USE OF MPP-DYNA FOR SIMULATING SHEET METAL FORMING PROCESSES

Galbraith, P. Christopher
Metal Forming Analysis Corporation
Medusa Computing Corporation

Thomas, Dylan N.
Centre for Automotive Materials and Manufacturing
Medusa Computing Corporation

Chris Galbraith
Metal Forming Analysis Corporation
2579 Highway #2 East
Kingston, ON K7L 4V1
Canada
Tel: 613-547-5395
Email: galb@mfac.com

Abbreviations:
ASCII - American Standard Code for Information Interchange
CAD – Computer Aided Design
CPU – Central Processing Unit
FMVSS – Federal Motor Vehicle Safety Standard
LAM/MPI – Local Area Multicomputer/ Message Passing Interface
MCC – Medusa Computing Corporation
MPI – Message Passing Interface
MPP – Massively Parallel Processing
RCB – Recursive Coordinate Bisection
SMP – Shared Memory Parallel

Keywords:
Clusters, Linux, LS-DYNA, MPP, Sheet metal forming, SMP

ABSTRACT

Sheet forming simulations have been shown to have a profound impact on the tool and die industry, but accurate solutions for large panels often require large amounts of CPU time. The development of MPP-DYNA has allowed a large number of CPUs to be applied to a single problem thus reducing total elapsed time. This paper discusses the use of MPP-DYNA for obtaining accurate solutions in small amounts of elapsed time using inexpensive PC-based clusters of computers

INTRODUCTION

Finite element simulations of sheet forming operations have long been considered to be an effective means of improving product quality and decreasing the cost and time to market of automobiles [5,6,12]. Until recently, these methods have been employed only by the largest companies –auto manufactures and some of their Tier 1 suppliers. Manpower requirements, initial costs of purchasing hardware and software, and a steep learning curve have prevented smaller companies from embracing this technology.

Gradually, these impediments are being overcome. Specialty software such as DYNAFORM [2] and Hyperform [7] have simplified the generation of an LS-DYNA sheet forming input deck. Instead of taking weeks to generate an input deck, an analyst can go from CAD to LS-DYNA in a matter of hours. Analysts no longer require an advanced degree to run the software, and the learning curve can be scaled much more quickly.

LS-DYNA software costs have remained constant for many years, and the cost of computers has been coming down even though the performance continues to increase. As a result, many smaller sheet forming companies are adding simulation capability. For some companies, this technology is viewed as a means to an end – being able to compete for larger stampings and being recognized as a higher quality producer. The addition of simulation capability is viewed as being strategically important to these companies.

It is important, therefore, that the predictions from the analyses are both accurate and timely. Accuracy comes from doing the little things right, particularly describing the tooling geometry with sufficient numbers of elements so that its shape is captured. A tool and die engineer would not tolerate a die radius that was composed of 3 flat sections. Accordingly, an analyst should use sufficient numbers of elements along curved surfaces to give a smooth representation.

The elements in the blank must be small enough so as to interact with small features in the tooling. As a result, for large stampings, it may be necessary to have well over 100,000 elements in the blank. Large numbers of elements results in small elements and hence small timesteps.

Accurate finite element solutions can result in long analysis times, which represents a problem for a tool and die shop. If the analyst can't come up with an acceptable tooling design in less time than the die designer does by trial and error, he/she may soon be looking for a new job. Further, if the computer hardware required to run the analyses in a timely fashion costs more than the savings available from the use of the analysis tools, it doesn't make sense to implement this technology.

LS-DYNA: SMP vs. MPP

LS-DYNA has its roots in the development of DYNA-3D at Lawrence Livermore National Laboratory in 1976. Initially ported only to supercomputers, in the subsequent 26 years the code has been ported to seemingly slower classes of computers – first to mainframes running operating systems such as VMS and UNIX, then to engineering workstations, and finally to PCs.

The low cost of today's PCs and their superior performance to the supercomputers of the 70's and 80's has brought an immense computational power to the desk of anyone who wishes to use it. Nonetheless, an accurate forming simulation of a large automotive panel is still a time consuming event. In this paper, we discuss how the Massively-Parallel Processing (MPP) version of LS-DYNA can be used in conjunction with low-cost PC-based Linux clusters to reduce the elapsed time to an acceptable level at a reasonable cost.

LS-DYNA is available in Shared Memory Parallel (SMP) and Massively Parallel Processing (MPP) versions. The SMP version of the code will run an analysis on a single computer with 1 or more CPUs. At each stage of the analysis, the governing equations are solved in parallel. Each processor can write/retrieve information to/from a centralized bank of memory (hence "shared memory parallel"). Communication between processors is done internally within the machine across the data bus. This is an extremely fast means of communication, but if enough CPUs are active the bus can overload, creating a bottleneck.

