TRANSITIONING FROM SMP TO MPP-DYNA3D FOR THE SIMULATION OF LARGE THERMAL-STRUCTURAL IMPLICIT PROBLEMS

Dr. Gurdip S. Kalsi AWE

AWE, Aldermaston, United Kingdom

1 Abstract

The LS-DYNA® family of codes have been used at AWE for many years. For a long time they were used on our shared memory platforms (SMPs) to carry out implicit structural and coupled thermal-structural analyses, amongst others. Over time processor speeds have continually increased and larger memory has become available at reducing costs. This has led to an increase in the size of models as meshes have been refined for better definition and realism of the problems under investigation. However the simulation of the long-term responses of engineering structures poses special difficulties when large models need to be analysed encompassing non-linear behaviours. These non-linearities can arise through sliding interfaces, and more commonly through the complex constitutive responses of non-traditional fabrication materials, such as foams and explosives, that act as structural, load-bearing components.

Unlike explicit analysis, in implicit problems the equations cannot be decoupled from each other, and so implicit simulations immediately make large demands on the amount of memory required to solve the problem in-core, and these requirements increase rapidly as the model is refined. For the most complex analyses the turnaround times can grow from weeks to potentially months, as model size increases. This problem is being addressed by re-writing implicit solvers to run in parallel mode on distributed memory platforms (DMPs). Although these developments have helped these codes to reduce turnaround times, work is required to further enhance their scalability.

Shared memory and MPI-versions of LS-DYNA have been used at AWE to investigate the transition from SMPs to DMPs for the solution of large, contact-dominated thermal-structural implicit problems. The hybrid version of these codes was also used in some simulations, but this is early work at AWE. This paper reports our findings. It also examines the influence of code characteristics on computing platform requirements. The significant reduction in turnaround time that was realised using MPI instead of the SMP version for a major test problem will be presented, and the scaling characteristics of the MPI and hybrid versions of LS-DYNA for this problem will be shown.

2 Introduction

In order to clearly present the work reported here, it will be convenient to use the following nomenclature for the codes used in analysis. ‘DYNA3D’ will be used to collectively refer to the earlier LLNL, and the successor LSTC versions of DYNA3D. ‘LSDYNA3D’ will be used when referring to the SMP version of LS-DYNA. ‘MPPDYNA’ will be used to refer to the parallel version of LS-DYNA, and ‘Hybrid-DYNA’ will be used for LSTC’s hybrid version of LS-DYNA.

The DYNA3D (i.e. the earlier LLNL and the recent LSTC) family of codes has been extensively used by AWE’s Engineering Analysis Group for nearly three decades. Until about a year ago, LSDYNA3D was our workhorse for simulating implicit problems, whilst MPPDYNA has over the years been used only for explicit analyses.

Historically, nearly all major implicit codes have started life as serial codes whilst explicit codes have always been parallelised. This is due to the fundamental difference in the formulations of implicit and explicit problems. The equations in explicit formulations are decoupled from each other, so that each single equation can be solved independently of all others. This greatly reduces memory requirements since large system matrices are not required to be set up, unlike the case of implicit formulations where the equations cannot be decoupled and must be solved simultaneously. The penalty one pays in using explicit methods is that the time step of integration is usually very small, and the solution is only conditionally stable. In implicit calculations memory requirements are large, but the solution is unconditionally stable thus allowing for much larger loadsteps compared to explicit analyses.

Over time, implicit analysis codes have developed from serial to shared memory parallel (SMP) processing codes, which allowed a small degree of parallelisation, but nothing comparable to the degree to which explicit codes can be parallelised. LSDYNA3D is such an SMP code, and in our experience it scales well to between 4-8 processors. The basic problem lies with the difficulty of parallelising direct solvers which are the preferred methodology for solving implicit problems, particularly those involving large degrees of non-linearity. Iterative solvers, which can be readily parallelised, can be used to solve implicit problems but they can face convergence difficulties in complex, contact-dominated, highly non-linear problems that involve a mix of element types in the mesh discretisations. However there is little evidence in published literature of the successful use of iterative solvers to analyse structural mechanics problems possessing these characteristics.

