Assessment of LS-DYNA Scalability Performance on Cray XD1

Author:

Ting-Ting Zhu, Cray Inc.

Correspondence:

Telephone: 651-605-9087

Fax: 651-605-9123

Email: tingting@cray.com

Keywords:

Parallel performance

MPI library

Cray XD1

Opteron cluster

RapidArray interconnect

Myrinet Interconnect

LS-DYNA

ABSTRACT

Technologists worldwide have now recognized that CAE (Computer Aided Engineering) offers an unprecedented opportunity to revolutionize product development. Today, CAE promises not only increased productivity but also faster time-to-market, lower warranty costs and above all, products that are safer, outperform and work better.

With the broad acceptance of the MPI based implementation of LS-DYNA, the manufacturing industry is pushing the limits of scalability as they scramble to meet stringent product design cycle constraints. Microprocessor based cluster systems are increasingly being deployed for production workloads. But, the scalability and system efficiency can be very poor on such systems. The industry goal to reduce time-to-market can be met only if the system has a balanced architecture and the interconnect technology is able to deliver sustained performance for actual applications.

In this study, an in-depth analysis will be performed to assess the performance of LS-DYNA on Cray's XD1 system. A correlation between the hardware features of Cray XD1 and the attributes of LS-DYNA will be made. Various phases involved in a typical crash simulation, such as - initialization, element processing, contact and rigid bodies calculations will be analyzed. An MPI profiling tool will be used to monitor the MPI performance in the context of computation, communication and synchronization aspects of LS-DYNA. The communication patterns and message sizes will be studied for variety of standard benchmarks (www.topcrunch.org). The role of Cray XD1's balanced architecture and the high speed interconnect technology will be presented in the specific context of LS-DYNA and production workloads. Performance results of LS-DYNA on Cray XD1 will be highlighted that truly demonstrate "Application efficiency at scale".

INTRODUCTION

The automotive manufactures have started to rely on MPP version of LS-DYNA for crash analysis and metal forming simulation in order to meet stringent product design cycle constraints and safety regulation standards. The parallel performance of MPP version of LS-DYNA on a given hardware system depends on the following factors: the speed of single processor, the communication characteristics, the load balance, the speed of interconnect and most importantly, the balance of the entire hardware system. In this paper, an in-depth analysis will be performed to study the parallel performance of LS-DYNA on Cray XD1 system (in comparison to an Opteron cluster). Various phases involved in a typical crash simulation, such as, initialization, element processing, contact and rigid bodies calculations will be analyzed. An MPI profiling tool will be used to monitor MPI performance in the context of computation, communication, and synchronization aspects of LS-DYNA. The communication patterns and message sizes will also be reviewed for two standard benchmarks (www.topcrunch.org).

In the following sections, benchmark descriptions, runtime statistics, the effectiveness of Linux Synchronized Scheduler to LS-DYNA performance, the scalability of various phases of LS-DYNA simulation, MPI communication and synchronization, and MPI communication patterns and message sizes will be given.

BENCHMARK DESCRIPTIONS

Model Descriptions

LS-DYNA standard benchmarks, Neon and 3-Car Collision models downloaded from www.topcrunch.org, are used in this analysis.

Neon is a frontal impact model and it has a total of 535,070 elements (532,077 shell elements, 73 beam elements and 2,920 solid elements). It has 2 contact interfaces and 324 materials. The simulation time used in this analysis is 30 ms (which corresponds to 29,977 problem cycles).

3-car collision model has a total of 794,780 elements (785,022 shell elements, 116 beam elements and 9,642 solid elements). It has 6 contact interfaces and 1,052 materials. The simulation time used in this analysis is the full simulation time of 150 ms (which corresponds to 149,881 problem cycles).

LS-DYNA Version and Binary Creation

LS-DYNA version 970, revision 5434a is used. The LS-DYNA source is compiled using PGI 5.2.4 compiler with single precision (32 bits). The LS-DYNA executable for Cray XD1 is dynamically linked to MPICH 1.2.5 library and Cray RapidArray [1] 1.1 library. The executable for the 1.8 GHZ Opteron cluster is statically linked to MPICH 1.2.5 library and GM 2.0.8 library.

Computer System Descriptions

Comparative study is performed for the scalability of LS-DYNA on both the Cray XD1 system and a 1.8 GHZ Opteron cluster.

The Cray XD1 system used in this study contains 6 chassis and each chassis has the following hardware characteristics:

- CPU: 64-bit AMD Opteron 200 series processors at 2.2 GHz. clock speed, 12 CPUs per chassis
- Main Memory: 48 GB PC3200 (DDR 400) registered ECC SDRAM per chassis
- SMP: Six 2-way SMPs per chassis
- Interconnect: 2 Cray RapidArray links per SMP (bandwidth of 4 GB/s per SMP); 2 μs MPI latency between SMPs.

The operating system used on Cray XD1 is Suse Linux 8 based with some optimizations made for high performance. Linux Synchronized Scheduler (LSS) that is the key contributor to superior system balance is one of the OS optimizations on Cray XD1. The effectiveness of LSS to the LS-DYNA performance will be discussed later in this paper.