The MPP version will run on one or more computers each with one or more CPUs. Each CPU has its own dedicated local memory, and is assigned a portion of the analysis (Figure 1) by a process called domain decomposition. The domain decomposition allows each processor to solve its piece of the puzzle independently of the other processors.

The CPUs are (possibly) scattered across different computers and connected by means of a network. The typical Ethernet network is not as fast as the data bus in a single SMP computer, but it might not saturate as quickly as extra CPUs are added. This is another reason why MPP-DYNA scales more readily than SMP-DYNA.

Over 90% of the features of SMP-DYNA are available in MPP-DYNA [9,10] and the input decks for MPP and SMP runs are essentially identical. However, the SMP code had to be completely rewritten when the MPP version was created. As a result, there are some important differences in how the two codes behave.

Perhaps the most important difference has to do with how contact is enforced. In the MPP code, each contact algorithm is supported by a new parallel contact algorithm that handles initial penetrations differently. An MPP specific option has been added which allows the initial penetrations to be dealt with without requiring them to be moved and without generating large contact forces. Details can be found in reference [10]. Our experience has shown that even with the use of these features, initial penetrations can lead to serious problems with the MPP code, particularly with seat belt elements, and that initial penetrations should be avoided whenever practical.

Another difference for the user has to do with how output files are post-processed. The ASCII databases produced by SMP versions of LS-DYNA are now combined into a series of binary files named *dbout.** where the asterix is a series of numbers increasing from 0000. After running the analysis, steps can be taken to extract the ASCII files from this series of files. Details can be found in [10].

Finally, the execution environment for SMP and MPP versions of LS-DYNA is quite different. For SMP versions, the user either has a graphical user interface on the PC for submitting jobs, or issues a single command line for the Linux and Unix versions. For MPP runs, the execution environment is somewhat more complicated. It is possible to run MPP-DYNA on single SMP machines, clusters of SMP machines, and clusters of single-CPU machines. In each case, software must be run that permits MPP-DYNA to access the multiple CPUs as if they belonged to a single computer. The steps for doing so are different for each hardware platform and cannot be covered in detail in this document. Generally, though, the user must use software libraries that create a single virtual computer from computers distributed across a network. These libraries are available from several sources, but the two that can be used with LS-DYNA are LAM/MPI [8] and MPICH [11].Once the virtual computer has been created, software that executes LS-DYNA on this virtual computer can be invoked. On the benchmark systems used in this paper, the command would look something like:

mpirun –np 8 –O mpp-dyna i=input.dyn p=pfile

In the above, the "mpirun" is the command that indicates the mpp-dyna code is to be run on a cluster of computers. "-np 8" indicates that 8 processors are to be used. "-O" indicates that the processors to be used are all of a kind (homogeneous), and that the software shouldn't waste time converting data to and from a LAM representation as it is passed between machines. mpp-dyna should be replaced by the name of the mpp-dyna executable, and input.dyn should be replaced by the name of the user's dyna input file.

Finally, the p=pfile portion of the command provides the name of a parameter file (in this case "*pfile*") that assists MPP-DYNA in how it allocates pieces of the model to the different CPUs. The use of the *pfile* can have a profound impact on the clock time taken to run the analysis. Details on the use of the parameter file can be found in Appendix L of the LS-DYNA Keyword User's Manual [10].

SCALABILITY

Scalability is a measure of how well a computer performs as the numbers of CPUs applied to an analysis increases. For example, a computer would scale perfectly (100%) on 10 CPUs if a job run on 10 CPUs completed in 10% of the time taken to run on a single CPU. If it took 20% of the time, then the scalability would be only 50%.

$$\%scalability \cong \frac{elapsedtimefor\,1 \quad processor}{elapsedtimefor\,N \quad processors} \times \frac{100}{N}$$

For finite element simulations, perfect scalability does not exist due to the overhead required to perform domain decomposition and the communication requirements. Each processor then computes the nodal deformations and element values for its own subdomain. Elements that lie on the boundaries between subdomains require input from other subdomains. This information is stored in memory on different computers, and requires communication across the internal bus (for runs on SMP machines) or across the network (for local area machines).

As shown in Figure 1, this process is handled differently by the SMP and MPP versions of LS-DYNA. For SMP, equations are generated for the whole of the structure. These equations are then solved in parallel by distributing them across the CPUs. Equations are formulated, distributed and solved at 4 different stages during each timestep.

In the MPP version, each processor is responsible for solving its subdomain. It only needs to communicate with other processors at two points during each timestep – when the contact forces are being calculated, and when the nodal displacements are being updated. Information about nodes and elements on the interior of the subdomain is all held within local memory. It may be necessary to communicate with other processors to handle contact along the boundaries, and to update the nodes at the end of the timestep, but other than this there is little need for communication.