As the size of an assembly that is to be modelled with an implicit code increases, and as the number of components that make up the assembly grows, then the demand on computing resources begins to increase quite rapidly if the behaviour of the whole assembly needs to be simulated. Simulation of environments or loadings that excite non-linear material behaviour makes yet bigger demands on computing resources, and this need is further increased if geometric non-linearities also need to be accounted for. As assemblies or structures become more complex, or are to be used in mission-critical applications, then the need to gain greater insight into their behaviour under varied and complex environments becomes increasingly important. This usually requires a greater refinement of the mesh discretisations used to model such structures. As soon as meshes are refined, then the memory and cpu-time requirements start to increase rapidly, particularly for implicit problems that are simulated with direct solvers.

For long analysis durations, where each loadstep takes many hours of elapsed time, the total turnaround time with SMP codes can stretch from weeks to months. This situation is clearly intolerable, particularly if that analysis is only one of a series of simulations that need to be carried out in sequence. One must then look to ways of speeding up the implicit analysis.

As it happens, steady progress has been made by researchers over the last ten years or so in developing improved methodologies for parallelising direct solvers. This has enabled code developers such as LSTC, HKS, ANSYS, etc. to produce versions that scale much better than their SMP antecedents, and produce answers on much shorter timescales. MPPDYNA possesses parallelised implicit solvers, and this code was used for the work reported here. Similar parallel solvers such as ABAQUS and others will be investigated in the future at AWE to assess their capabilities in relation to MPPDYNA. The main reason for concentrating on MPPDYNA in this initial study is that it is a sister code of LSDYNA3D, and in theory our legacy finite element models that ran with LSDYNA3D should quickly be able to run with MPPDYNA requiring few modifications. However such expectations can often be misplaced. A second reason for using these two codes is that AWE has developed and added to them a user material model which is useful for modelling the creep behaviour of polymeric materials [Kalsi, 2009].

This paper reports on work which will inform the process of transitioning from shared memory platforms to distributed memory platforms for the solution of large contact-dominated thermal-structural implicit problems at AWE. The influence of code design on computing platform requirements is discussed, and some of the problems associated with large file sizes, and data transfer and storage are mentioned. The significant reduction in turnaround times, using MPP- and Hybrid-DYNA instead of LSDYNA3D, which was achieved for a major test problem will be shown.

3 Influence of Solver Methodologies on Computer Platforms

To support the different super-computing needs of the Engineering and Physics communities, AWE has provided two types of high-performance computing architectures, SMP and DMP, on site for a long time. This dual-platform arrangement was put in place to support the serial or limited-scalability SMP codes that were used by the Engineering Analysis community for its implicit analyses, and the highly scalable and parallelised codes used by the Physics community for their explicit calculations. Supporting and maintaining two different types of machines with different architectures and each with its own particular requirements is an obviously expensive arrangement, involving some duplication of effort. It would be preferable to have a bigger, single system that comprehensively caters for all users on site.

LSDYNA3D has historically been used on shared memory platforms at AWE, and the attraction of shared memory platforms is that even a single processor or core can address the whole of memory, if required. This is a legacy from the days of serial codes, and later a very useful feature for implicit codes that achieved limited parallelisation. As model sizes have grown, it has been necessary to acquire larger and larger amounts of memory. Two SGI machines were purchased in 2010, one with 2TB of memory, and the other with 3TB. This enabled increasingly larger models to be analysed, and allowed for a number of analyses to be run concurrently. However, a point is reached at which although the memory is more than adequate, it is the turnaround times that become unacceptably long. This point was reached when the largest model sizes were of the order of two million elements.

In itself this model size is not very large: what increases run-times are the complexities introduced by the presence of a large number of contact interfaces, the activation of highly non-linear rate and/or temperature-dependent material behaviours, the coupling of thermal-structural interactions, and the total time for which the simulation needs to be run. When the total simulation is composed of a series of sequential analyses, each of which can take weeks to months, then the situation is untenable. The only thing to do on SMP platforms in that case is to reduce model sizes, which is unacceptable since this will lead to decreased solution accuracy.

AWE's distributed memory platforms have thousands of cores, but each core typically has only a limited amount of memory attached to it, between 2GB-4GB. These are optimal machines for carrying out transient, explicit calculations. As mentioned earlier this is because the explicit equations can all be decoupled from each other, and each core needs to address only a small amount of memory at any one time, independently of all other cores.