The 1.8 GHZ Opteron cluster has the following hardware configurations:

- CPU: 64-bit AMD Opteron 200 series processors at 1.8 GHz. clock speed
- Interconnect: Myrinet

The operating system used on the 1.8 GHz. Opteron cluster is Suse Linux 8.

Domain Decompositions

MPP970 supports two decomposition methods, RCB (the Recursive Coordinate Bisection algorithm), and GREEDY (a simple neighborhood expansion algorithm). RCB, the default method, was used for the Neon and 3-car collision models because it generally performs better than GREEDY.

MPI Profiling Tools and Descriptions

In this analysis, FPMPI [2] is used to understand the communication patterns and load balances in LS-DYNA. FPMPI is a simple MPI profiling library that was developed by the Argonne MCS and was ported by Cray to XD1 and to the Opteron cluster. Using the MPI Profiling hooks, FPMPI traps calls to many MPI message-passing routines and records data about the messages sent and received. At the call to MPI_Finalize, FPMPI summarizes this data and writes it out to a text file.

To use FPMPI, MPP970, revision 5434a is relinked to FPMPI library for both Cray XD1 and the 1.8 GHz. Opteron cluster. The runtime overhead of FPMPI is almost negligible.

RUNTIME STATISTICS

Table 1 shows the runtime statistics of the Neon model on both 2.2 GHz. Cray XD1 and the 1.8 GHz. Opteron cluster with Myrinet interconnect. The relative speedup from Cray XD1 to the Opteron cluster ranges from 30% to 57% for processor counts of 2 to 64, as shown in the left chart of Fig. 1, which is 8-35% better than the CPU clock rate speedup (22%). The right chart of Fig. 1 shows the comparison of parallel speedup between Cray XD1 and the Opteron cluster for the Neon model. At high processor counts (24-64 CPUs), the Cray XD1 scales 12-31% better than the Opteron cluster. The Opteron cluster stops scaling at 48 CPUs and Cray XD1 continues to show good scaling even beyond 64 CPUs.

Table 1: Runtime statistics for the Neon model

Processors	Cray XD1 Wallclock T_{XD1} (s)	Cray XD1 Parallel speedup	Opteron Cluster Wallclock T_{OC} (s)	Opteron Cluster Parallel Speedup	Cray XD1 Relative Speedup $T_{OC}/T_{XD1}-1$ (%)
1	12713	1	15254	1	20.0
2	6494	1.96	8456	1.80	30.2
4	3389	3.75	5043	3.02	48.8
8	1795	7.08	2334	6.54	30.0
16	969	13.12	1294	11.79	33.5
24	740	17.18	1037	14.71	40.1
32	607	20.94	796	19.16	31.1
48	455	27.94	610	25.01	34.1
64	380	33.46	598	25.51	57.3

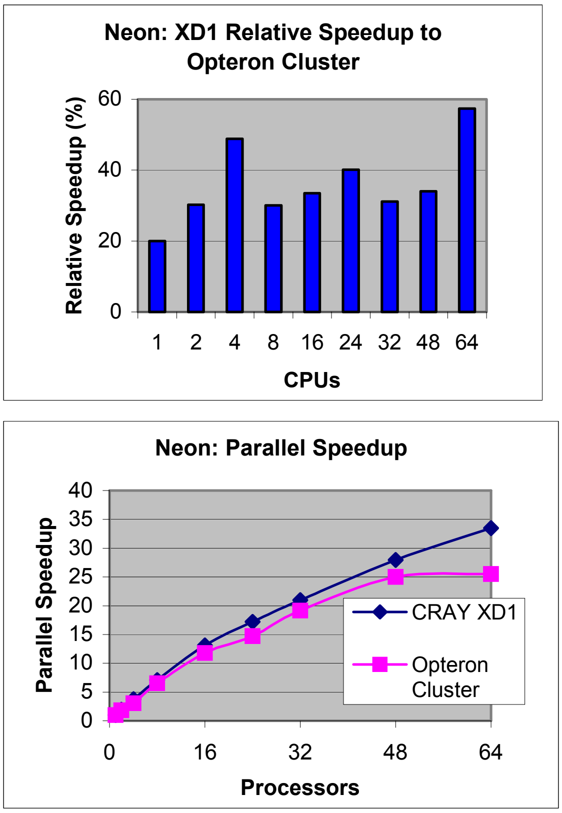


Figure 1: Relative and parallel speedup for the Neon model

Table 2 shows the runtime statistics of the 3-car collision model on both 2.2 GHz. Cray XD1 and the 1.8 GHz. Opteron cluster with Myrinet interconnect. The relative speedup from Cray XD1 to the Opteron cluster ranges from 30% to 46% for processor counts of 4 to 64, as shown in the left chart of Fig. 2, which is 8-24% better than the CPU clock rate speedup (22%). The right chart of Fig. 2 shows the comparison of parallel speedup between Cray XD1 and the Opteron cluster for the 3-car collision model. At high processor counts (32-64 CPUs), Cray XD1 scales 9-12% better than the Opteron cluster. Note that the parallel speedup for the 3-car model is calculated based on the 4 CPUs' elapsed time because the single CPU time is not available.

