Performance Analysis and Tuning of LS-DYNA* for Intel® Processor-Based Clusters

George Chaltas
Intel Corporation, DP2-230
2800 Center Dr.
DuPont, WA 98327
253-371-7168
George.Chaltas@intel.com

W. R. Magro
Intel Americas, Inc.
1906 Fox Drive
Champaign, IL 61820
217-356-2288
Bill.Magro@intel.com

Abbreviations

MPI: Message Passing Interface
CPU: Central Processing Unit
SIMD: Single Instruction Multiple Data
SSE: Streaming SIMD Extensions
IBA: InfiniBand Architecture
VIA: Virtual Interface Architecture
VIPL: Virtual Interface Provider Library

Keywords

Cluster, Performance, Intel, MPP-DYNA

ABSTRACT

Using Intel software tools, including Intel® VTune™ Performance Analyzer and Intel® Fortran Compiler, we analyze and tune the performance of MPP LS-DYNA* for clusters of Intel processors. We discuss the impact of various performance features of Intel processor-based systems, including vector/streaming instructions, on real LS-DYNA workloads. We compare single-precision performance and measure the impact of various cluster interconnect technologies.

INTRODUCTION

Clusters of relatively inexpensive, general-purpose computers are now able to perform tasks that once required specialized (and expensive) hardware. Using software such as the MPI implementation of LS-DYNA, a modest cluster of Intel® Xeon™ processors can perform analyses both more quickly and more cost effectively than the supercomputers that were until recently used for this purpose. Apart from faster CPUs, the main opportunities for improvement lie in the cluster interconnect technology (both hardware and software) and in the performance of the software running on each node. We have investigated both areas, and the latest versions of LS-DYNA now incorporate some improvements resulting from these investigations.

Conventions

Within this paper, we have adopted the following conventions. Unless otherwise stated, all software commands and command-line options are for software running on the Red Hat* Linux operating system. Performance is measured as elapsed (wall-clock) time to complete a workload, however elapsed times are not stated. The performance of systems and software are compared and expressed as ratios. The configurations under consideration are stated along with the performance data. While most of the workloads used for performance measurement and analysis are publicly available, others are not. The workloads used are described in the appendix.

Benchmark Disclaimer

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel® products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing.

Because LS-DYNA is a large, complex application with a broad range of features, the measurements of performance enhancements discussed in this paper are specific to the workloads used.

APPROACH

In mid-2001, we became aware that the performance of a cluster of Intel® Xeon™ processor-based systems running LS-DYNA (v940.2a, single-precision), while good, did not meet our expectations. Accordingly, we undertook an analysis of the problem and worked with engineers at LSTC to effect improvements. We investigated both single-node and cluster performance. Single-node performance improvements have focused primarily on the compilation of LS-DYNA*; cluster performance has been investigated within the context of MPI implementations and interconnect technology. We worked in both the Linux* and Microsoft Windows* environments, but focused primarily on the former.

SINGLE-NODE PERFORMANCE

Performance of clusters of Intel® Pentium® III processors and Pentium® III Xeon™ processors running LS-DYNA was generally considered to be good, but Intel® Xeon™ processors and Intel® Pentium® 4 processors did not meet our expectations for relative performance on systems with these faster processors and their increased memory bandwidth. Important differences exist between the micro-architectures of these processor families; accordingly, our investigation focused on these differences. The primary tools for addressing such architectural differences are compilers. Intel has developed a family of compilers (available for both the Linux and Microsoft Windows* operating systems) for precisely this reason; these compilers are aware of the performance characteristics of broad range of Intel processors and produce code specifically tuned for them. Since we desired that our work ultimately result in performance enhancements to the LS-DYNA product, we placed the following constraints upon this effort: first, the resulting binary should run well on a broad range of processors; second, only minimal source code modifications would be considered. To include the overhead of the MPI library, but avoid any affects due to cluster interconnects, we used the MPI version for this testing and analysis, but confined our tests to a single node with two CPUs.

