Scaling Study of LS-DYNA MPP on High Performance Servers

Youn-Seo Roh
Sun Microsystems, Inc.
901 San Antonio Rd, MS MPK24-201
Palo Alto, CA 94303 USA
youn-seo.roh@sun.com

ABSTRACT

With LS-DYNA MPP, scalable Solaris™ operating system and the MPI library, Sun Microsystems' Starfire™ server proved to be capable of producing a scalable solution for large-scale automotive crash simulation problems. It was found that a proper decomposition plays a significant role in achieving optimal scaling results for large-model, computation-intensive runs. Also, a large amount of external cache memory on the Starfire SMP server was found to be crucial for optimal runtime performance. With proper decomposition, a Starfire server was able to achieve 30X speedup and 93% efficiency with 32 processors in the simulation run of the NCAC Neon model consisting of 270,000 elements. Version 940.2a of LS-DYNA MPP showed good repeatability over largely different numbers of processes. It also displayed an exact repeatability on different runs when the command setting and number of processes are kept constant.

INTRODUCTION

The SMP version of LS-DYNA has been widely used for reliable simulation of automotive crashes. As the finite element model becomes larger, the need for parallel computation and a scalable solution becomes more important. However, it is known that the loop-level parallelism of SMP binaries shows significantly limited scaling behavior for any number of processors above 8 [1]. On the other hand, massively parallel processing (MPP) binary of LS-DYNA takes advantage of better scaling properties through domain decomposition technique and message passing interface (MPI) programming. Therefore, there has been an increased interest in recent years in introducing LS-DYNA MPP in larger scale crash simulation runs [2-7].

Sun Microsystems' Starfire server [8] has been widely accepted in the financial and telecommunication markets. With its widespread acceptance in various industries, one of the areas that Starfire server covers is the high performance computing arena. With its highly scalable performance characteristics together with proven reliability and availability features, the Starfire is considered to be an uninterrupted computational resource in automotive industries. In this study, this consideration was tested and verified through a scaling study of the LS-DYNA MPP running large-scale public domain crash models.

APPROACH

Benchmark Problems

Two public domain finite element crash models were used for this study of representative cars. The first one is a 270,000-element Plymouth Neon model for a full frontal barrier crash. The second one is a 28,000-element Ford Taurus model also for a full frontal barrier crash. Both models were developed at the FHWA/NTSA National Crash Analysis Center at the George Washington University [9]. Table 1 shows the summary of both models studied.

Table 1. Summary of benchmark problems

	Neon	**Taurus**
Number of nodes	286,103	26,737
Number of Solid Elements	2,956	341
Number of Shell Elements	269,329	27,873
Number of Beam Elements	63	140
Number of Active Parts	325	124

Benchmark systems
For the benchmarks, LS-DYNA MPP binary version 940.2a was used.

Hardware systems used were two Sun Microsystems' Enterprise™ 10000 (E10000) or Starfire, servers. Server A was with 64 333MHz UltraSPARC processors with 4MB of external cache and 64 GB of shared memory. Server B was with 64 400MHz UltraSPARC processors with 8MB of external cache and 64 GB of shared memory. For both servers, Solaris 7 was used as the operating system. The multi-process runtime environment used was Sun's HPC Cluster Tools version 3.0 [10]. Job execution and load balancing were done by Platform Computing's LSF 3.2.3. Table 2 shows the summary of benchmark systems.

Table 2. Summary of Benchmark Systems

Executable	LS-DYNA MPP v.940.2a	
Hardware	Server A: E10000 64 x 333MHz UltraSPARC, 4MB external cache, 64GB shared memory	Server B: E10000 64 x 400MHz UltraSPARC, 8MB external cache, 64GB shared memory
Operating System	Sun Solaris 7	
Runtime Environment	Sun HPC Cluster Tools v.3.0, MPI v.6.0	
Load Balancing	Platform Computing LSF v.3.2.3	

As a true shared memory processors (SMP) system that can be scaled up to the full 64-cpu configuration, a Starfire has a certain characteristics that suits scalable applications:

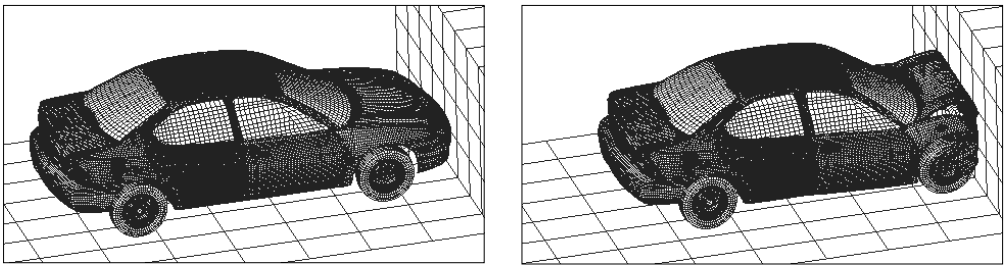
- fast interconnect called Gigaplane-XB supporting 12.8 GBytes/second memory bandwidth and 6.4 GBytes/second peak I/O bandwidth.

- Constant low latency to memory: 400-600nsec latency, equal access time to all processor boards.

This feature together with the scalable Solaris operating system results in a very good scalability up to 64 processes. In many user benchmarks, the system shows super-linear or near-linear scaling up to full 64 processes. In this study, this scalability of Starfire servers has been tested against number crunching crash simulation benchmarks.

DISCUSSION OF RESULTS

- Neon Model



Model at the start of the simulation. Model at the end of simulation time, 80msec.

Figure 1. Results of Neon model simulation.

Initial scaling study was done on the server A of Table 2. Table 3 shows the scaling result for the Neon model. The simulation time was from 0 to 10msec. The data were obtained with the default decomposition of the model.

Table 3. Scaling results of Neon model on the server A. Run to 10msec with default decomposition.

NCPU	Elapsed Time(sec)	Scaling	Efficiency(%)
1	92566	1.0	100
2	56708	1.6	82
4	29586	3.1	78
8	14269	6.5	81
16	8298	11.2	70
32	4401	21.0	66
48	3548	26.1	54
56	3230	28.7	51

The data of Table 4 show improved scaling with the aid of better decomposition. The decomposition scheme uses pfile command line input. The pfile used in the study was:

```
decomp {
     expdir 2 expsf 15
     silist 2,6
}
```

With this pfile, two different kinds of decomposition were being applied to the problem. As shown in the first line, by changing the scale factor in a certain direction, it is possible to assign more computations in one part of the model. In the case shown above, direction 2 is the vehicle longitudinal axis, and with expsf of 15, the decomposer assigns more processes for the front part of the vehicle where computationally more expensive contacts occur. The net effect then becomes a better load balancing among processors that in turn allows for better scalability.

The second line of pfile directs the decomposer routine to apply decomposition first to the specified sliding interfaces 2 and 6 using the total available processors where computationally more expensive contact calculations happen. The decomposition of the remaining workload consisting of less expensive typical element processing follows afterwards. By doing this, more even distribution of workload to the total available processors, which in turn results in better scalability. It turns out that a single large contact definition is beneficial for good scalability. In addition, it is possible to group the contact definitions of existing models into one big contact definition in pursuit of better scalability, although the procedure involves modification of the model that can be time consuming [11].

Table 4. Scaling results of Neon model on the server A. Run to 10msec with decomposition using pfile.

NCPU	Elapsed Time(sec)	Scaling	Efficiency(%)	Timing Improvement w.r.t. Table 3
1	94130	1.0	100	0.98X
2	41594	2.3	113	1.36X
4	21773	4.3	108	1.36X
8	10500	9.0	112	1.36X
16	6258	15.0	94	1.33X
32	3361	28.0	88	1.31X
48	2542	37.0	77	1.40X
56	2433	38.7	69	1.33X

Table 5 shows another scaling results of the Neon test case on the other Starfire server with 400MHz, 8MB external cache memory (server B). It shows super-linear scaling up to 8 processes and near-linear scaling up to 32 processes where the efficiency reaches 93%. The run still scales well above 32 processes, although the rate starts to level off. This seems to be because for the given amount of the total computation, which is determined by the size of the model, the cost of inter-process communications starts to cancel off the performance gain from parallel computation. Compared to the results of Table 4 from server A, the elapsed time shows up to 46% of improvement. Taking into account the 20% increase in processor speed, the rest of the improvements up to 23% are from the larger external cache size allowing more computations to be done within the cache memory.

Table 5. Scaling results of Neon model on the server B. Run to 10msec with decomposition using pfile.