Table 2: Runtime statistics for the 3-car collision model

Processors	Cray XD1 Wallclock T_{-XD1-} (s)	Cray XD1 speedup	Opteron Cluster Wallclock T_{-OC-} (s)	Opteron Cluster Speedup	CRAY XD1 Relative Speedup T_{-OC-}/T_{-XD1-}-1 (%)
4	12713	1	15254	1	30.2
8	6494	1.96	8456	1.80	31.1
16	3389	3.75	5043	3.02	34.1
24	1795	7.08	2334	6.54	33.5
32	969	13.12	1294	11.79	42.1
48	740	17.18	1037	14.71	43.9
64	607	20.94	796	19.16	46.2

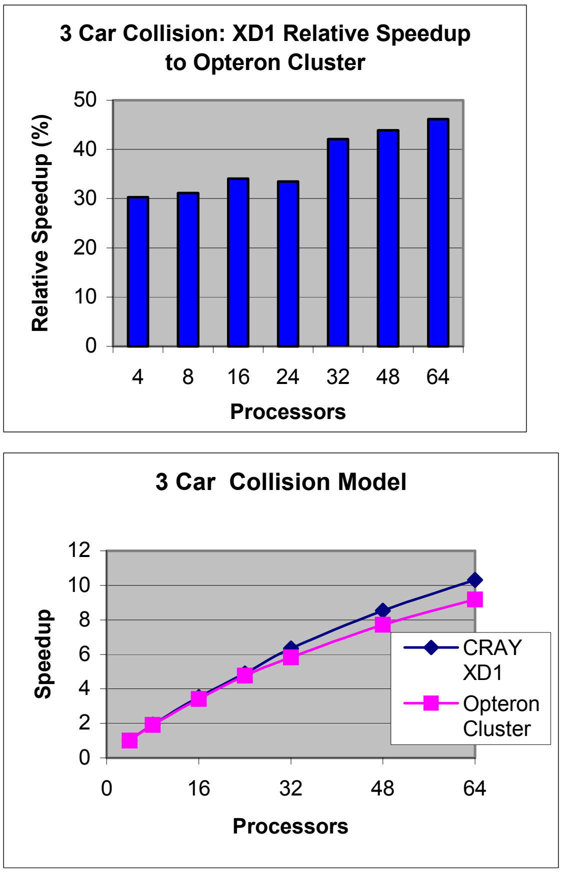
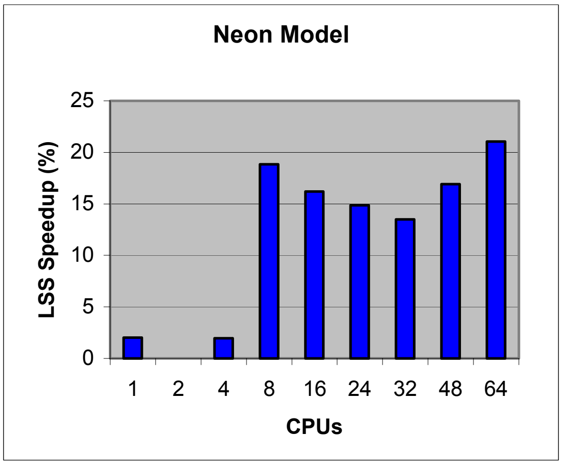


Figure 2: Relative and parallel speedup for the 3-car collision model

The Effectiveness of Linux Synchronized Scheduler to LS-DYNA performance

Fig. 3 shows that the effectiveness of Linux Synchronized Scheduler (LSS) to LS-DYNA performance on Cray XD1 for the Neon and 3-car collision models. With LSS, LS-DYNA performance on Cray XD1 is improved by up to 21%. LSS was implemented on Cray XD1 to ensure that processes execute in the same time slot system-wide. It is one of key factors that made Cray XD1 a very well balanced system.



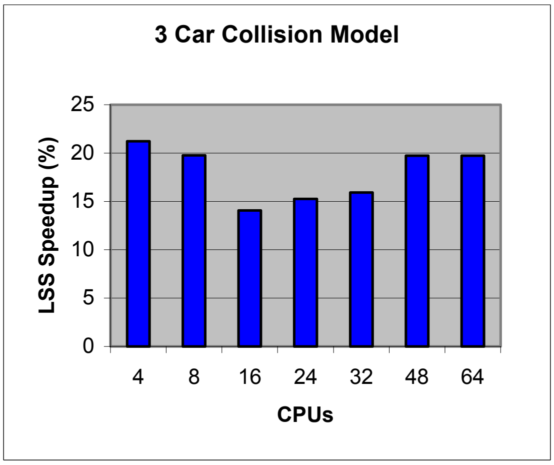


Figure 3: The effectiveness of Linux Synchronized Scheduler

The Scalability of Various Phases of LS-DYNA Simulation

To investigate the parallel scaling of the various components of the LS-DYNA program, we have extracted from d3hsp (output file) the average times for three parts of the LS-DYNA runs for the Neon and 3-car collision models on Cray XD1 and the Opteron cluster, as shown in Figures 4 and 6

:

- **Initialization.** Initialization does not scale because reading the input deck, allocating memory, initializing variables, and domain decomposition is performed serially. As the processor count is increased, initialization costs do represent a higher percentage of the runtime. Overall, however, the amount of time spent in initialization is still quite minimal.