4 Test Runs with MPP- and Hybrid-DYNA

Test analyses were first run using MPPDYNA on our DMP machines, and later some were repeated with Hybrid-DYNA. Each node in our DMP platforms has 16 cores, each of which has an attached 4GB of memory, i.e. a total of 64GB/node.

In running implicit problems with MPPDYNA, there are two limits: as more and more cores per node are used for analysis, the memory available on the node gets split into smaller and smaller blocks. Ultimately a limit is reached where the memory cannot be sub-divided any further without the analysis aborting. Equally, as more and more nodes (i.e. cores) are used for analysis, the system overheads increase, to the point that any further increase in core-count actually increases the turnaround time, i.e. one has gone beyond the optimal computing resource allocations for that problem. So for any particular problem there is a sweet spot that provides optimal turnaround times.

For smaller models all of the 16 cores on a node can be used to run MPPDYNA. However as the model size grows, it is found that each core must be given a minimum amount of memory otherwise the analysis fails to run - this means that not all available cores can be used for analysis, and some will remain idle, thus resulting in wasted compute cycles. There is no clear or reliable guidance on how much this memory should be, it appears to be problem dependent and has to be determined by trial and error. This is something that could be improved in the code. This minimum memory requirement has quite a surprising impact on the amount of computing resources required to run a large-sized model.

A few years ago the author created a series of test problems prefixed 'CYLxxx'. 'CYL' is a descriptor for the model which consists of a series of co-axial cylinders. 'xxx' denotes the approximate model

size in terms of the number of eight-noded brick elements that are used in the finite element mesh. The purpose of creating these models was to generate an unclassified finite element model that could be transmitted for use outside of AWE for comparing platform and code performances, using an example that typified our code feature usages without being too complicated. The applied displacement boundary conditions ensured that the resulting relative motions would properly exercise the cpu-intensive contact logic algorithms.

Figure 1 shows a view of the 'CYL1e6' model, which consists of 6 co-axial cylinders, and has 1026432 nodes and 921600 elements. The boundary and loading conditions are also described in Figure 1: all the cylinders are fixed at their bases, there are internal and external pressures applied, and cylinders 2 and 6 are subject to a positive time-dependent axial displacement, whilst cylinder 4 is given a similar negative displacement. All materials are modelled as elastic. Sliding-with-voids contact surfaces are defined between contacting cylinders.

Different sizes and versions of these 'CYLxxx' generic model have been sent out from AWE to those interested in using it to study code or platform performances [Wang, 2011; Lin, 2012], e.g. CYL2e6, CYL4e6, etc. Usually the size of the model is increased by lengthening the cylinders, whilst at the same time maintaining the previous element sizes. Examples also exist where this is done by increasing the number of co-axial cylinders instead. Non-linear materials are also used in more complicated tests, and coupled thermal-structural models have also been used in simulations.

Analysis of the CYL1e6 model using LSDYNA3D on a shared memory platform took over 1200 minutes for 5 loadsteps. The same model was analysed using 16 nodes but an increasing number of cores with MPPDYNA, and the turnaround times for these analyses are shown in Figure 2. The optimal numbers of cores to use for this problem is 192, and with this arrangement the quickest MPPDYNA analysis runs 12 times faster than LSDYNA3D on our large-memory SMP machine. Clearly, although 16 nodes are used in all analyses, not all cores are being used in all the runs. To recover these lost compute cycles, and to investigate the performance of Hybrid-DYNA, a number of analyses were run using this code. These results are also presented in Figure 2. Hybrid-DYNA runs in MPI mode between nodes, but in SMP mode on the processor boards within a node. This functionality enables the recovery of the compute cycles which are lost by the idle cores mentioned earlier when running MPPDYNA.

Comparison with MPPDYNA results shows that the Hybrid version runs faster when using the same number of cores. In fact, the results show that running on 10 nodes (i.e. 10x16 cores) Hybrid-MPPDYNA completes the job in 72.5 minutes, whilst MPPDYNA takes 91.7 minutes on 16 nodes. Much more telling is the data presented in Table 1, which shows the number of nodes used to run the Hybrid-MPPDYNA tests. So instead of running only one analysis across 16 nodes with MPPDYNA, one could potentially run three Hybrid-MPPDYNA analyses on 18 nodes, or 2 analyses on 16 nodes, in roughly the same elapsed time or faster, thereby very significantly increasing throughput of work. This capability is a real boost in comparison to the way LSDYNA3D was performing at AWE until very recently.