Features of Intel® Pentium® 4 processors and Intel® Xeon™ processors

Intel® Pentium® 4 processors and Intel® Xeon™ processors share a common microarchitecture (called Intel® NetBurst™) and a similar feature set. The primary difference is that Intel® Xeon™ processors are intended for multiple-CPU systems, whereas Intel® Pentium® 4 processors are designed for single-CPU systems. The processors have similar performance characteristics, and the performance work described in this paper applies to both processor families. The following are some common characteristics of the current generation of these processors[1]:

- Intel® NetBurst™ microarchitecture
- 400 MHz System Bus
- Level 1 Execution Trace Cache
- 8 KB Level 1 Data cache
- 512 kB Level 2 Advanced Transfer Cache[2] - 8-way set associative, 128 Byte lines
- Streaming SIMD Extensions 2 (SSE2) [superset of SSE]

SSE2 instructions extend the SSE instructions implemented in the Intel® Pentium® III processor. They incorporate single-precision floating-point vectors (length 4), double-precision floating-point vectors (length 2), and integer vectors of various lengths. The aggregate vector length for all data types is 16 bytes, corresponding to the size of the registers used to implement these instructions. For more information on these instructions see IA-32 Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture.

Targeting Intel® NetBurst™ Microarchitecture processors

The Intel® Fortran Compiler is able to generate SSE2 instructions, and effectively using these vector instructions was critical to success. Since we focused on the single-precision version of LS-DYNA* and wished to maintain binary compatibility with Intel® Pentium® III processors, we constrained our work to the SSE subset. The compiler is also able to optimize its code scheduling for particular processor families while maintaining compatibility with others. We used this feature to target the Intel® Pentium® 4 and Xeon™ processors. The compiler options used to accomplish this are as follows:

- -xK : use SSE instructions (compatible with Intel® Pentium® III or later processors)
- -tpp7 : optimize instruction generation for Intel® Pentium® 4 or Xeon™ processors.

Using these options with the Intel® Fortran Compiler 5.0 for Linux*, we were able to achieve a modest improvement in LS-DYNA 940.2 performance, in the range of 6% to 9%. This gain was less than we expected, but suggested that we were on the right path. Because LS-DYNA version 960 was nearing release, we refocused our efforts on that version.

Hotspot Analysis

The next step was to use the Linux hierarchical profiler, `gprof`, to identify the parts of LS-DYNA 960 that were taking the most time. From its output, we determined that over 60% of the run time in our Small Car workload was spent in only six functions:

Table 1: Function Hotspots

Function Name	%Time
shl24s	16.24
trnfbt	14.32
tranbt	11.58
blytsy	10.35
elem2d	6.96
tbscls	5.52

[1] A broader overview of processor features may be found in Intel® Xeon™ Processor with 512 KB L2 Cache for Applied Computing Product Brief.

[2] Earlier processors in this family have 256 kB Level 2 cache

Repeating the analysis with a broader set of workloads confirmed the importance of these functions and identified a few more. By comparing this data with profiling data gathered on other systems, we determined that several of these functions were running approximately half as fast as we might expect.

Root Cause Analysis

To systematically identify and correct the performance problems, we turned to the Intel® VTune™ Performance Analyzer (hereafter referred to as VTune™). Intel® Pentium® 4 and Xeon™ processors incorporate programmable hardware performance counters that can be used to monitor a variety of events, from clockticks (the processor's internal clock) to cache behavior. Using VTune, occurrence of these events can be correlated closely to small groups of instructions in program code. If the executable contains debug information, VTune is also able to map these events to specific areas of program source code. VTune enabled us to quickly identify not only the functions that took the most time, but also the responsible lines of code within these functions. Once problem areas were identified, we began the search for causes. This search was fairly extensive, but focused initially on the following kinds of events:

- Mispredicted branches
- Level 2 cache misses
- Level 1 data cache misses
- Trace cache misses
- Store-forward
- Split cache lines
- 64K aliasing

These events – and techniques to ameliorate their effects – are described in the Intel® Pentium® 4 and Xeon™ Processor Optimization Reference Manual,

To evaluate the effect of these events, we used VTune™ to map both the occurrence of clockticks and the events under consideration to specific areas of code. Areas where the elapsed time (clocktick count) is disproportionately high can be performance problems or simply areas of code that execute frequently. Potential causes of problems are suggested by corresponding high values in the event counters listed above. For example, a disproportionate number of L2 cache misses per clock tick would suggest a cache utilization problem. While all of these events occur to some extent in LS-DYNA* (and indeed in nearly all programs) VTune showed that most of them did not occur with undue frequency, if at all, in the problem areas of LS-DYNA.

Thermal Throttling

Intel® Pentium® 4 and Xeon™ processors have the ability to reduce their clock speed to prevent overheating, resuming normal operation once they are sufficiently cooled. Although this thermal throttling should not occur in a properly cooled system, we checked for and ruled out its presence. VTune has no mechanism to check for thermal throttling, so we used a special tool (not generally available) to determine that thermal throttling was not taking place.

Vectorization

Since the –xK compiler option enables the compiler's vectorizer, we checked to determine how effectively the vector SSE instructions were being used, using VTune to count the relative number of vector SSE, scalar SSE, and X87 instructions retired. Only vector SSE instructions lead to improved throughput. We discovered that the slowest parts of the functions identified above made little if any use of these vector instructions. We focused on the slowest, most time-consuming loop and observed that it performed substantial data movement. Each iteration, read from several arrays, calculated a few intermediate results, and stored results to 18 separate arrays. Because the loop bounds were passed as parameters to the function and the number of arrays accessed was relatively large, the compiler was unable to determine if this loop would perform well if vectorized and so declined to do so.

Write Combining

Examining the memory addresses in a debugger revealed that each of the target addresses stored to in one loop iteration lay on a different cache line. Intel® Pentium® 4 and Xeon™ processors perform write combining in hardware, aggregating a number of small, contiguous writes into a single, larger write to make better use of cache bandwidth. Non-contiguous storage patterns such as this can preclude write combining, as the Intel® NetBurst™ microarchitecture maintains only six buffers for write combining and a large number of target addresses results in frequent flushing of these buffers. The Intel® Pentium® 4 and Xeon™ Processor Optimization Reference Manual recommends applying loop fission in such cases, but vectorization can also help. Scalar single-precision floating-point instructions can store or load only four bytes at a time, whereas vector instructions can load or store 16 bytes. Using vector instructions effectively reduces the number of store operations by a factor of four, making more effective use of write combining and available cache bandwidth.

Solution

Manually splitting many loops would violate our goal of minimizing source code modifications, so we applied it on only the most critical location and focused instead on increased vectorization as the primary mechanism for improving performance. The Intel® Fortran Compiler can report which loops it vectorizes, which it does not, and why. Enabling this report, via the –vec_report3 compiler option, indicated that the compiler believed that vectorization of these key loops would be inefficient. Fortunately, the compiler supports a directive (!DIR$VECTOR ALWAYS) to override this heuristic. After applying this directive, we observed a roughly four-fold increase in the performance of the most critical loop. We reproduced the behavior in a separate test case, to ensure the future releases of the compiler will be able to identify this optimization without the use of directives. Additional vectorization yielded similar improvements. The resulting improvement in LS-DYNA* performance on our test cases is shown below.