- **Element processing.** The element-processing phase scales quite well due to good load balance, as indicated in the left chart of Fig. 5 for the Neon model and in the left chart of Fig. 7 for the 3-car model. In comparison, the Cray XD1 still scales 9-22% better than the Opteron cluster in element processing for the Neon model.

- **Contact and rigid bodies.** Contact and rigid bodies calculations show limited scaling (largely due to poor load balance) as indicated in the right chart of Fig. 5 for the Neon model and in the right chart of Fig. 7 for the 3-car model. In comparison, the Cray XD1 still demonstrates 8-67% better scalability than the Opteron cluster in contact and rigid bodies calculations for the Neon model at processor counts of 16-64. In the 3-car collision model, Cray XD1 scales 3-30% better than the Opteron cluster at the same processor counts.

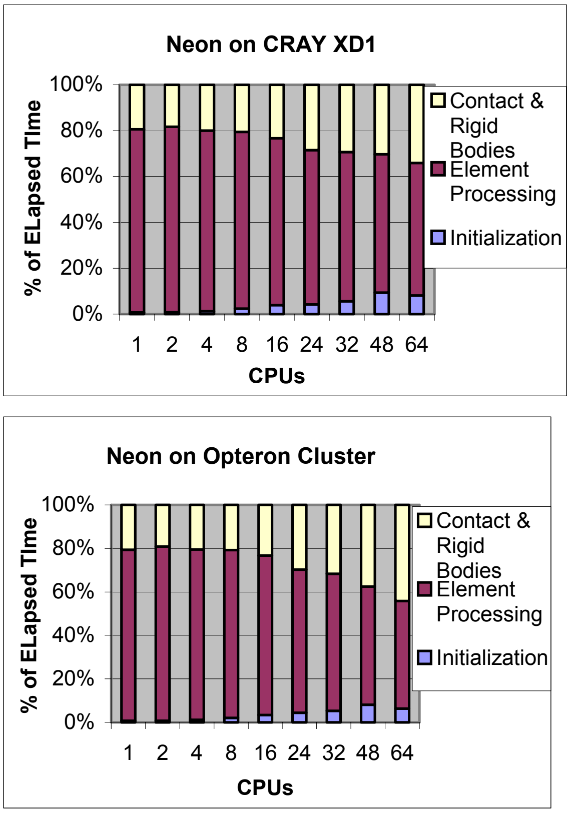


Figure 4: Distribution of initialization, element processing, and contact and rigid bodies calculations on Cray XD1 and the Opteron cluster for the Neon model

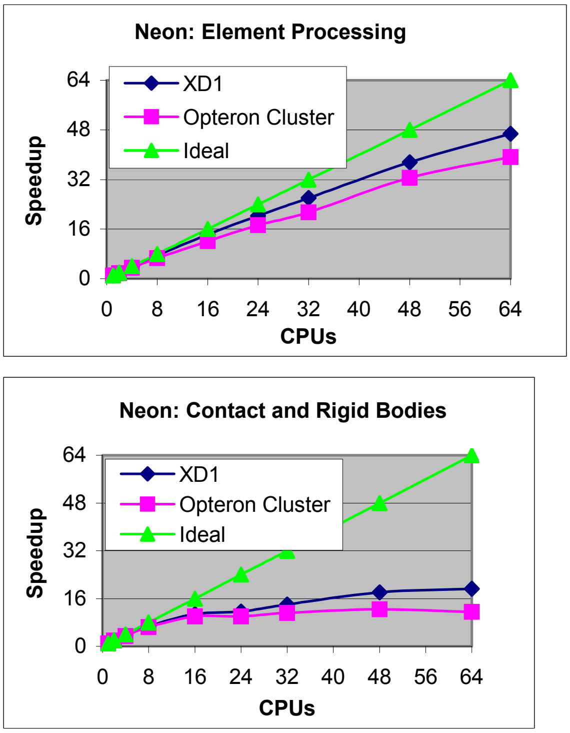
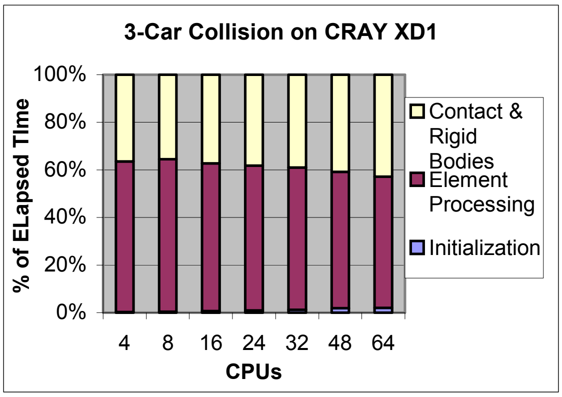


Figure 5: Parallel speedup of element processing, and contact and rigid bodies calculations on Cray XD1 and the Opteron cluster for the Neon model



