Message Passing and Advanced Computer Architectures

Brian Wainscott and Jason Wang
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, CA 94550

ABSTRACT

The emergence of distributed memory and cluster based computers is a recognized trend, with obvious cost/performance benefits. Here we discuss strategies for efficient utilization of these architectures on industrial problems. Results are presented which investigate the communication bandwidth and latency requirements for production environments.

INTRODUCTION

In the past decade, CPU performance has nearly doubled every 2 years while cost has remained constant. This is due to advanced RISC technology and increases in the CPU clock speed. It is not obvious that we will see the same leaps in CPU performance technology for the next decade, especially since the chip manufacturers are struggling to break the 1 GHz barrier.

At present, the most affordable solution for satisfying the growing demand for computing power is multiprocessor shared-memory computing, which is referred to as SMP. This technology has been used in the computing industry for many years. In the past, SMP was limited by several factors. The first is the fact that many processors have to share a single memory bus to access global memory. The second is that there are too many synchronization checks for parallel applications. These result in a computational bottleneck, which limits the scalability of SMP applications past 8 processors.

Given the constraint of current computer architectures, a cost effective solution to eliminating the SMP barrier would be to implement distributed computing across SMP clusters using a commodity network (e.g. 100 Mbit Ethernet). In this paper, we discuss the importance of network bandwidth and latency for LS-DYNA/MPP applications. We also discuss a new message-passing algorithm implemented in LS-DYNA/MPP that takes advantage of using SMP MPI protocol within the SMP box to reduce the amount of network contention and only use the TCP/IP MPI protocol between SMP boxes.

APPROACH

Bandwidth and Latency

The ideal parallel application is one in which there is no communication required between the separate tasks. Such an application will scale perfectly linearly to any number of processors and the performance is independent of the communication network. Unfortunately, this is not the case for applications as complex as LS-DYNA. The bandwidth and latency of communication does affect the performance of a given problem. *Latency* is the measured time it takes to send a message of 0 length between two processors. *Bandwidth* is the measured rate of data transfer, typically measured in megabytes per second, or equivalently, bytes per microsecond. For an application with few long messages, bandwidth is of primary importance. Conversely, for an application with many short messages, latency is more critical. We performed some experiments to determine the relative importance of bandwidth and latency in the case of LS-DYNA/MPP.

A car crash model having 128,000 elements was run on a 12-processor system. Details of the communication were captured for 10,000 cycles near the middle of the simulation. The results show that, on average, each processor sent 28 messages each cycle, the average length of which was only 1266 bytes. The vast majority, over 99%, were under 8K bytes long, while almost 67% were less than 1K long. There were also about 9 collective communication operations (vector summation and broadcast type operations) each cycle, most on the order of 50-100 bytes in length.

We measured the actual latency and bandwidth on a cluster of 12 workstations hooked up in two separate configurations. In the first case, which we will call configuration 1, a 100Mbit, half-duplex switch was used. In configuration 2, the switch was reset to 10Mbit, half-duplex. All hardware and cables were identical. The bandwidth and latency of the two configurations

are presented in Table 1. In the case of configuration 1, the latency is roughly equivalent to the time it would take 175 bytes to cross the network. In configuration 2, the latency is equivalent to increasing the message size by only 50 bytes. In either case, with an average message length of 1266 bytes, the communication time in our sample problem will be only 4% to 14% due to latency. The collective operations will be much harder hit due to their smaller data size. But in general, bandwidth appears to be more significant than latency.

Table 1: Latency and Bandwidth

	Latency (μ-sec)	Bandwidth (Mbytes/sec)
100 Mbit	95-120	1.75-1.78
10 Mbit	250-275	0.21
Ratio	~2.5	~8.3

This expectation was tested by timing two different problems on each of these machine configurations. Two models were chosen, a 28,000 element crash model, and the 128,000 element crash model. The timings for the runs are shown in Table 2. On the smaller model, the difference in speed was a factor of about 2.28. In the case of the larger model, the difference was a factor of 1.43. These numbers would seem to indicate that in fact the actual bandwidth is not driving the performance, in the sense that a factor of 8 difference in bandwidth does not produce a factor of 8 difference in speed. Then again, the difference in performance was lower even than the difference in the latency between the two configurations.