Cache Blocking

With the main performance problem resolved, we turned our attention to cache blocking. LS-DYNA blocks its data, gathering it into several contiguous arrays of size *N* to achieve better cache utilization and facilitate vectorization. Cache size is a consideration; it is highly advantageous for all the blocked data to fit within the L2 cache. Within that constraint, it is advantageous to use the largest possible blocking size. The latest Intel® Xeon™ processors (and Intel® Pentium® III Xeon™ Processors) have 512KB L2 cache, twice that of their predecessor. Accordingly, we tuned the cache blocking for this size cache. For some workloads, this may result in slight performance degradation on earlier processors with smaller cache. We also tuned cache blocking to improve utilization of the processor's floating-point vector instructions. We tested a broad range of blocking sizes and observed that the best performance was achieved when the array size, *N*, was an integer multiple of the SIMD vector size (four).

Results

The results of this optimization work were incorporated in LS-DYNA 960 build 1145, which was built with the Intel® Fortran Compiler V6.0. To illustrate the difference, we compared this with build LS-DYNA 960 build 447, which was built with an earlier version of the Intel® Fortran Compiler and without any of the tuning described above. The tables below illustrate the relative performance of these two builds, on systems with pairs of three different Intel® Xeon™ processors. Numbers greater than 1.0 indicate increased performance of LS-DYNA 960 build 1145 over LS-DYNA 960 build 447.

The latest Intel® Xeon™ processors were the primary focus of our optimization efforts. Cache blocking in build 1145 was tuned for the 512KB L2 cache size of these processors

Table 2: Relative Performance
LS-DYNA* 960 447 P4 vs. LS-DYNA* 960 1145 SSE
Two 2.2 GHz Xeon™ Processors (Configuration A)

Small Car A	Caravan	Bogie20	Pendulum	PCB	rp_lsd93	Small Car B
1.31	1.20	1.14	1.01	1.25	1.19	1.45

Since a specific goal of this effort was to maintain compatibility with Intel® Pentium® III processors, we evaluated the two builds of LS-DYNA on a system equipped with two 1.4 GHz Intel® Pentium® III Xeon™ processors. These processors also have 512KB L2 cache.

Table 3 Relative Performance
LS-DYNA* 960 447 vs. LS-DYNA* 960 1145 SSE
Two 1.4 GHz Intel® Pentium® III Xeon™ Processors (Configuration C)

Small Car A	Caravan	Bogie20	Pendulum	PCB	rp_lsd93	Small Car B
1.09	1.13	1.11	1.04	1.05	1.06	1.34

Finally, since this effort was specifically focused on improving LS-DYNA performance on Intel® Xeon™ processors, it is interesting to compare the performance measured at the beginning of this effort (LS-DYNA 960 447, 1.7 GHz Intel® Xeon™ processors) against the performance measured at the conclusion (LS-DYNA 960 1145, 2.2 GHz Intel® Xeon™ processors). The combination of updated software and faster processors demonstrates improvement well above that expected from clock speed improvements alone (2.2 / 1.7 = 1.294) across all six tested workloads.

Table 4: Overall Change in Performance
Two 1.7 GHz Intel® Xeon™ Processors (Configuration B), LS-DYNA 960 447 P4
Two 2.2 GHz Intel® Xeon™ Processors (Configuration A), LS-DYNA 960 1145 SSE

Small Car A	Caravan	Bogie20	Pendulum	PCB	rp_lsd93	Small Car B
1.77	1.71	1.52	1.41	1.68	1.84	1.96

Table 5: System Configurations

Configuration A
Two 2.2 GHz Intel ® Xeon™ processors
512 KB L2 cache
1 GB 800MHz RDRAM
IWILL* DX400-SN motherboard
Intel® 860 chipset
18 GB SCSI-LVD Disk
Red Hat* Linux 7.2
LAM MPI 6.5.2
Configuration B
Two 1.7 GHz Intel® Xeon™ processors
256KB L2 cache
1 GB 800MHz RDRAM
IWILL* DX400-SN motherboard
Intel® 860 chipset
18 GB SCSI-LVD Disk
Red Hat* Linux 7.2
LAM MPI 6.5.2
Configuration C
Two 1.4 GHz Intel® Pentium® III processors
512KB L2 cache
1 GB 133MHz SDRAM
Intel® Server Board SCB2
ServerWorks* Enterprise Serverset* III HE-SL chipset
18 GB SCSI-LVD Disk
Red Hat* Linux 7.2
LAM MPI 6.5.2