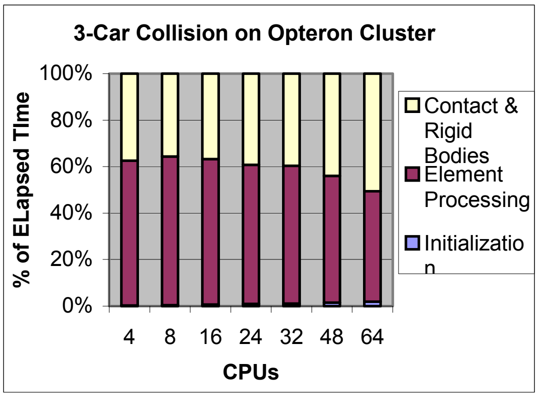
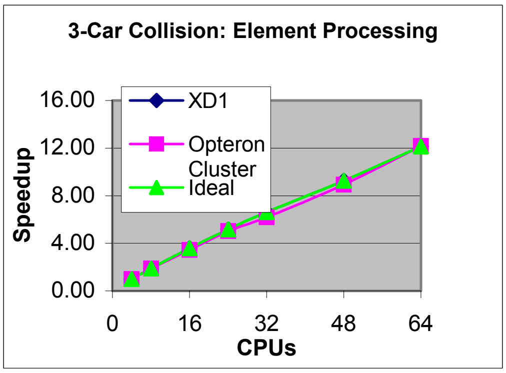


Figure 6: Distribution of initialization, element processing, and contact and rigid bodies calculations on Cray XD1 and the Opteron cluster for the 3-car collision model



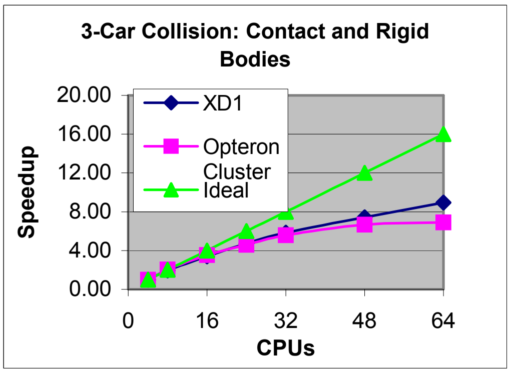


Figure 7: Parallel speedup of element processing, and contact and rigid bodies calculations on Cray XD1 and the Opteron cluster for the 3-car collision model

MPI Communication and Synchronization

The MPI profiling library FPMPI is used to monitor the MPI performance in MPP970 for the Neon and 3-car collision models. FPMPI provides the statistical information for minimum, maximum, and average time spent on computation, synchronization, and communications for each run. It also provides information on minimum, maximum, and average communication time spent on each MPI function call at various message sizes.

The following definitions apply to these graphs:

- *Average communication time* is the average time that each processor spends executing the MPI communication calls without including the synchronization time.
- *Average synchronization time* is the average time that each processor spends synchronizing with other processors. The higher the load imbalance, the longer the synchronization time.

Fig. 8 shows the distribution of the computation, communication and synchronization time on Cray XD1 for the Neon and 3-car collision models.

The communication time on Cray XD1 is only about 1/3 of that on the Operon cluster for the Neon model at 48 and 64 CPUs and for the 3-car collision model at 64 CPUs, as shown in Fig. 9. For the Neon model at 16-32 CPUs and for the 3-car collision model at 24-48 CPUs, the communication time on Cray XD1 is about 1/3-1/2 of that on the Opteron cluster. The reduction in communication overhead on Cray XD1 is due to the use of high speed interconnect, RapdArray.

The measured MPI latency of RapidArray is about 2 µs, which is about 1/3 of that of Myrinet, and the measured MPI long message bandwidth of RapidArray is 1.3 Gbytes per second, which is about 3 times of that of Myrinet.

In the Neon model, the synchronization time on Cray XD1 is about 1/2 to 3/4 of that on the Opteron cluster, as shown in the left chart of Fig. 10. In the 3-car collision model, the synchronization time on Cray XD1 is about 1/2 to 2/3 of that on the Opteron cluster, as shown in the right chart of Fig. 10. The reduction in synchronization time on Cray XD1 is largely due to the use of Linux Synchronized Scheduler (LSS).