Table 2: 12 Processor Execution Speed

	28,000 elements (NZC)	128,000 elements (NZC)
100 Mbit	6990	2965
10 Mbit	15956	4239
Ratio	2.28	1.43

This is because the application doesn't spend all of its time communicating. For the 128,000 element problem, we can compute the expected difference in total communication speed based on the average message size. In configuration 1, a 1266 byte message should take 100+(1266/1.75) = 823 microseconds. In configuration 2, the expected time would be 250+(1266/0.21) = 6280 microseconds. Thus, the actual communication rate should be about 7.6 times slower in configuration 2. To result in an overall speed difference of 1.43, it must be the case that about 7.0% of the total runtime is spent in communication, when running with the hardware of configuration 1. For this particular problem, on this hardware, the actual speed of this problem is unknown on 1 processor, because the problem is too large to run on a single processor. The best estimate we have is that the 12-processor run was about 10 times faster than a 1 processor run. This would indicate that about 15% of the 12 processor run is due to communication (including the overhead of packing and unpacking data, and other algorithmic overhead) and load imbalance. Thus it is reasonable to believe that 7.0% of the run time is due to data actually crossing the network.

Clusters

Contention. Another factor that greatly influences parallel performance is contention during communication. When two or more processors are trying to send data through the same network path, contention occurs. In an IP based network of workstations, if two are trying to communicate across the same path, one of the messages is delayed while the other completes. Often, the delay is much longer than the actual time necessary to send the two messages in immediate succession, because the sender of the second message doesn't know exactly when

the first message will complete, so a of timeout factor is involved. Essentially, this amounts to an extra, semi-random contribution to communication latency.

Some hardware configurations suffer from this more than others. For example, in old style Ethernet networks, only one machine at a time can be sending on the local network. The same limitation holds for hub based networks where only one message at a time can cross the hub. For small numbers of processors, say 4 to 8 or 16, shared memory machines have a distinct advantage here. Not only do they typically have high bandwidth and low latency, but they are also much less likely to suffer from contention.

Workstation networks, when hooked up over a switch, have excellent connectivity – each processor has an I/O channel that it does not share with anyone. Clusters of SMP machines, on the other hand, can suffer from contention problems when all the processors need to communicate with other processors which are not available via the local shared memory. It appears that this can be a significant problem and a barrier to performance.

Multilevel Message Passing

In an attempt to address some of these issues for cluster based systems, a coordinated "gather/scatter" system of message passing has been designed, which we refer to as Multilevel Message Passing.

Consider a cluster machine consisting of four nodes (A, B, C, and D), each having 4 processors (A1, A2, etc.). LS-DYNA/MPP traditionally sees this as 16 individual processors. If each processor needs to share data with every other processor, 240 messages are sent. No coordination is done between A1 and A2, for example, to prevent contention for the data channels to nodes B, C, and D. With the Multilevel approach, processors A2, A3, and A4 will send to processor A1 all the data they have for all the processors in node B. A1 then delivers this data to B1, who distributes it to processors B2, B3, and B4. In this example, the 240 messages are reduced to 48+84=132 (3*16 for each processor to send to the 3 others in its node, plus 4*3*7: each node (4) sends to three other nodes (3), with a total of (7) messages each time). Best of all, of the original 240 messages, 192 of them passed between nodes, while 48 were internal to the nodes. With the Multilevel method, only 12 messages pass between nodes, with 120 being internal messages. This is a significant advantage, as communication between processors in a node is generally much faster than communication between nodes.