Our recent work shows that Hybrid-MPPDYNA provides a huge advantage over MPPDYNA when running large implicit problems. Consider the case of THERM_CYL6e6 (which failed to run on SMP machine with LSDYNA3D) which is a model that consists of approximately 6 million solid elements and is set up for a coupled thermal-structural analysis. It was found that out of the 16 cores available per node, only 2 cores could actually be used for computations since the analysis would not run without each core being allocated at least 24GB of memory. So running on 1 or 2 nodes is almost the same as running LSDYNA3D on an SMP machine, and takes a prohibitively long time to complete even a single loadstep. Apart from this, the remaining 14 cores are being wasted, unable to contribute to the computations. So to reduce turnaround times, it became necessary to run the problem on a larger number of nodes.

Figure 3 shows the reduction in turnaround times for this test problem with increasing numbers of cores, with only 2 cores per node being used for computations. It is pointed out that the number of cores actually reserved for MPPDYNA was **eight** times the numbers shown in this Figure. Note that the run times shown are for a single loadstep only. Table 2 lists the number of nodes and cores used per analysis. In going from 100 to 600 cores that were actually used for calculations, the turnaround time was reduced by roughly 3.

This test example raises a number of interesting issues. Firstly, as shown in Table 2, only 2 of the 16 cores available on a node are actually doing any calculations, the remaining 14 are idle. Secondly, as the number of nodes is increased in order to reduce run times, a bigger and bigger fraction of the super-computer is taken up by a single application. Thirdly, for long duration simulations, this fraction of computing resources is unavailable to other users for long periods of time, which degrades overall system performance; this happens because compute nodes are generally locked for the use of only one user, and sharing with others is not permitted. This situation is made worse if a number of such large implicit calculations need to be run concurrently.

One solution to this is to have so-called 'fat' nodes making up a fraction of the super-computer. Fat nodes are those that have bigger amounts of memory per core, e.g. 8, 16, 24GB, etc., compared to the standard 2-4GB. In that way fewer nodes out of the total available would be occupied by any single implicit calculation, and many fewer computational cycles would be lost from the system.

One THERM_CYL6e6 analysis, also indicated in Figure 3, has so far been run using Hybrid-MPPDYNA, and more will be run and reported in the future. This analysis was run using only 16 nodes, and for this configuration Hybrid-DYNA went out-of-core because of insufficient memory. Even so, it completed the analysis in about 225 minutes, which is roughly twice as long as running MPPDYNA on 300 nodes. Running in-core on a larger number of nodes will reduce this turnaround time, and as alluded these studies have not yet been done. So for waiting a little longer, it is possible to make huge gains by running implicit analyses with Hybrid-DYNA. For example nearly twenty Hybrid-DYNA analyses could be carried out in parallel using 320 nodes instead of only one on 300 nodes with MPPDYNA without major differences in elapsed times. This is a huge performance increase over MPPDYNA when analysing such large and complex implicit problems.

5 Comments on the Constraints Associated with Using Commercial Codes

In an ideal world, in establishments such as ours one would have access to in-house analysis codes so that model or code-related bugs could be debugged as rapidly as possible, especially where the finite element models cannot be released to outsiders for reasons of commercial or security sensitivities. Otherwise unpredictable delays can arise.

Given the worldwide user bases that run commercial codes such as MPPDYNA and other similar codes, it is generally the case that these codes are essentially bug-free for the majority of common applications because of the constant interactions with that user community. However it can happen in state-of-the-art, continually advancing codes, that the less-frequently used features when used in particular or rare combinations, may hit latent bugs. Whilst rigorous QA testing is regularly performed by code vendors, they usually do not test their codes on long duration, very large complex problems which may need to be run over hundreds of cores – and they cannot really be expected to do so. So as in our experience, situations can arise where a smaller model of the same problem will run successfully, but a larger one encounters difficulties due to yet-undiscovered code bugs.

Significant delays can occur when a large model fails to run successfully: the code vendor will not release the source code to the user, and the user cannot release the finite element model to the code vendor because of commercial or security sensitivities. The complexity and size of our actual model is such that it is not easy to create exactly the same conditions in a non-sensitive model that could be passed to the code vendors. Significant efforts were made to create such equivalent models, but the errors could not be replicated with these. In this situation it becomes very difficult to isolate and then correct the code bugs, and the time delays that arise can be long.