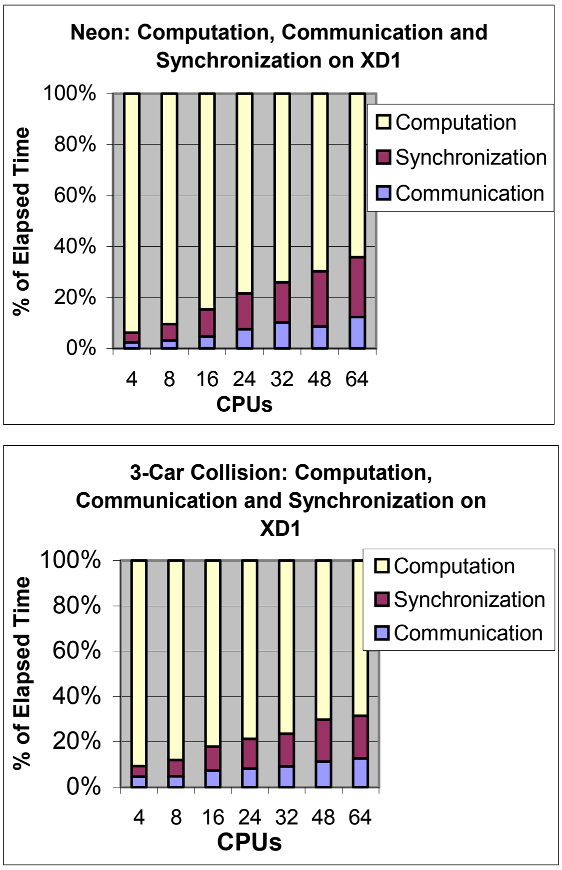
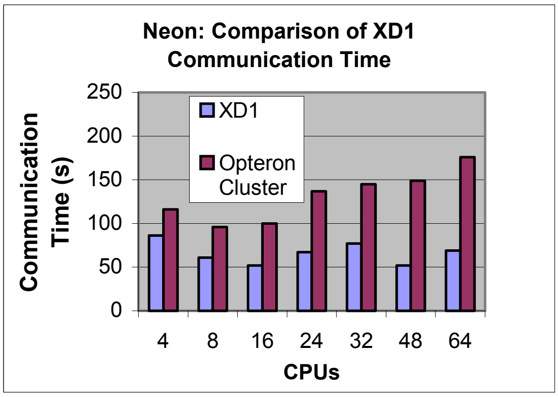


Figure 8: Distribution of computation, communication and synchronization on Cray XD1 for the Neon and 3-car collision models



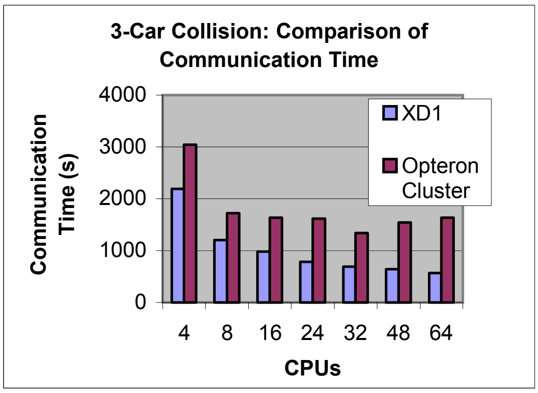
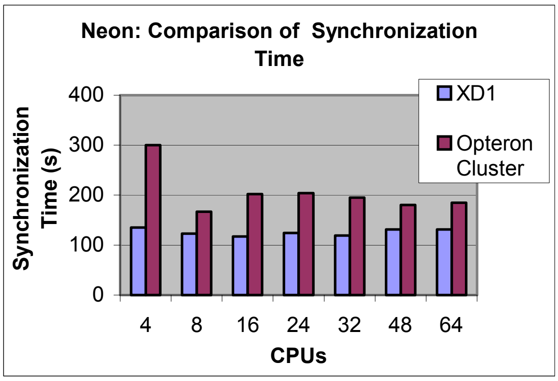


Figure 9: Comparison of the communication time for the Neon and 3-car collision models



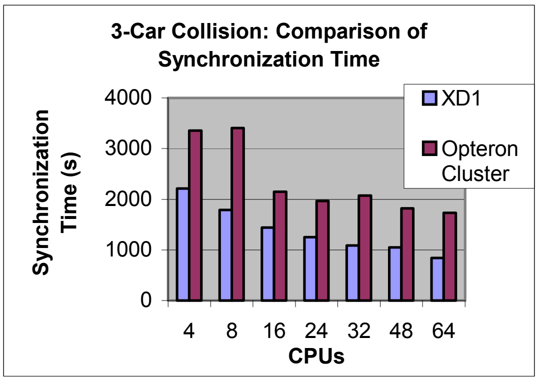


Figure 10: Comparison of the synchronization time for the Neon and 3-car models

MPI Communication Patterns and Messages Sizes

Figure 11 shows the distribution of MPI message passing function calls made in LS-DYNA on Cray XD1 for the Neon and 3-car collision models. The most expensive MPI calls used in LS-DYNA on Cray XD1 are MPI_Recvl, MPI_Allreduce and MPI_Alltoall. These three MPI calls are also the most time consuming ones on the Opteron cluster except with a different distribution. On Cray XD1, at lower processor counts (2-24 CPUs), the MPI_Recv dominates the MPI communication time; at higher processor counts (32-64 CPUs), both the MPI_Recv and MPI_Allreduce dominate the MPI communication time. The distribution of the MPI calls can vary from one platform to another depending on the interconnect speed and the MPI library implementation. It can also vary from one input deck to another

.

Figure 12 shows the comparison of total time spent on the MPI_Alltoall call between Cray XD1 and the Opteron cluster (with Myrinet) for the Neon and 3-car models. It is measured that the message size for MPI_Alltoall in LS-DYNA is 1 to 4 bytes. For message size in that range, the communication time is very sensitive to the MPI latency. As indicated earlier in this paper, the MPI latency of the RapicArray interconnect is one third of that of the Myrinet interconnect and, furthermore, the MPI_Alltoall function has been optimized specially for Cray XD1. As a result, the MPI_Alltoall on Cray XD1 is ten to twelve times faster than that on the Opteron cluster at high processor counts (48 to 64 processors) and it is about 3 to 4 times faster than the Opteron cluster at smaller processor counts (8-32 processors).

