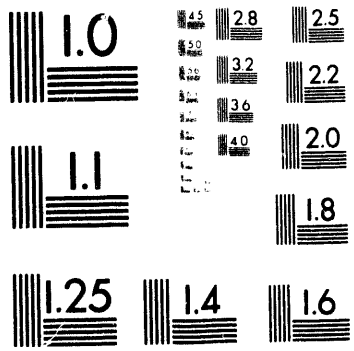
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

1 of 1

SAND93-2597C
Conf-9310232--1

Achieving High Performance on the Intel Paragon

David S. Greenberg * Barney Maccabe Rolf Riesen
Stephen Wheat David Womble

Abstract

When presented with a new supercomputer most users will first ask "How much faster will my applications run?" and then add a fearful "How much effort will it take me to convert to the new machine?" This paper describes some lessons we learned at Sandia while asking these questions about our new 1800+ node Intel Paragon. We conclude that the operating system is crucial to both achieving high performance and allowing easy conversion from previous parallel implementations to a new machine. Using the Sandia/UNM Operating System (SUNMOS) we were able to port a LU factorization of dense matrices from the nCUBE2 to the Paragon and achieve 92% scaled speed-up on 1024 nodes. Thus on a 44,000 by 44,000 matrix which had required over 10 hours on the previous machine we completed in less than 1/2 hour at a rate of over 40 GFLOPS. Two keys to our achieving such high performance were the small size of SUNMOS (less than 256kbytes) and the ability to send large messages with very low overhead.

1 How fast can it run?

During the summer and early fall of 1993 Sandia has been installing a very large Intel Paragon at its Massively Parallel Computing Research Laboratory. The machine consists of 31 cabinets each capable of holding 64 nodes. Over 1800 of the nodes are used for computation, 64 are used to control RAID disk controllers and the remaining serve as service nodes and ports for ethernet and HIPPI. The vast size of this machine promised potential peak performance of 140GFLOP. What performance could we achieve in practice?

As an indicator of performance we chose an LU factorization code which we had previously implemented on our 1024 node nCUBE2. LU is an important computation kernel for applications such as molecular dynamics, electrodynamics, and boundary element methods. It has the advantage of being highly computational intensive and thus can potentially make use of the peak 75MFLOP rating of the i860 nodes. However, it is not embarassingly parallel in the classical sense of having little or no communication. In fact, the LU requires the communication of very large messages with low overhead.

In this paper we report results only for a version of the LU code which was derived directly from the nCUBE code. No changes were made to the communication routines to optimize for the mesh topology (as opposed to the nCUBE's hypercube topology) and consequently the code could only run on a power of two number of nodes. Thus the largest number of

*Sandia National Laboratories Mail Stop 1423, P.O. Box 5800, Albuquerque, NM 87185-5800. Supported in part by the U.S. Department of Energy under contract DE-AC04-~~76DP00789~~. 94AL85000

computation could be cast in terms of the BLAS operations[2]. BLAS stands for Basic Linear Algebra Subroutines. They consist of a set of optimized assembly codes for matrix and vector calculations. As we will see below, the use of these subroutines was crucial for high performance.

Although the communication required by LU factorization does not grow as fast as the computation it is nonetheless significant. For an $n \times n$ matrix on p processors there are $O(n^2\sqrt{p})$ bytes of interprocessor communication. Most of this communication occurs during the broadcasting of rows or columns of the matrix. These broadcasts involve large amounts of data in a single message and thus efficient schemes for buffering the data are essential.

We are also interested in the effect of I/O on the ability of the Paragon to solve matrices which are too big to fit in the local memories. In previous implementations on the nCUBE2 we have seen that the number of I/O operations grows at best at the rate $O(\frac{n^3}{B\sqrt{M}})$ where B is the disk block size and M is the total size of memory. In fact a simpler to code algorithm in which the I/O grows as $O(\frac{n^4}{BM})$ may be preferable for realistic size problems. (For further details on I/O usage see [3].) Because of the large amount of local memory which SUNMOS left available for matrix entries we have not yet begun to look at I/O issues on the Paragon but expect to encounter them soon.

3 Local Memory size

The amount of memory available on each node of a parallel computer can have critical impact on performance. We have already noted that larger memory means that fewer I/O operations are required for a given sized problem. More importantly, if the memory if sufficiently large then no I/O is necessary during the computation.

SUNMOS requires less than 256 kbytes per node for the OS itself. In addition it manages all message buffers from a common pool. This means that buffers need not be assigned on a per sending node basis. Requiring buffers for each possible sender has catastrophic consequences when the machine is scaled to very large numbers of nodes. The default OS used per sender buffers in order to help prevent the user from creating message deadlocks. This is a worthy goal but was a price we were not willing to pay.

The net result of SUNMOS' frugal use of memory is that almost all of memory is available to the application. We were able to assign a 2000 x 2000 double precision submatrix to a 32Mbyte node - the matrix alone used 32 million bytes. SUNMOS, the program text, and all auxilliary variables and buffers fit in the difference between 32 Megabytes and 32 million bytes. On 1024 16Mbyte nodes we were able to fit a 44,016 x 44,016 entry matrix. This was larger than the largest problem we had been able to run *out of core* on our previous machine.

The large size of the submatrix assigned to each node gave us a large grained parallelism. Large grain size is the key to good scalability. Between each need for communication there was a large amount of computation to do. Further aiding our efforts was the interaction of memory size and BLAS performance. When larger blocks are fed to BLAS it is more efficient. Block size is increased either by increasing the size of the submatrix on a node (see Figure 1) or by increasing the BLAS block size (see Figure 2).

The BLAS block size requires some explanation. Our original implementation of LU factorization used BLAS level 1 vector-vector operations. However, the i860 processor requires high cache reuse in order to make full use of the arithmetic units. Level 1 operations use each data item once and yield almost 100% cache misses. It was essential to convert the

assigned to rows than to columns because this yields longer vectors for the BLAS1 routines such as the pivot searches. In general the communication is occurring fast enough to make tuning the communication portion of the code a second order effect.

Processors per column	Processors per row	MFLOPS
32	1	1044
16	2	1208
8	4	1297
4	8	1320
2	16	1321
1	32	1267

7776 x 7776 matrix on 32 nodes

Figure 3: Dependence on processor assignment

5 Performance under SUNMOS

Tuning the code for cache usage by BLAS and for low communication volume would not have resulted in high performance if the operating system were weak. We needed the OS to leave most of memory for application use and to provide high bandwidth, low startup communication. Fortunately SUNMOS does exactly this. As can be seen from Figure 4 we were able to achieve quite high scaled efficiency. The efficiencies reported in the figure are compared to the one node speeds. Even if we compare to the peak speed claimed by the authors of BLAS for the routines we used (46MFLOPS) the 1024 node run achieves more than 85% of the possible speed.

Proc	n	MFLOP/node	Total MFLOP	Efficiency
1	1376	42.8	42.8	-
4	2752	42.1	169	98.4
16	5504	41.4	662	96.7
64	11008	40.6	2604	94.9
256	22016	39.7	10156	92.8
1024	44032	39.4	40314	92.1

Figure 4: Scaled Speedup

6 Summary

We have shown that it is possible to achieve very high performance on the Intel Paragon. The most important factor was that we had access to the SUNMOS operating system which supplied us with fast message passing but did not require large amounts of memory. Running

DATE
FILMED
1 / 24 / 94
END