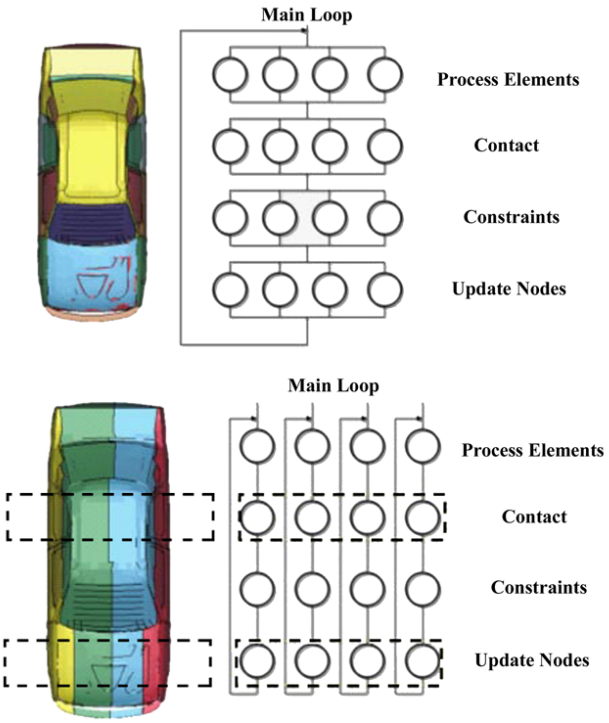

Figure 1. LS-DYNA parallelization strategies. Top: SMP-DYNA. Equations are formulated at each of four stages of the calculation of a timestep. These equations are then solved by a bank of processors. At each stage the code must wait for each processor to finish before going to the next step. Bottom: MPP-DYNA. The model space is

divided into subdomains and each processor calculates values for its own subdomain. A limited amount of information is passed between processors at only two times during each timestep.

The computational efficiency will go up if the ratio of boundary elements to internal elements is reduced because this should reduce the amount of time spent waiting for information from other processors. Similarly, it is ideal to have each processor solving an equal portion of the analysis (in terms of CPU cost). The processors can only solve one timestep before waiting for all the other processors to complete their work.

The domain decomposition can play a vital role in assuring a balanced work load for each processor. If each element in the model required the same amount of CPU time to process, then the ideal domain decomposition would be one in which each CPU processed the same number of elements. However, different elements require differing amounts of CPU time.
Contact further complicates things, because it requires CPU time and is highly non-linear – objects move in and out of contact with other objects during the analysis.

During domain decomposition, the model geometry is sliced into pieces using a Recursive Coordinate Bisection (RCB) and assigned to certain processors (Figure 2). If an even number of processors is used, the model is split into 2 equal domains. The split plane cuts the largest model dimension (x, y, or z) at a location that roughly balances the number of elements in each half. Half of the processors are then allocated to each domain. If an uneven number of CPUs is to be used, the model is split into 2 unequal domains in proportion to the nearest integers to half the number of CPUs. The remaining portions would be split along their largest dimensions and a portion allocated to each processor until each domain has roughly the same number of elements and one processor assigned to provide a solution.

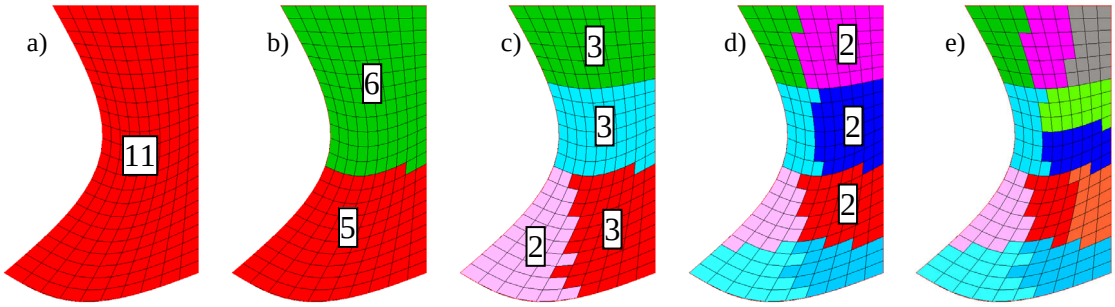

Figure 2. Recursive Coordinate Bisection. The numbers in the boxes indicate the number of CPUs assigned for solving each subdomain (if greater than 1). a) The largest dimension before splitting is the y-direction, so the first cut is made perpendicular to the y-axis. b) If 11 processors are assigned to this analysis, the first cut is made at the 5/11ths mark (i.e. 5/11ths of the elements below the cut, 6/11ths of the elements above. c) the second cuts are made at the 2/5ths marks for the lower domain and 3/6th mark for the upper domain. Note that the lower domain was split perpendicular to x, the upper was split perpendicular to y. d) and e) the bisection continues until each domain has a single CPU assigned to it.