Figure 13 shows the comparison of total time spent on the MPI_Allreduce call between Cray XD1 and the Opteron cluster for the Neon and 3-car models. For the Neon model, the MPI_Allreduce on Cray XD1 is 1.4 to 2.5 faster than that on the Opteron cluster, as shown in the left chart of Fig. 13; for the 3-car collision model, the MPI_Allreduce on Cray XD1 is 1.5 to 2 times faster than that on the Opteron cluster, as shown in the right chart of Fig. 13.

Figure 14 shows the comparison of total time spent on the MPI_Recv call between Cray XD1 and the Opteron cluster for the Neon and 3-car collision models. For the Neon model, the MPI_Recv on Cray XD1 is 1.4 to 2.1 times faster than that on the Opteron cluster for processor counts of 4-64, as shown in the left chart of Fig. 14; for the 3-car collision model, the MPI_Recv on Cray XD1 is 1.4 to 2.4 times faster than that on the Opteron cluster for the same processor counts, as shown in the right chart of Fig. 14.

Figure 15 shows the average message size used for the MPI communication in LS-DYNA for the Neon and 3-car models. The average message size is calculated as follows: dividing the total number of bytes transferred in each run by the total number of MPI calls made in the same run. For the Neon model, the average message size ranges from 1 Kbytes to 4 Kbytes; for the 3-car model, the average message size ranges from 2 Kbytes to 5 Kbytes. In both these models, the average message size decreases as the number of processor increases.

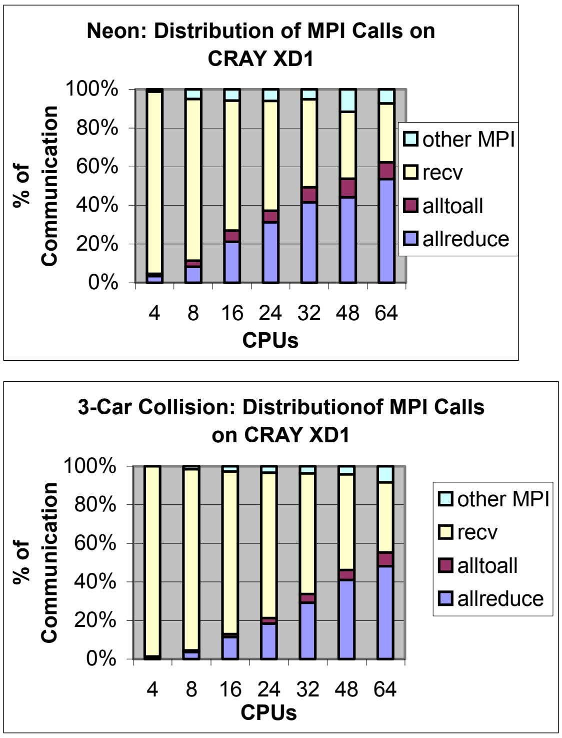


Figure 11: Distribution of MPI calls made in LS-DYNA for the Neon and 3-car collision models

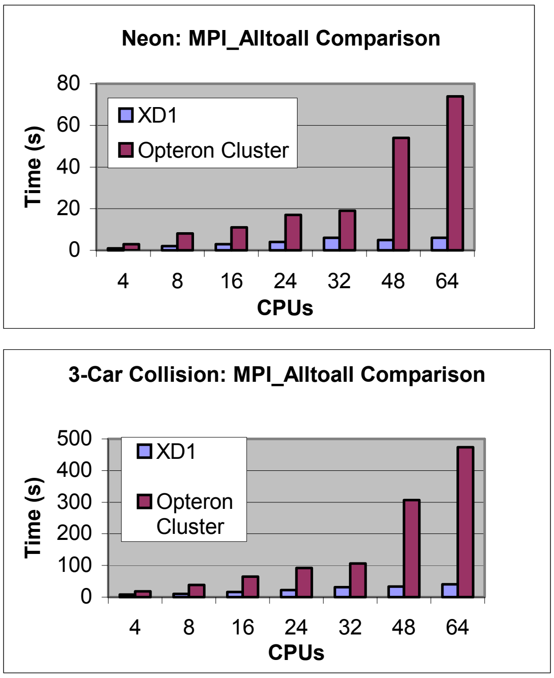
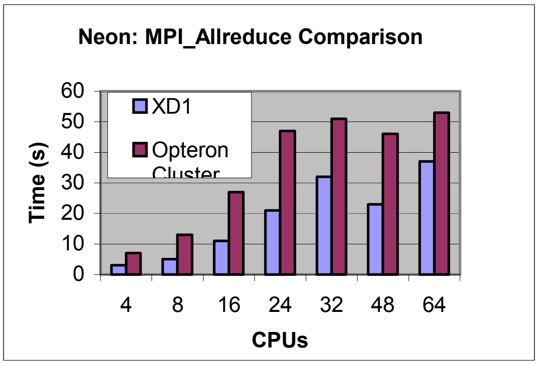