Cluster Scaling

The MPP version of LS-DYNA* is capable of exploiting clusters of connected systems to improve turnaround time for suitably large analyses. The speed of the cluster interconnect – normally characterized by its latency and bandwidth – largely dictates the extent to which the completion time of a given workload can be further decreased by utilizing additional processors. Low latency, the time to send an empty message from one host to another in the cluster, and high bandwidth, the asymptotic transfer rate for large messages, are characteristics normally found in a high-performance interconnect.

Interconnects for Message Passing

Users of MPP-DYNA* on clusters benefit from high-speed interconnects through shortened turnaround time on jobs. While a number of proprietary interconnects are available that each handily outperform standard Ethernet, no single interconnect has emerged as standard. As a result, many clusters are simply connected using standard 100BaseT Ethernet*.

While the performance of the proprietary interconnects is often excellent, there exist some drawbacks in their use for both software vendors and end users. Normally, each interconnect is accompanied by its own MPI libraries, forcing software vendors unable to support all available interconnects to choose a subset, leaving some unsupported. Worse, the myriad choices drive some vendors to support only the least common denominator, MPI over TCP/IP, to achieve broader hardware support at the expense of performance. Likewise, end users suffer when their chosen interconnect is supported by only a subset of the software packages needed in their work. Proprietary interconnects have other disadvantages, including limitations on simultaneous multi-user access or limited ability to simultaneously handle MPI, TCP/IP, and NFS or other shared I/O traffic.

InfiniBand Architecture

A new industry standard interconnect architecture, InfiniBand Architecture* (IBA), has several key features that give it the potential to address these shortcomings. Most important among these is IBA's notion of virtual channels, which presents to each process the illusion of a dedicated interconnect. This, when paired with IBA's low latency, high bandwidth design, allows multiple users and the operating system to simultaneously and transparently share a single interconnect. On IBA-connected systems, networking, message passing, and shared disk I/O traffic run over a single wire. Further, because IBA is a multi-vendor, industry standard, host adapters and switches from multiple sources interoperate, presenting a single and common high performance interconnect for end users to install and software vendors to support.

Message Passing over InfiniBand Architecture

To demonstrate the impact of a fast interconnect on LS-DYNA* performance, we measured the parallel speedup of a small workload on an cluster connected with both 100BaseT switched Ethernet and Intel InfiniBand Architecture host channel adapters. We used Argonne's MPIch implementation with NCSA's VMI 1.0 abstract device implementation. VMI differs from most MPIch devices in its ability to target multiple interconnects without rebuilding the application. This approach offers a key advantage for software vendors, since a single validated executable can target both standard and proprietary interconnects, while maintaining the performance characteristics of each. Rather than hard code the communications library, VMI instead implements a thin communication wrapper layer that determines the fastest interconnect between each pair of processes and automatically loads shared libraries to support that connection. VMI provides communication libraries for TCP/IP, shared memory, and a number or high-speed interconnects, including InfiniBand adapters via the Intel™ Virtual Interface Provider Library (VIPL)[3]. New communication libraries are simple to write. For this work, we used shared memory to communicate within each dual-processor node and compared TCP/IP to InfiniBand for inter-node communication.

[3] These tests were performed on Microsoft Windows 2000 Advanced Server, due to the earlier availability of IBA VIPL libraries on this platform. VIPL is now available for both Windows and Linux.