Because the domain decomposition can have a profound impact on how quickly the model is solved, the user has some control over how this calculation proceeds. For example, in sheet forming, it is ideal if each processor has an equal portion of the model, both in terms of numbers of elements, and the amount of contact. This can often best be done if the model is divided into columns oriented in the forming direction (Figure 3). To assure that the model is not divided perpendicular to the forming direction, the forming direction can be scaled by a factor of 0.0 . This calculation, applied only during the RCB, ensures that the forming direction (usually the z-direction) will never be the largest in the model. Details on the structure of the *pfile*, which controls the coordinate bisection, can be found in [10].

The goal of the recursive coordinate bisection is to equally balance the load between the processors. At the same time, it would be ideal if the amount of network traffic was kept to a minimum.

This has an impact not only on how the user sets-up the problem (for recursive coordinate bisection) but also on how systems should be designed for solving MPP-DYNA runs. To keep the network traffic down, you want to keep the number of “boundary” nodes small compared to the number of “interior” nodes. In 3-dimensions, this is best done

by creating cube-sized domains. However, cube-sized domains can result in a single domain needing to communicate with as many as 26 other domains. With standard Ethernet networks, communication with multiple domains is done serially. If each domain had to talk to 26 other domains a great deal of time would be spent waiting for information to flow before any calculations could be done.

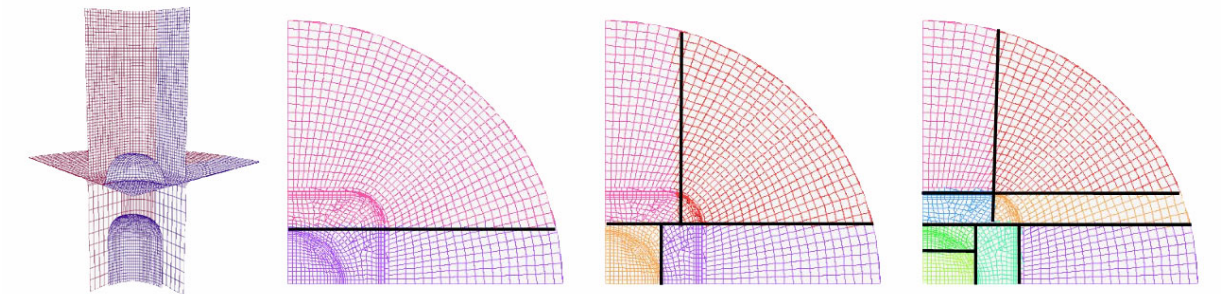

Figure 3. Domain decomposition for sheet forming. a) Each CPU handles a column that runs through the model in the direction of tooling travel. b) Top view showing 2-processor domain decomposition. c) 4 processors d) 8 processors.

It is probably better to keep the number of neighboring domains small, while still attempting to keep the number of boundary nodes small. For example, in crash analyses, it is not uncommon to create domains that run from the front of the car to the back of the car (Figure 1). Although each domain has a large number of boundary nodes as a result, the number of neighbors is at most 2, and communication can occur in an orderly fashion.

SYSTEM DESIGN

If a large amount of data needs to be transmitted across the network, it is essential that the system be built with a high-speed network. Low latency and high transmission rate networks can be created using advanced technologies such as Myrinet and Gigabit Ethernet products, but these are costly. If the number of CPUs applied to the problem is small, then the amount of time performing calculations will be large compared to the amount of time spent waiting for information. In this case, it would be preferable to use cheap commodity technology such as Fast Ethernet and spend any available hardware money on additional CPUs or faster CPUs.

If the network is proving to be the bottleneck because of inter-processor communication, then adding more CPUs can actually increase the elapsed time and any additional money should be spent upgrading the network. As a first pass, channel bonding is a relatively cheap solution that can provide vast improvements in throughput. In channel bonding, each computer has multiple Fast Ethernet cards and network traffic can travel across different network channels. Although the latency is not reduced, the throughput is increased because the pipeline for transmitting data is increased in size.

In extreme cases, where each CPU is solving only a small number of elements, it may be necessary to use expensive network technology to reduce model run times. Benchmark Case #1 below illustrates how networks can become the limiting factor on scalability.