Figure 12: MPI_Alltoall comparison betweem Cray XD1 and the Opteron cluster for the Neon and 3 car collision models



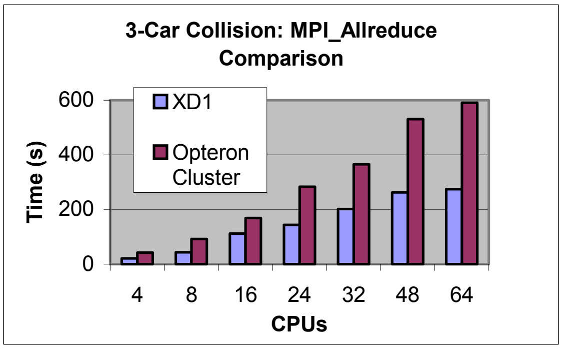


Figure 13: MPI_Allreduce comparison between Cray XD1 and the Opteron cluster for the Neon and 3-car collision models

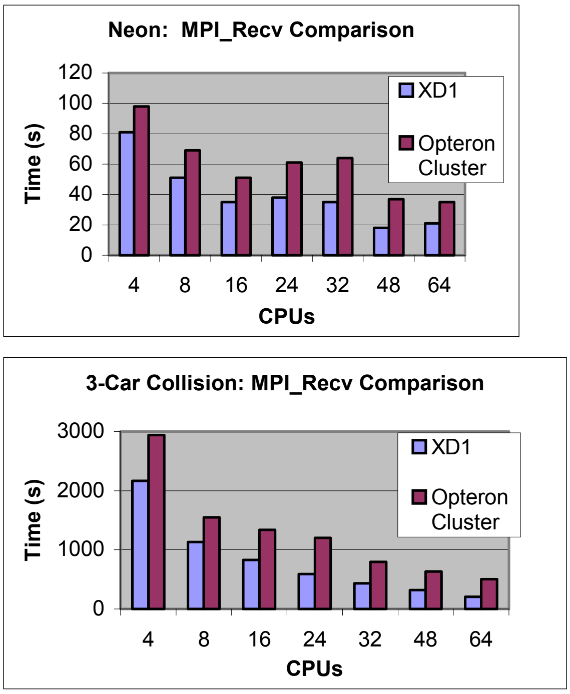


Figure 14: MPI_Recv comparison between the Cray XD1 and the Opteron cluster for the Neon and 3 car collision models

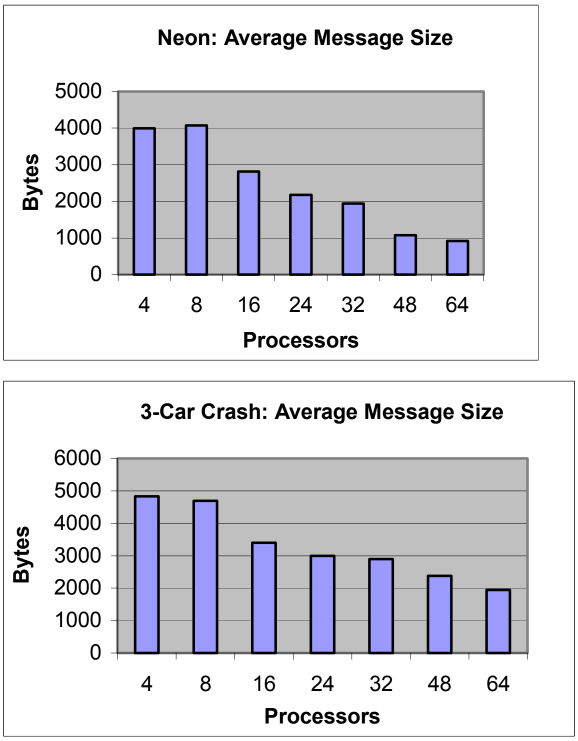


Figure 15: Average message size used for MPI communication in LS-DYNA for the Neon and 3-car collision models

Summary and Conclusions

The LS-DYNA scalability performance on the 2.2 GHz. Cray XD1 has been assessed. The runtime on Cray XD1 is 30-57% faster than that on the Operon cluster. The parallel speedup on Cray XD1 is 9-31% better than that on the Opteron cluster at processor counts of 24-64. The communication time on Cray XD1 is about one third of that on the Opteron cluster because of the use of high speed interconnect, RapidArray. The synchronization time on Cray XD1 is about one half of that on the opteron cluster because of the use of Linux Synchronized Scheduler. The MPI_Alltoall is about 10 to 12 times faster on Cray XD1 than on the Opteron cluster. The MPI_Allreduce is about 2 times faster on Cray XD1 than on the Opteron cluster. The MPI_Recv is also about 2 times faster on Cray XD1 than on the Opteron cluster. In conclusion, Cray XD1 system's RapidArray Interconnect and the Linux Synchronized Scheduler are the two primary reasons why Cray XD1 outperforms other microprocessor based clusters, for the standard LS-DYNA benchmarks (www.topcrunch.org).

References

1. http://www.cray.com/products/xd1/specifications.html
2. http://archive.ncsa.uiuc.edu/lists/perftools/apr02/msg00005.html