Point-to-Point Latency and Bandwidth

We began by measuring the pair wise, inter-node performance of TCP/IP over 100BaseT and VIPL over IBA, using a standard "ping-pong" test, in which a message is sent from one host to another, then echoed back to the original host. The total time is measured for a range of message sizes, and half the round-trip time is taken as the one-way measurement. The results are presented in Figure 1. The y-intercept of the curves is the latency, the time to send an empty message, while the asymptotic slope is inversely related to the bandwidth rate for large messages. A flatter curve with smaller intercept indicates better performance. The VIPL/IBA device achieves a latency of less than 17 microseconds, compared with 152 microseconds TCP/IP over 100BaseT Ethernet. The VIPL/IBA device also achieves better performance for large messages, achieving over 135 MB/s compared with 9.5 MB/s for Ethernet.[4]

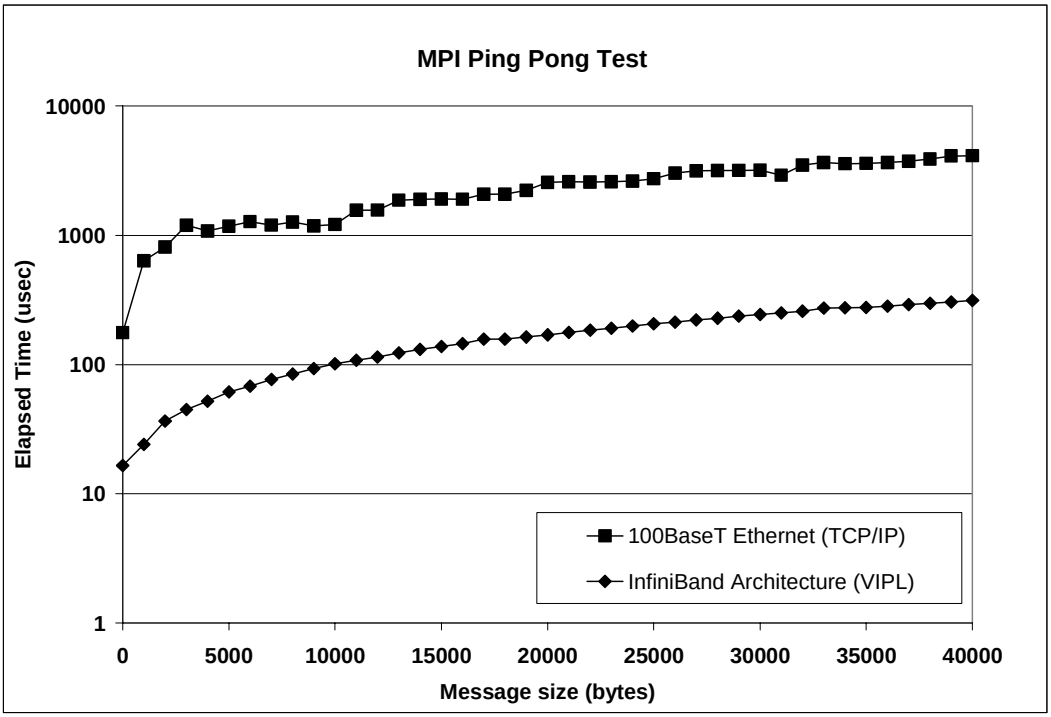

Figure 1 : Comparative performance of point-to-point messaging over Ethernet and InfiniBand Architecture interconnects. Ethernet performance was measured using TCP/IP as the connection transport through a dedicated 100BaseT switch. InfiniBand* Architecture performance was measured using VIPL as the connection transport through a dedicated InfiniBand Architecture switch. The *y*-intercept is the zero message size latency. Lower is better. The slope (when plotted linearly) is the inverse bandwidth. Flatter is better.