Regardless of which technology is used for building a local area network for cluster applications, it is essential that network traffic is passed through a switch and not a hub. When a hub is used, traffic is handled serially. For example, if computer 1 needs to talk to computer 2 and computer 3 needs to communicate with computer 4, a hub would need to finish the connection between 1 and 2 before 3 and 4 could begin talking. With a switch, both data streams could be handled simultaneously.

BENCHMARK RESULTS

Benchmark problems were run on a cluster of computers provided by Medusa Computing Corporation, a wholly owned subsidiary of Metal Forming Analysis Corporation. The system details are shown in Table 1. Medusa Computing Corporation (MCC) was created in order to provide access for LS-DYNA users to pre-configured clusters of PCs, including specialty software to make running MPP-DYNA as easy to use as possible. The hardware and Linux kernel are optimized to best run MPP-DYNA.

Benchmark problems came from MFAC customers interested in purchasing a cluster from MCC. The customer's originating hardware is identified and the run times on their hardware are listed. In each case, the originating hardware was several years old and due to be replaced, so the timings here are not indicative of the timings that would result from newer hardware.

Table 1. Computer systems for running benchmark problems. System #1 is 18 months old. System 2 uses state-of-the-art components. Systems 3 through 6 are customer systems that are due to be replaced.

System Number	1	2	3	4	5	6
System Descriptor	MCC P-III system	MCC P-4 system	Sun Sparc Ultra 60	HP J-5000	SGI Origin 2000	Compaq Alpha Cluster
Max. # of CPUs	8	8	2	2	18	5
RAM	4 GB	4GB	?	?	9GB	5GB
Clock Speed	800 MHz	2000 MHz	450 MHz	?	300 MHz	667 MHz
Network	Fast Ethernet	Channel bonded Fast Ethernet, Gigabit	Internal	Internal	Internal	Fast Ethernet
LS-DYNA version	MPP	MPP	SMP	SMP	MPP	MPP

Case 1

This model was provided by an automotive seat supplier and consisted of the slide assembly that attaches the seat to the floor and permits the seat to be positioned fore and aft. The model was a simulation of the testing required to meet Federal Motor Vehicle Safety Standard 225 [3]. This rule establishes a new Federal motor vehicle safety standard that requires motor vehicle manufacturers to provide motorists with a new way of installing child restraints.

There are many moving parts in the slider and extensive contact between parts. The part is small in size, but is costly to solve due to the complex shape and high degree of contact. Accordingly, it is not a great candidate for an MPP analysis because of the difficulties in performing a domain decomposition that balances the load fairly between processors. It is extremely difficult to balance the needs for equal element computation time and equal contact processing. The network communication is already a significant portion of the elapsed time because the model is physically small and each processor can finish its calculations very quickly.

This model was run using SMP-DYNA on a single CPU of System 3 (from Table 1). The analysis took 19.2 hours to complete. This model was also run on 1 and 2 CPUs of the MCC P-4 Linux cluster (system 2). The analysis completed in 5.57 and 2.96 hours respectively. For additional CPUs the model didn't scale well, and required more elapsed time than the 2 CPU case.

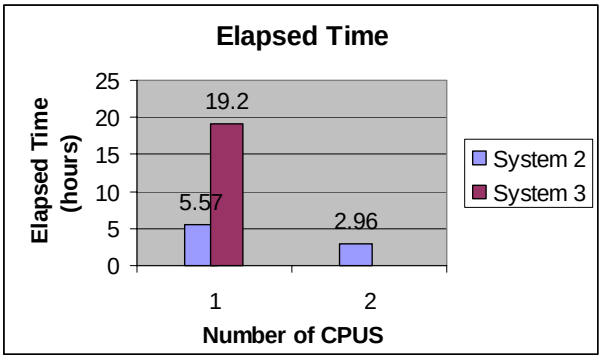

Figure 4. Elapsed time for a small seat slide assembly model run on two different computer systems using MPP-DYNA (system 2) and SMP-DYNA (system 3).

Case 2
This model was supplied by the same automotive seat supplier, and was a model of a full seat with a rigid dummy and seatbelts. The run simulated the testing required to meet FMVSS 210 [4] which evaluates the strength of the seat with the effects of a person restrained by seat belts, undergoing a frontal crash. This model had many initial penetrations of the seat belt assembly that were handled well by the SMP version of the code. However, these initial penetrations had to be removed before the MPP version could properly handle the analysis.

On the original Sun hardware (system 3, 2 CPUs), the analysis completed in 35.92 hours of elapsed time. On a single CPU of the MCC P-4 cluster, the model completed in 20.4 hours. Adding a second CPU dropped the elapsed time to 13.7 hours. This domain was also difficult to decompose adequately, because almost all of the contact occurred in the slide assembly beneath the seat. Geometrically, most of the elements were above the seat. Adding additional CPUs did not reduce the elapsed time significantly.