The method adopted to help debugging at AWE was for the vendor to supply an executable of the code with print statements embedded near to the suspected source of the error. The printout from an abortive analysis with this code was then returned to the vendor, to get an updated executable with print statements moved further on. The process was then repeated. In this instance it took over three weeks to fix one particular bug. It is to be expected that similar situations may arise in the future as we explore the complex multi-physics capabilities of MPPDYNA and other commercial codes, compared to those we have already exercised for many years in LSDYNA3D. It is worth recording here that LSTC provided very good, high quality support in our de-bugging exercises.

6 Results of Our Investigations

The baseline finite element model which was analysed with MPPDYNA consisted of over 1.5 million elements, and simulated a coupled thermal-structural problem dominated by extensive contacting regions. A test simulation was carried out over a period of 100s, taking 20 variable-sized loadsteps, with the structural loads being ramped up over 1s, whilst temperature changes occurred over the full 100s.

The reduction in turnaround time with increasing core count, running MPPDYNA on 16 nodes, is shown in Figure 4. With LSDYNA3D running on our SMP machine, this analysis took over 1600 minutes. So the speedup that MPPDYNA provided for this analysis in comparison to LSDYNA3D is a factor of about 10. The run-times for analysing this problem with Hybrid-DYNA are also shown in Figure 4 and indicate that for this problem Hybrid-DYNA runs slightly slower than MPPDYNA until it is run over six nodes or more. The quickest turnaround, in the analyses shown in this Figure, is with Hybrid-DYNA running over 8 nodes. It is important to point out that MPPDYNA could not run this problem on fewer than 4 nodes, whilst Hybrid-DYNA was able to run on even as few as 2 nodes, running only about 50% slower than MPPDYNA on 16 nodes. This again emphasises the efficiency and economy of Hybrid-DYNA over MPPDYNA, as described in the earlier examples.

This very significant reduction in turnaround times for implicit problems now makes it possible to plan extensive, chronologically sequential series of analyses that previously could not have been undertaken because of the very long turnaround times with LSDYNA3D.

Demonstration of successful solution of large, implicit problems on distributed memory machines suggests that there is little need in future for buying expensive, shared (large-) memory platforms, so that a single machine architecture can be envisaged for future use. Whether ‘fat’ nodes are a part of this architecture is an open question at the moment.

Since Hybrid-DYNA now provides much faster turnaround times with great efficiencies, there is a natural tendency to increase mesh refinements to produce higher quality representations of actual structures. This will result in very large results files being produced. As an example, even with the relatively small 1.5 million element model, one complete set of results files was sized at 15TB. Post-processing of such large data sets to extricate critical data values is a time consuming, laborious business. Moving files this large across disks is another time-consuming task. File sets this large, resulting from only one of a set of many such analyses, also makes exacting demands on the backup and permanent storage facilities. In fact, the ability to perform the types of large implicit calculations now possible with codes such as MPPDYNA means that the supporting computing infrastructure, such as networks, disk sizes and connectivities, storage silos, etc., also needs to be carefully reviewed and optimised for maximum throughput and greater user convenience.

7 Conclusions

In our experience, moving from LSDYNA3D to a highly parallelised MPPDYNA version has not been a straightforward exercise. In retrospect it was probably unrealistic to expect a smooth and quick transition from the one code to the other in view of the complexities of our analyses, on top of the major code modifications required to develop the required implicit parallel capability, which should not be underestimated. Further work is needed to ensure that more efficient use is made of available memory by MPPDYNA.

We investigated Hybrid-DYNA after initially running our test problems with MPPDYNA, which had itself shown huge improvements over LSDYNA3D. The performance of Hybrid-DYNA has easily outpaced that of MPPDYNA for our test problems, and it is a much more efficient code in the way it employs computing resources. Hybrid-DYNA shows faster turnarounds whilst using a lot less resource, permitting a number of parallel analyses to be run on the same computing resources that MPPDYNA would have needed for a single run.