The approach is termed "Multilevel" because it generalizes to hypothetical machines in which there may be several layers of clustering. For example, several IBM SP2 machines (which are shared memory nodes connected via a fast switch) might be hooked together using 100Mbit Ethernet, resulting in three different levels of networking. The same approach can be used, with processors in each node gathering information (through shared memory) and passing it to a master node (via the high performance switch) on one machine, which passes it across the Ethernet.

Testing

To test this approach, LAM/MPI was installed on a pair of SGI Origin 2000 machines. Latency and bandwidth tests were performed to check the installation. For messages between processors within a node, latency was 10-15 microseconds, with bandwidth of about 15Mb/s. For messages between nodes, latency was 300 microseconds, and bandwidth was about 2.5 Mb/sec.

The 128,000 elements crash model was run on this system using two versions of LS-DYNA/MPP. In the first case, the old message passing method was used. In the second case,

the Multilevel method was used. The executables are otherwise identical. The problem was run for 5000 cycles, which is only about 4.6 milliseconds of simulation time. (Previous experience with this model indicates that the relative speed of the different runs out to 5000 is representative of what would be obtained for a full run of 100 milliseconds.) Speed is measured in nanoseconds per zone cycle (NZC), which is computed as nanoseconds of CPU time divided by the product of the number of time step cycles and the number of elements in the problem. For a fixed problem and a fixed number of cycles, NZC is exactly proportional to actual run time. Table 3 summarizes the results.

Table 3: Multilevel Performance

	Standard Message Passing	Multilevel Message Passing	Speedup
1 Processor	36677	---	0%
4 Processors	11590	11547	0.4%
8 Processors	5161	5134	0.5%
12 Processors	3971	3825	3.8%
14 Processors	3544	3475	2.0%

As expected, the Multilevel approach is faster than the old method. The benefit is insignificant for small numbers of processors, but is worthwhile for 12 processors and it must be admitted that the results are less than expected based on the above discussion.

DISCUSSION OF RESULTS

Analysis of test results

There are several reasons for the observed level of speed increase. The discussion above neglected to account for the overhead required in packing and unpacking the data to be delivered. The extra complexity of the Multilevel method causes it to have more of this kind of overhead. Further, in the case of LS-DYNA/MPP, each processor generally only shares data with *some* of the other processors, not all of them. There is a great deal of spatial and topological locality to the data, so that in general each processor will share the vast majority of its data with only a few other processors. The existing message passing style is thus not so bad as we had imagined. Finally, the existing contact algorithms in LS-DYNA/MPP have some special message ordering optimizations in them, to minimize the amount of time processors spend waiting on each other. An optimization of this type can be done in the Multilevel case, but it is more complex and has not yet been implemented.

Even so, a speedup of 4% is not insignificant. In this case, we can run this problem on 1 processor and so we know the actual speedup from 1 to 12, with the original algorithm, is 9.24. Thus each of the 12 processors spent 77% of their time doing useful work and 23% was lost to communication, load imbalance, and algorithmic overhead associated with communication. The load imbalance will be unaffected by changing to the Multilevel method, but the actual amount of imbalance is unknown. Even so, a reduction of 4% in total runtime corresponds to a substantial reduction in the communication portion of the total overhead.

Furthermore, it is expected that on larger machines, such as an 8 way cluster of 8 processor nodes, that the Multilevel method could have a greater impact.

CONCLUSIONS

The past few years have seen a growing market for LS-DYNA/MPP. With the emergence of large distributed memory computers, clusters of SMP machines, and faster local area networks, the potential for running large problems has grown tremendously. As the hardware on which we run problems evolves, it is critical that we continue to explore new algorithms and strategies for fully utilizing these new systems. As problem sizes continue to grow, and problems are run on increasingly large numbers of processors, low latency/high bandwidth communications systems will become more important than ever. The potential of the Multilevel communication strategy investigated here, demonstrates that continued research and application of new strategies will continue to push the performance of LS-DYNA/MPP to new levels.