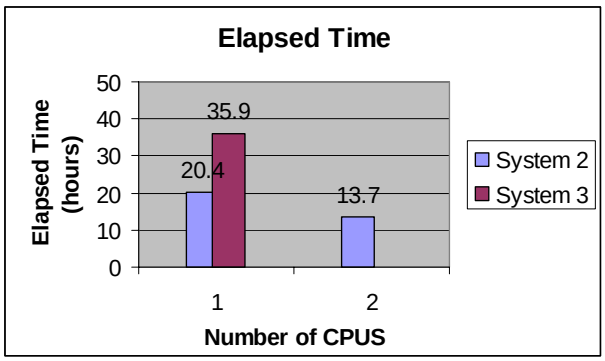

Figure 5. Elapsed time for a full seat assembly model, including seat belts and a rigid dummy.

Case 3
The third case studied was an automotive crash simulation of a mini-van (Figure 6). There were 376,703 elements in the model, which was a public domain model of the vehicle. Using SMP-DYNA, the model took 68 hours to complete on a single-CPU XP1000 (system 6). The MPP-DYNA run on system 6 took 55.5 hours using 2 CPUs and the default domain decomposition (Figure 6b), and 25.5 hours using 4 CPUs (Figure 7).

The model was run on 8 CPUs of the MCC clusters – both the P-3 and P-4 based systems. The model ran in 29 hours on the P-3 system and in 15.15 hours on the P-4 system.

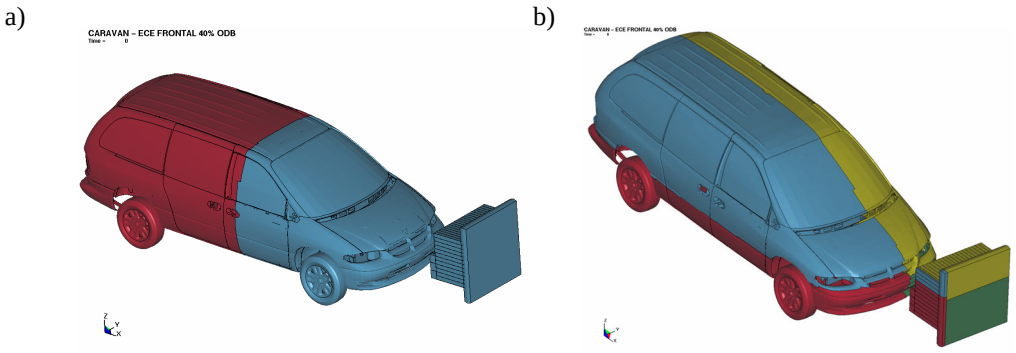

Figure 6. The mini-van model. The model contained over 376 thousand elements. a) the default decomposition obtained by subdividing the longest dimension is unsuitable for good load balancing. All of the contact occurs in the front domain. b) better results were obtained with the 4-CPU decomposition.

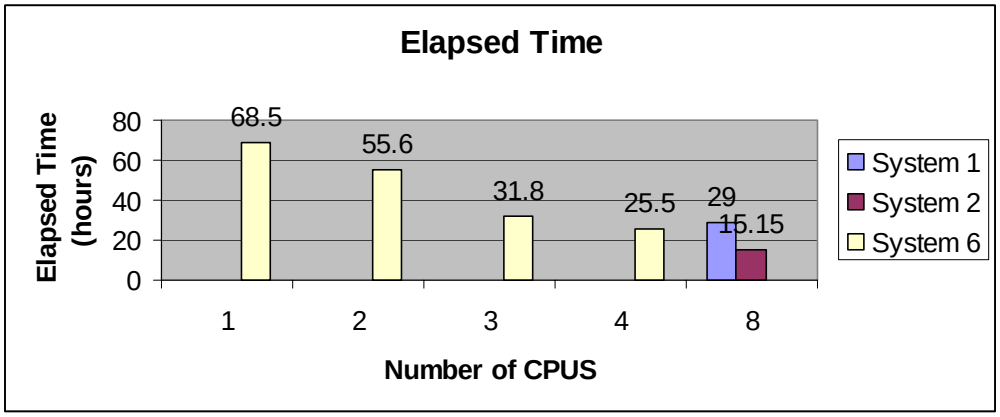

Figure 7. Elapsed time for solving the mini-van impact problem.