NCPU	Elapsed Time(sec)	Scaling	Efficiency(%)	Timing Improvement w.r.t. Table 4
1	68599	1.0	100	1.37X
2	31735	2.2	108	1.31X
4	15406	4.5	111	1.41X
8	8077	8.5	106	1.30X
16	4493	15.3	95	1.39X
32	2310	29.7	93	1.46X
48	1805	38.0	79	1.41X
56	1664	41.2	74	1.46X

Figure 2 summarizes scaling results of the Neon model, revealing the noticeable effect of decomposition on scalability.

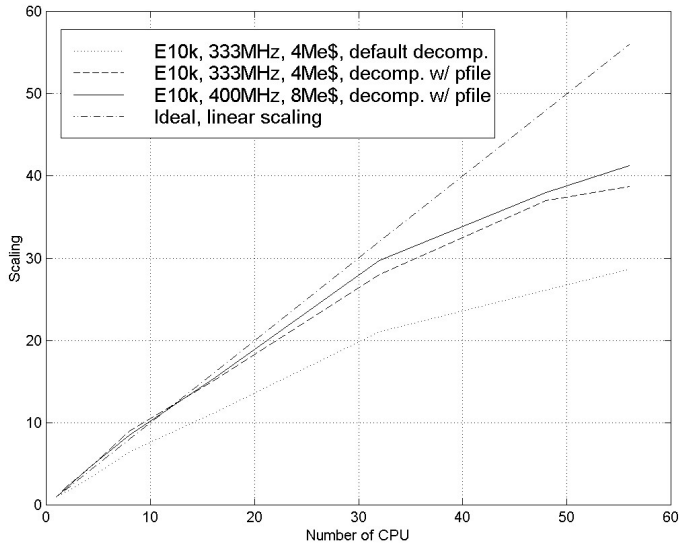


Figure 2. Scaling results of Neon model. Runs to 10msec.

Table 6 shows the elapsed times for the full 80msec simulations on Server B. With the same expsf value used for the data of Table 3-5, a floating point exception was caused at around 40msec. However, if a different value of expsf for the number of processors equal to 32 was used, this made the full-time run finish without error. The floating-point exception needs to be examined further.

Table 6. Elapsed time for Neon model. Runs to 80msec with decomposition on Server B.

NCPU	Elapsed Time (sec)
8	67619 (18.8 hr)
16	FPE
32	18594 (5.2 hr) (expsf = 30)
48	14631 (4.1 hr)
56	13581 (3.8 hr)

-Taurus Model

The Taurus model is much smaller than the Neon model, however the same principle of decomposition was applied to verify the scalability. With the pfile of:

```
decomp {
     expdir 2 expsf 15
}
```

It was possible to obtain a superlinear scaling up to 8 processors and 95% efficiency at 16 processors as shown in Table 7.

Beyond 16 processors, the scalability rapidly decays when the cost of inter-process communication starts to outweigh the benefit of parallel computations.

Table 7. Scaling results of Taurus model on the server A. Run to 5msec with pfile decomposition.

NCPU	Elapsed Time(sec)	Scaling	Efficiency(%)
1	3730	1.0	100
2	1771	2.1	105
4	788	4.7	118
8	428	8.7	109
16	246	15.2	95
32	184	20.3	63

CONCLUSIONS

Correctness of the results

In this section, the correctness of multiprocessor parallel runs with LS-DYNA MPP was verified by comparing test results with the Neon model.

Figure 3 shows the normal force component of rwforc output acting on the front rigid wall barrier. Three full-time outputs are compared for the number of processors: 8, 48 and 56. The plots show good coherence for a vastly different number of processors and the amplitude deviations are within 5 per cent of the full scale. Especially for processor number of 48 and 56, the two outputs are very close to each other.

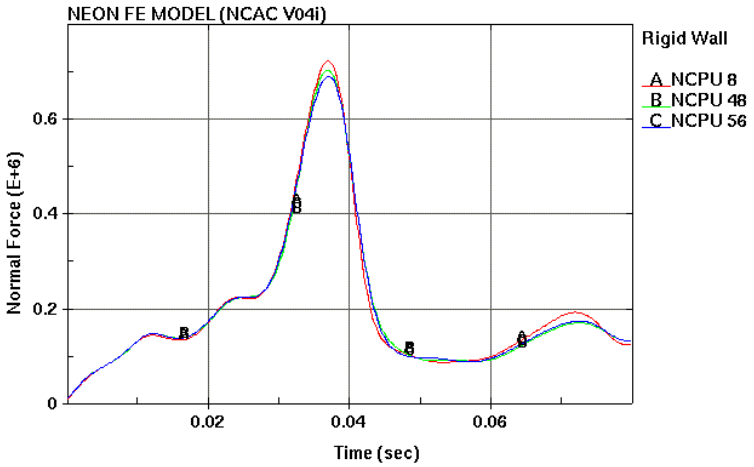


Figure 3. Rigid wall normal force outputs from different runs with 8, 48, and 56 processors. 60Hz SAE filters were used.

