimal for an SMP cluster, as we will soon see. However, with some relatively simple modification to the global collective algorithm, one can improve the performance of such operations by a significant amount, without having to rewrite the entire code to redistribute work among the processors.

Our approach is rather simple and involves two steps. Since reduction operations within a node is faster, the algorithm first completes the operations on all the processors within each node of the cluster, and then do a final reduction among the number of SMPs in use. The main challenge is to design a spin wait mechanism for synchronizing between the two steps mentioned above. This turns out not to be too complicated, and we have seen improvements of up to 50% in performance timing using our algorithm.

2 An Example: the HP X-Class Exemplar

At the Jet Propulsion Laboratory, our SMP cluster is an HP X-class Exemplar distributed memory computer which actually consist of 16 S-class SPP2000 hypernodes, connected by a coherent toroidal interconnect (CTI). Each SMP hypernode contain 16 processors, for a total of 256 processors. Published numbers for the Exemplar shows a peak bandwidth of 3.84 GB/s per link between SMPs, and a bandwidth of 15.36 GB/s within the SMP itself, about 4 times the speed difference[1]. The operating system runs on HPUX 10.01, with HP's implementation of MPI version 1.4. Timing performance comparison for a global sum operation using the MPI_allreduce call and our new

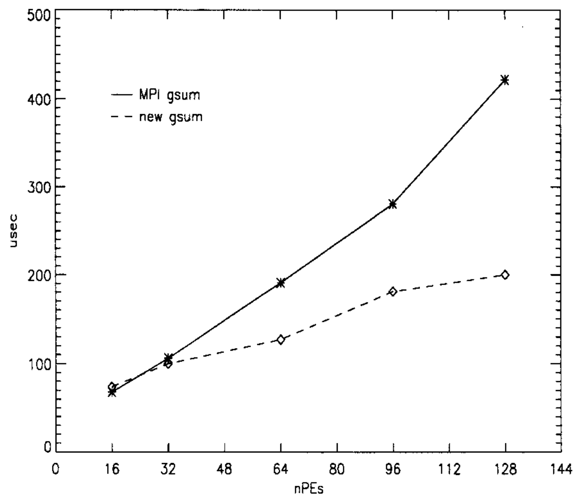

Figure 1: MPI vs new global sum performance

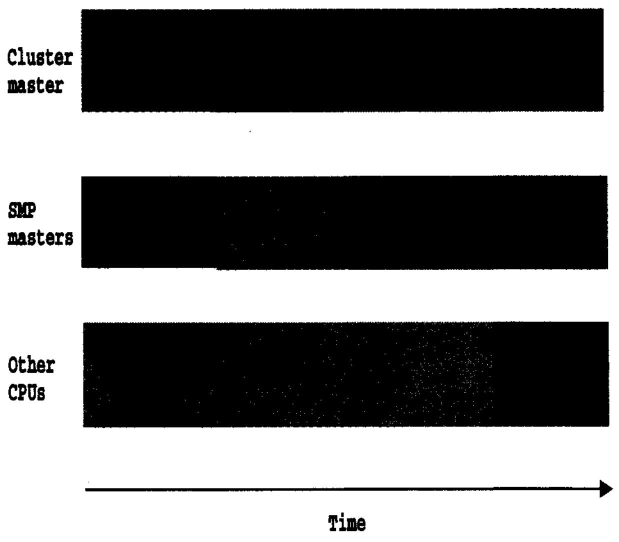

Figure 2: Block diagram of new gsum code

global sum code are shown in figure 1.

The figure indicates that a substantial improvement in global sum performance can be obtained beyond the confines of a hypernode. When running our ocean modeling code on this machine, the two-dimensional conjugate gradient solver mirrors the performance of the above global sums, hence benefits from the new global sum code with increasing number of processors.

3 The Algorithm

We illustrate our technique with the global sum operation, though in practice, the algorithm works for any global collective operation. For each SMP node, a region of (near) shared-memory is reserved, and declared to be accessible by any other node as far-shared memory. We implement this procedure using `mmap`, although there are several ways to do this, such as using `shmget`. The global sum is then computed as follows. Within each SMP node, we designate a randomly chosen CPU as the SMP master, which computes the sum over all elements within the node itself and stores it in it's local memory region. For the whole cluster, another CPU, randomly chosen from one of the SMP master CPUs, is designated as the cluster master. The cluster master spin-wait for all the SMP masters to complete their operations by polling their local shared-memory regions. When it detects that the SMP masters have computed their local sums, it then adds up the results of the SMP masters with itself, which now contains the global sum. In the meantime, all SMP masters spin-wait on the cluster master, and when the cluster master completes, the value of the global sum is then copied to each SMP's local memory. This is detected by all the other CPUs which copy the global sum from their respective SMP master. A block diagram of how the above procedure works is shown in figure 2.

4 Conclusion

Due to its cost and simplicity, clustering SMP computers is becoming a major design among supercomputer makers such as the HP Exemplar, and SGI Origin 2000. These SMP clusters can handle computation that run in parallel over hundreds, even thousands of processors. An inherent weakness that comes with the cluster design is that the communication architecture between the SMP computers are simply not as fast as those within the SMP computers.

Very often, they are simply high speed network interconnects, rather than the faster system buses that connect processors inside the SMP computer. This architecture naturally leads to performance bottlenecks for parallel codes that communicates with more than the number of processors in a single SMP computer. Standard message passing communication architecture such as MPI do not take into account the cluster design when performing global collective operations, for the simple reason of maintaining portability. However, we show that with a simple modification of the global collective operation algorithm, the performance of such operations on clusters can be improved significantly. It is definitely worth your while to implement such an algorithm when encountering time-critical projects that spend most of its time doing global collective operations.

Acknowledgement

This work is performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[1] Hewlett Packard Technical Report on High Performance Computing System, 1996.