Case 4
A medium-sized (90,234 element) sheet forming model was studied. The part in question was an aluminum structural part (not shown) which was modeled without adaptivity. Originally, the model was run on an SGI Origin 2000 R12K system running at 300 MHz. The model was run on 1,2,4, and 8 processors and took 67.95, 33.62, 17.41 and 7.88 hours respectively to complete. Chu (2000) would suggest that the apparent super-scalability in comparing the 8-processor timings to any of the other models is probably explained by the size of the computing cache relative to the portion of the model being solved. As more processors are used, the amount of data handled by each computer decreases until eventually the data will fit entirely into cache. At this point, the computing efficiency jumps. It is also clear that the use of the MPP code on an SMP machine provides for excellent scalability at low numbers of processors.

Proving that scalability isn't everything, when the same model was run on the P-4 system it completed in 16.88 hours on a single CPU and 3.34 hours on 8 CPUs (Figure 8). Although the MCC cluster running across a local area network did not scale nearly as well as the larger SMP machine, it provided an answer more quickly at a fraction of the cost.

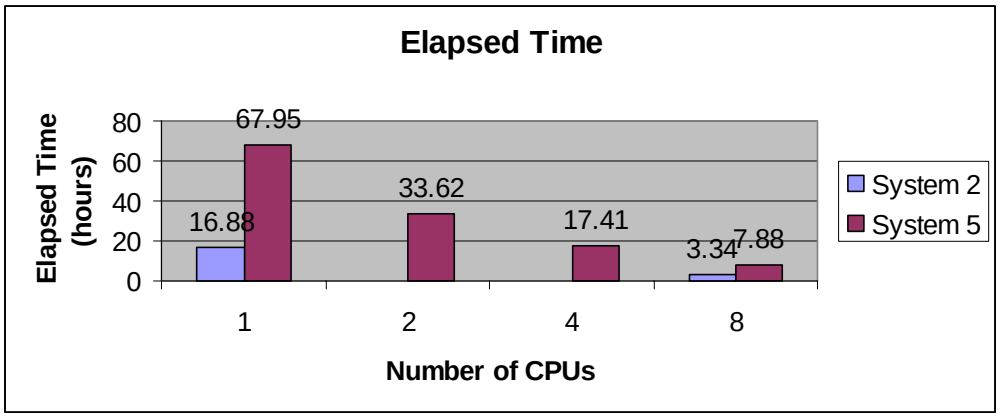

Figure 8. Elapsed time for running a medium-sized sheet forming problem on an SGI Origin 2000 and a Pentium-4 based Linux cluster. The SGI system (system 5) scales linearly, but the P-4 system (system 2) completes the analysis more quickly.

Case 5
In this model, a large sheet forming operation was simulated. The panel was a hood inner, and required 205,800 elements to describe the tooling geometry. The blank initially required 4800 elements, with a mean element size of 20 mm. The model was run adaptively, so at the end of the analysis, the model was comprised of nearly 320,000 elements. The originating hardware was the same SGI system for benchmark #4. On 8 processors of the SGI, the model ran in just over 61 hours. Using 8 CPUs of system 2, the model completed in 32.5 hours.

Case 6
Three successive stages of a hydroforming operation were simulated using LS-DYNA. During the first stage, the tube was bent. In the second stage, the dies were closed. In the final stage, end feed and internal pressurization caused the tube to expand inside the die cavity.

The models were initially run on an HP-J5000 system (system 4) with 2 CPUs running SMP-DYNA. Model times quoted were for a single-CPU run only because the operating system HP-UX 10.2 did not support parallel execution. For comparison purposes, the 800 MHz system (system 1) ran all three analyses on 1,2,4,6, and 8 processors. As seen in Table 2, it took two of the Pentium-III CPUs to match the speed of the single HP J-5000. It should also be noted that the cluster scaled almost perfectly for this application. Hydroforming, it turns out, is an ideal application for scaling of these clusters.

The reasons for this excellent scalability has to do with the nature of hydroforming tubes. The longest dimension in the model is along the length of the tube. If the domain decomposition proceeds as it should, then each subdomain consists of a length of the tube and the associated nearby tooling. This means that:

1. each subdomain borders on at most 2 other subdomains, and
2. the boundary nodes are oriented along the tube diameter, which is small compared to the tube length.

The consequences of these facts are that network traffic is minimal, because communication is required for a relatively small number of nodes and communication from one subdomain is with at most 2 other subdomains.

Table 2. Elapsed time (hours) to run 3 different hydroforming models. The HP models ran the SMP version of LS-DYNA. The MPP version was run on the P-III cluster. It took 2 of the 800 MHz P-III CPUs to match the speed of the HP J-5000 system. Note how well the cluster scales for hydroforming simulations. Based on previous benchmarks, it is anticipated that the P-4 cluster would run in about 50% of the time of the P-II cluster.