For a given number of processes, outputs from two different runs have been compared. Figure 3 shows the comparison results where the two separate runs were using 48 processes. One ASCII output curve of rwforc fell exactly on top of the other. In fact, for the same number of processes and same runtime setting, two runs produced exactly the same ASCII output files generated by the dumpbdb executable. This proves that the new LS-DYNA MPP is capable of producing identically repeatable results when the same number of processes is used between different runs.

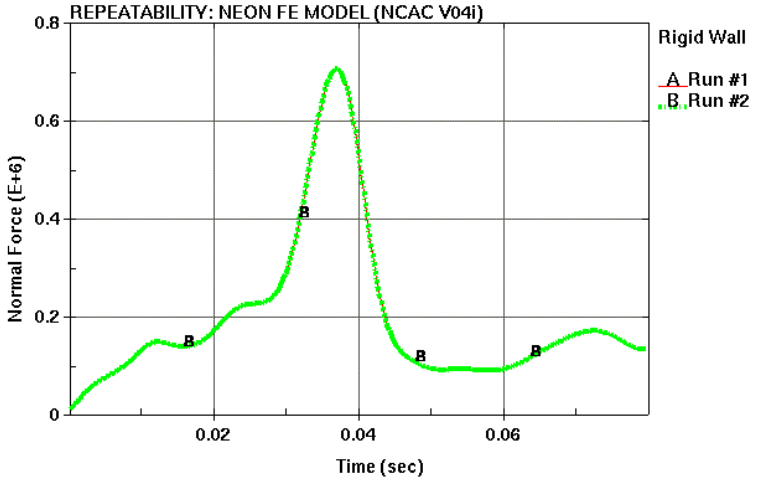


Figure 4. Repeatability test of LS-DYNA MPP. Output is the rigid wall normal force, rwforc. Runs were made on server B with number of processes of 48. 60Hz SAE filters were used.

ACKNOWLEDGEMENT

Dr. C.D. Kan of FHWA/NHTSA National Crash Analysis Center is acknowledged for providing with the Neon model. Drs. Jason Wang and Philip Ho of LSTC are acknowledged for their helpful comments on the decomposition and the use of LS-POST, respectively.

REFERENCES

[1] KAN, C.-D. and LIN, Y.-Y. "Evaluation of High Performance Computer Systems Using a Large Size Finite Element Model," 1999, Second European LS-DYNA Users Conference.

[2] WAINSCOTT, B., WANG, J. and STANDER, N., "LS-DYNA/MPP Version – Development, Quality Assurance, Status and Progress," 1998, 5th International LS-DYNA Users Conference.

[3] KAN, C.-D., Lin, Y.-Y. and HOLLAMBY, R., "Evaluation of MPP Version of LS-DYNA and Its Comparison with the SMP Version," 1998, 5th International LS-DYNA Users Conference.

[4] LIN, Y.-Y. and KAN, C.-D., "Crash Simulation on Parallel Multiprocessors," 1998, 5th International LS-DYNA Users Conference.

[5] LAM, D., SKINNER, G. and PATTANI, P., "On the Road to Achieving LS-DYNA Parallel Performance on High Performance Computers," 1998, 5th International LS-DYNA Users Conference.

[6] ROBICHAUX, J., AKKERMAN, A., BENNETT, C., JIA, R. and LEICHTL, H., "LS-DYNA 940 PARALLELISM ON THE COMPAQ FAMILY OF SYSTMES," 1998, 5th International LS-DYNA Users Conference.

[7] CHU, R. and ZAIS, J., "Activities of MPP-DYNA at SGI-Cray," 1998, 5th International LS-DYNA Users Conference.

[8] http://www.sun.com/servers/highend/10000/index.html

[9] http://www.ncac.gwu.edu

[10] http://www.sun.com/software/hpc

[11] Private conversation with Jason Wang, Livermore Software Technology Corp.