Scaling Results

This point-to-point message performance is expected to translate into improved parallel application speedup in LS-DYNA*, so we next measured the completion times of a small workload. We intentionally chose a small problem to illustrate the impact of interconnect performance on speedup. The rp_lsd93 model represents the impact of a small car with a rigid pole, but it has only 5,400 elements. For this test, we again measured the performance of TCP/IP over 100BaseT and VIPL over IBA. The performance results are illustrated in Figure 2, where we see that 100BaseT Ethernet achieves speedup on two and four processors. For more processors, the elapsed time actually begins to increase, as the inter-node communication time begins to dominate the computation. The much faster messaging performance of IBA, however, shows continued performance improvements up to 14 processors, which is perhaps surprising, given the small size of the workload. To achieve these results, we used the default domain decomposition

[4] An InfiniBand 1X link operates at a line speed of 2.5 Gbits/second in each direction. Consequently, throughput of as much as 300 MB/second is possible. The result of 135 MB/sec was achieved with first generation hardware from Intel's InfiniBand Product Development Kit. Future hardware is expected to deliver significant improvements in bandwidth. Achieving the full line speed is not possible, due in part to MPI's two-sided communication protocol, which requires at least one extra message or one buffer copy to deliver a message. The InfiniBand specification also defines 4X and 12 X connections, which provide four and twelve times the bandwidth of a 1 X, link, respectively.

settings, but supplied the "pfile" option to specify a local directory on each node for scratch space. Writing scratch files to a shared file system normally decreases parallel performance, particularly for the large cluster configurations.

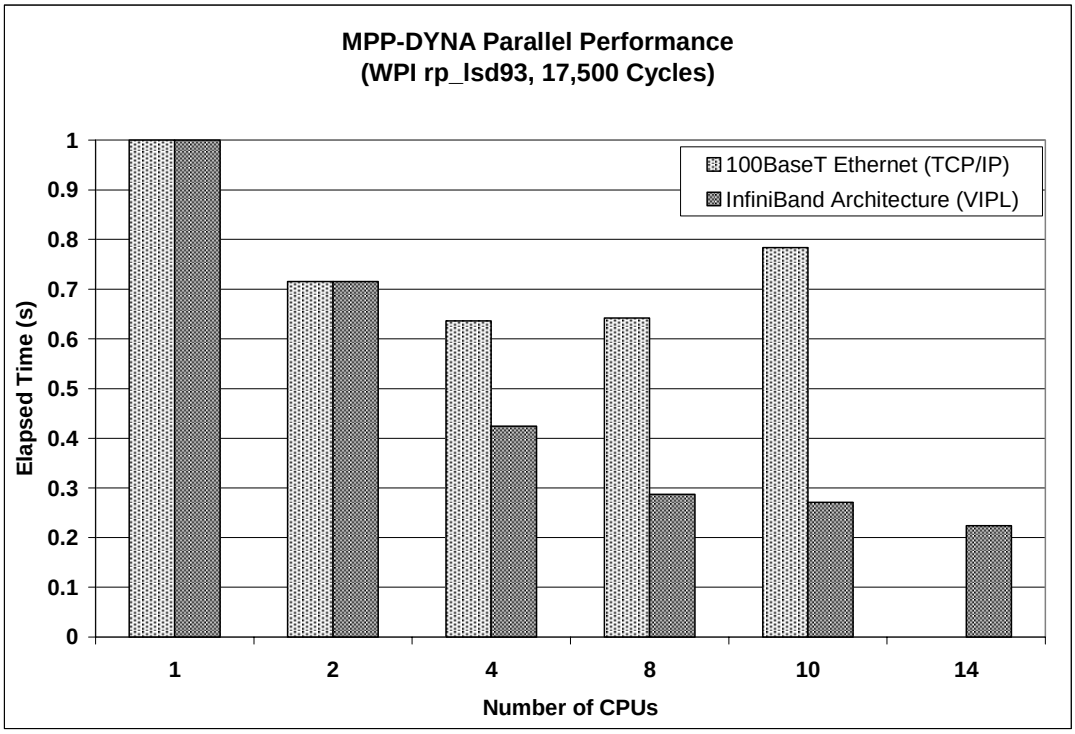

Figure 2 : Comparative speedups of the WPI rp_lsd93 small car crash test case over Ethernet* and InfiniBand* Architecture interconnects. The network configuration was the same as used in the point-to-point measurements in Figure 1. The bars represent relative elapsed time to complete 17,500 cycles. The Ethernet-connected cluster peaks in performance at 4 CPUs (two nodes); beyond 4 CPUs, the calculation runs more slowly. The InfiniBand Architecture-connected cluster maintains speedup through 14 CPUs, due to the lower latency and higher bandwidth of the interconnect.