Model Description	HP J-5000 (System 4)	P-III system (system 1)				
	1 CPU	1 CPU	2 CPUs	4 CPUs	6 CPUs	8 CPUs
Tube bending	4.02	7.99	3.96	2.08	1.47	1.17
Die closure	1.30	2.24	1.30	0.72	0.56	0.47
pressurization	2.38	4.01	2.04	1.11	0.801	0.63

DISCUSSION AND CONCLUSIONS

The use of multiple processors has been shown to be an effective method for reducing the elapsed time required for finite element simulations. LSTC currently sells two versions of LS-DYNA capable of using multiple processors. Of the two versions, the MPP version has been shown to scale better over a larger range of processors.

Using MPP-DYNA on a cluster of Linux-based computers can provide super-computing performance at workstation prices. This technology, once found only in computer-science laboratories and at research facilities, is now available for the industrial user. It remains a challenge, however, to set up and maintain such a cluster. Medusa Computing Corporation was established in 2001 to provide this service to LS-DYNA customers.

From the results presented here, it is clear that over a limited number of CPUs, SMP machines scale much better than clusters distributed over a Fast Ethernet network. Having said that, the raw speed of the PC architecture is such that clusters of PC's will often beat the more expensive UNIX architectures. While the benchmark results presented here were comparisons between current generation PC's and older generation Unix workstations, it seems clear that for price/performance considerations it is hard to ignore the power of the clusters.

Scalability is a function not only of the computer hardware, but also of the problem being solved. Some problems lend themselves to effective domain decomposition. Others prove more difficult. Crash analysis is a particularly difficult problem to balance, because contact early in the analysis is very different from contact midway through. Hence an optimal decomposition at the start of the analysis may be a faulty decomposition later.

A good domain decomposition is easier to obtain for sheet forming problems, because contact can be more easily predicted. There are fewer contact interfaces, and the main problem is essentially a 2-dimensional one. The blank is

made of shell elements, so it is broken down into rectangles (at most 8 neighbors) and not cubes (with up to 26 neighbors).

The benefit of moving from 3-D in crash to 2-D in sheet forming is duplicated when moving to 1-D for hydroforming. Granted, hydroforming is not a one dimensional problem, but the domain decomposition quickly becomes a 1-D sort. The resulting problem scales so well because the number of neighboring domains is at most 2 and the ratio of boundary nodes to interior nodes is small. If enough processors are applied to the problem, the hydroforming simulation will again become a 2-D decomposition, but the elapsed time is likely to be so small at that time as to be insignificant.

With the ease-of-use of today's pre-processors, the power displayed by clusters of PC's, and their relatively low price, even small tool and die shops can start to embrace sheet forming simulations.

REFERENCES

1. Chu, R. and Li, G. (2000) "Scalability of LS-DYNA on SGI Systems". Proceedings of the 6th International LS-DYNA User's Conference. Dearborn, MI. (April 9-11) p. 17.35 – 17.39.
2.
3. DYNAFORM User's Manual Version 3.0. December, 1998.
4. Federal Motor Vehicle Safety Standards. **"**Child Restraint Systems; Child Restraint Anchorage Systems". Department Of Transportation, National Highway Traffic Safety Administration. 49 CFR Parts 571 and 596.
5. Federal Motor Vehicle Safety Standards. **"**Seat Belt Assembly Anchorages". Department Of Transportation, National Highway Traffic Safety Administration. 64 FR 29617; REG: 49 CFR Part 571.210.
6. Galbraith, P.C., Finn, M.J., and Bull, M.J. (1996) "Industrial sheet forming simulations: current capabilities and future requirements". High Performance Computing In Automotive Design, Engineering, and Manufacturing. Proceedings of the 3rd International Conference on High Performance Computing in the Automotive Industry. 539-556
7. Honecker, A. and Mattiasson, K. (1989) "Finite element procedures for 3D sheet forming simulation". Numiform89: Numerical Methods in Industrial Forming Processes. Thompson, E.G., Wood, R.D., Zienkiewicz, O.C. and Samuelsson, A. (eds.) A.A. Balkema. p. 457-463.
8. HyperForm 5.0 User's Guide, Altair Engineering., Troy, Michigan, 2001
9. LAM/MPI. http://www.lam-mpi.org
10. LS-DYNA User's Manual Version 960, Volume I. (2001) Livermore Software Technology Corporation. Livermore, CA.
11. LS-DYNA MPP User's Guide. (2001) LS-DYNA User's Manual Version 960, Volume II.. Appendix L. Livermore Software Technology Corporation. Livermore, CA.
12. MPICH. http://www-unix.mcs.anl.gov/mpi/mpich
13. Wertheimer,T. (1991) Numerial simulation of metal sheet forming processes. FE-Simulation of 3-D Sheet Metal Forming Processes in Automotive Industry. VDI Berichte 894 , 517-548.