These performance results show that InfiniBand Architecture is a viable interconnect technology for high performance computing applications such as MPP-DYNA*.

SUMMARY

Using Intel® software tools, we have analyzed and optimized the performance of LS-DYNA on Intel® Pentium ® 4 and Xeon ™ processor-based systems. These performance improvements have been incorporated into LS-DYNA 960 build 1145.

We have measured the impact of an InfiniBand Architecture interconnect on the scaling behavior of MPP-DYNA. The low latency and high bandwidth of the interconnect allow even small problems to achieve speedups to 14 CPUs, while 100BaseT Ethernet peaks at 4 CPUs.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the following individuals. Without their support and assistance our work would not have been possible:

Drs. John Hallquist, Jason Wang, and Chen Tsay, Livermore Software Technology Corporation
Rob Pennington, Mike Showerman, and Avneesh Pant, National Center for Supercomputing Applications, University of Illinois.

Clay Breshears, Thomas Huff, Timothy Prince, Kevin B. Smith, and Sreelekshmy Syamalakumari, Intel Corporation

REFERENCES

Intel® Pentium® 4 and Xeon™ Processor Optimization Reference Manual, # 248966-05, © 1999-2002 Intel Corporation.

IA-32 Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture, #245470-006, © 1997-2002 Intel Corporation.

Intel® Pentium® 4 Processors for Applied Computing Product Brief, #273657-01, © 2002 Intel Corporation.

Intel® Xeon™ Processor with 512 KB L2 Cache for Applied Computing Product Brief, # 273697-01, © 2002 Intel Corporation.

Intel® Fortran Programmer's Reference Manual, #687928-5001, © 1996-2001 Intel Corporation

A.J.C. Bik, M. Girkar P.M. Grey and X. Tian. Automatic Intra-Register Vectorization for the Intel(R) Architecture. International Journal of Parallel Programming, Volume 30, pages 65--98, April 2002.

A.J.C. Bik, M. Girkar, P.M. Grey, and X. Tian. Efficient Exploitation of Parallelism on Pentium(R) III and Pentium(R) 4 Processor-Based Systems. Intel Technology Journal, February, 2001.

InfiniBand Trade Association, www.infinibandta.org

Workloads

The following table summarizes the data sets and parameters used in the performance analyses described in this paper.

Table 6: Workload Descriptions

Name	Element Count	ncycles	Source	Description
Small Car A	430000	Complete	Customer data	A small car striking a rigid barrier, simulation time .010 sec.
Caravan	329300	5000	NCAC	Dodge* Caravan*, detailed model
Small Car B	530000	Complete	Customer data	A small car striking a rigid barrier, simulation time .030 sec.
Bogie20	1800	80000	NCAC	FOIL* Bogie, 20 MPH
Pendulum	2100	Complete	NCAC	FOIL* pendulum
PCB	23300	60000	NCAC	Portable Concrete Barrier
rp_lsd93	5400	Complete[5]	WPI	Small car rigid pole impact

NCAC: National Crash Analysis Center at George Washington University, http://www.ncac.gwu.edu/archives/model/

WPI: Worcester Polytechnic Institute, http://www.wpi.edu/Academics/Depts/CEE/Roadsafe/bench.html

Software Tools

Intel® VTune™ Performance Analyzer 6.0
Intel® Fortran Compiler for Linux*, 5.01 and 6.0
Available at http://www.intel.com/software/products/index.htm

[5] For the scaling studies, the model was terminated after 17,500 cycles.