The ability to perform large-scale analyses using parallel codes such as Hybrid-DYNA on DMPs is a very significant advance in the simulation of implicit, mission-critical, highly non-linear, contact-dominated coupled thermal-structural analyses. The shorter turnaround times that are now achievable means that problems which previously would have taken weeks to months using SMP codes can be analysed within hours to days. This means that meaningful progress can be made for example

towards demonstrating mesh convergence. Sensitivity studies that previously could not be undertaken can now be planned for completion within acceptable timescales. Equally, real-time detailed design support can be provided for urgent requirements where complex models are readily available.

The fact that Hybrid-DYNA solves complex, implicit problems successfully and much faster on distributed memory machines means that expensive large-memory, SMP platforms which were previously needed to run LSDYNA3D will no longer be required in the future.

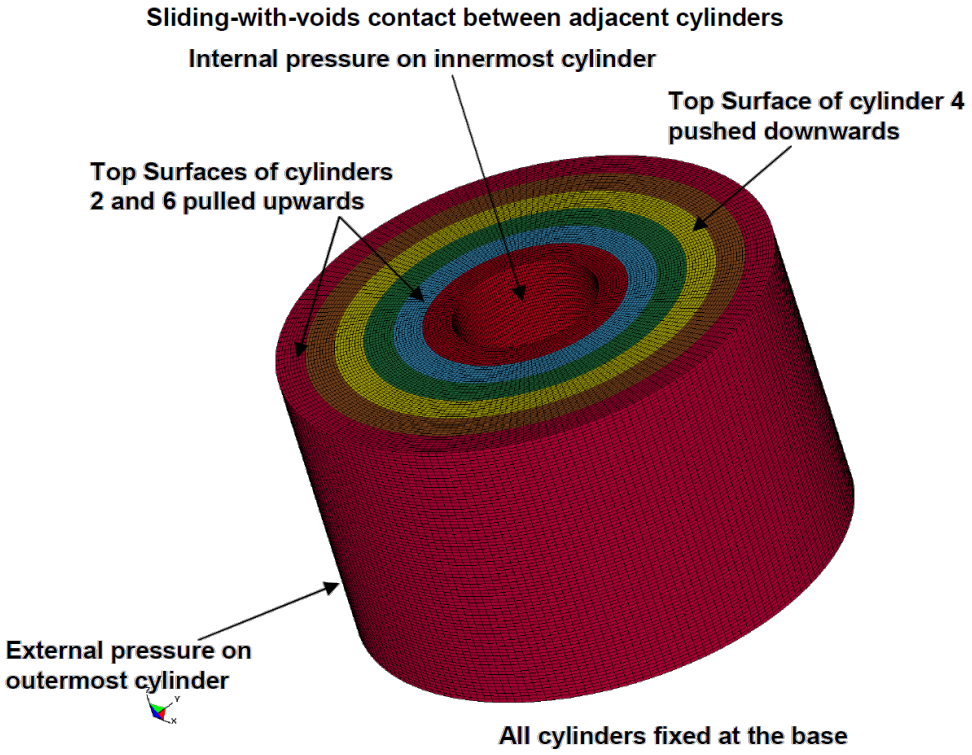

Fig. 1: Details of the Generic CYL1e6 Finite Element Model

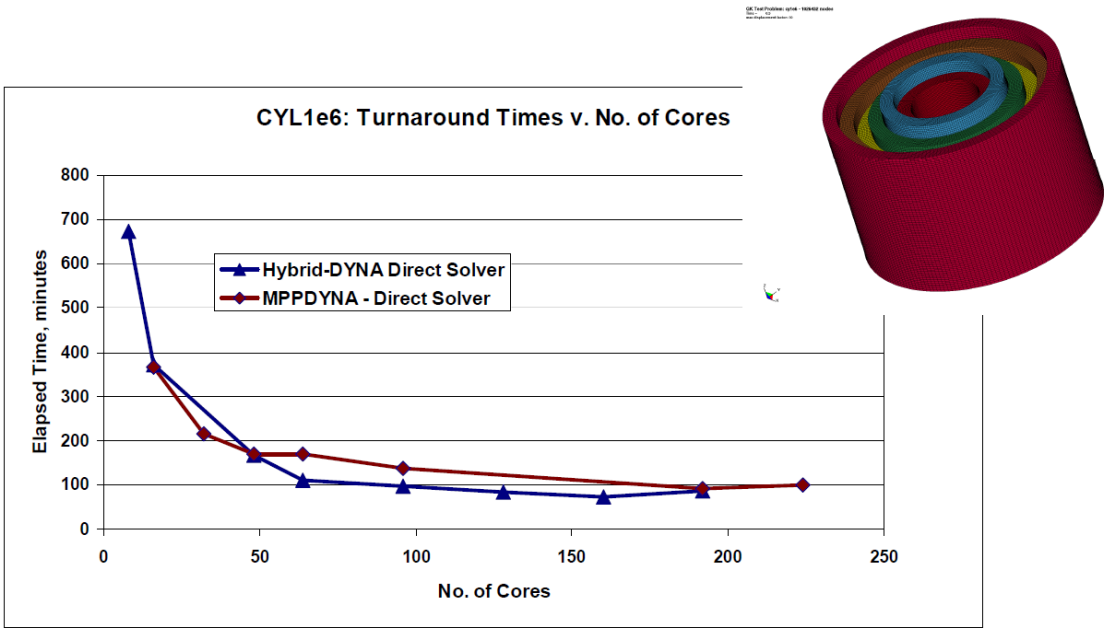

Fig. 2: MPP- and Hybrid-DYNA Scaling Behaviour with CYL1e6 Model

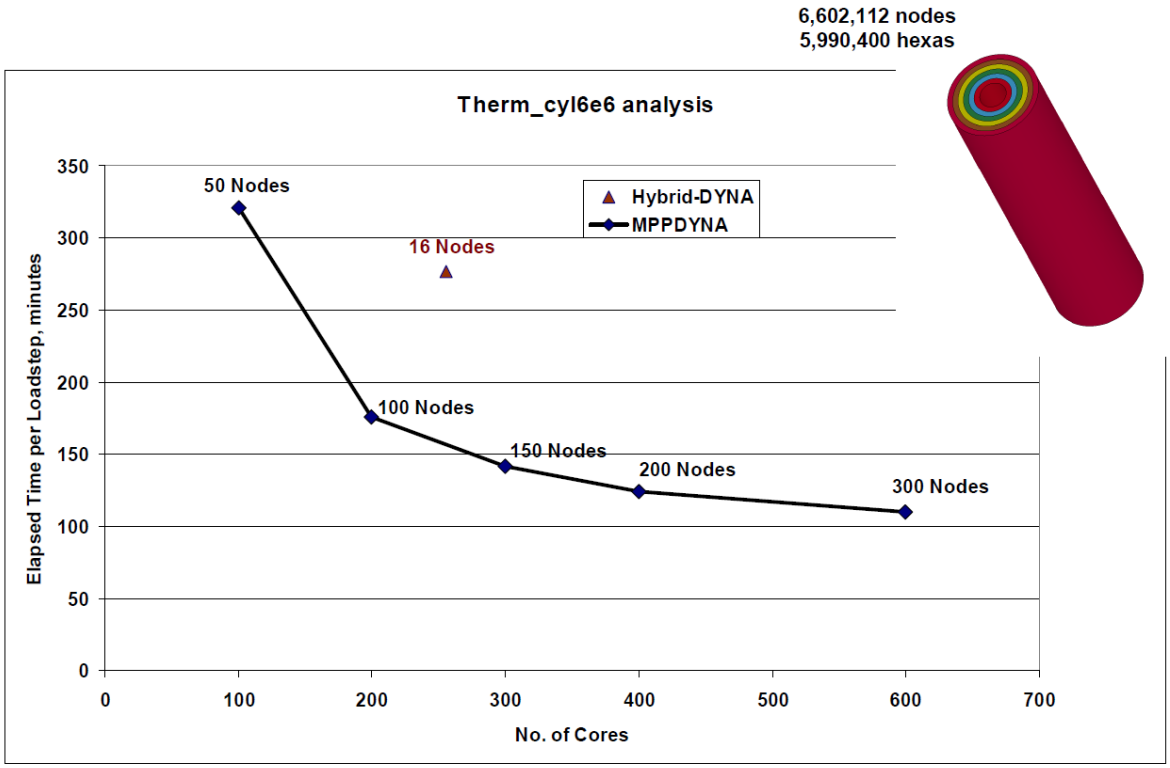

Fig. 3: MPP- and HYBRID-DYNA Run Times with Therm_CYL6e6 Model

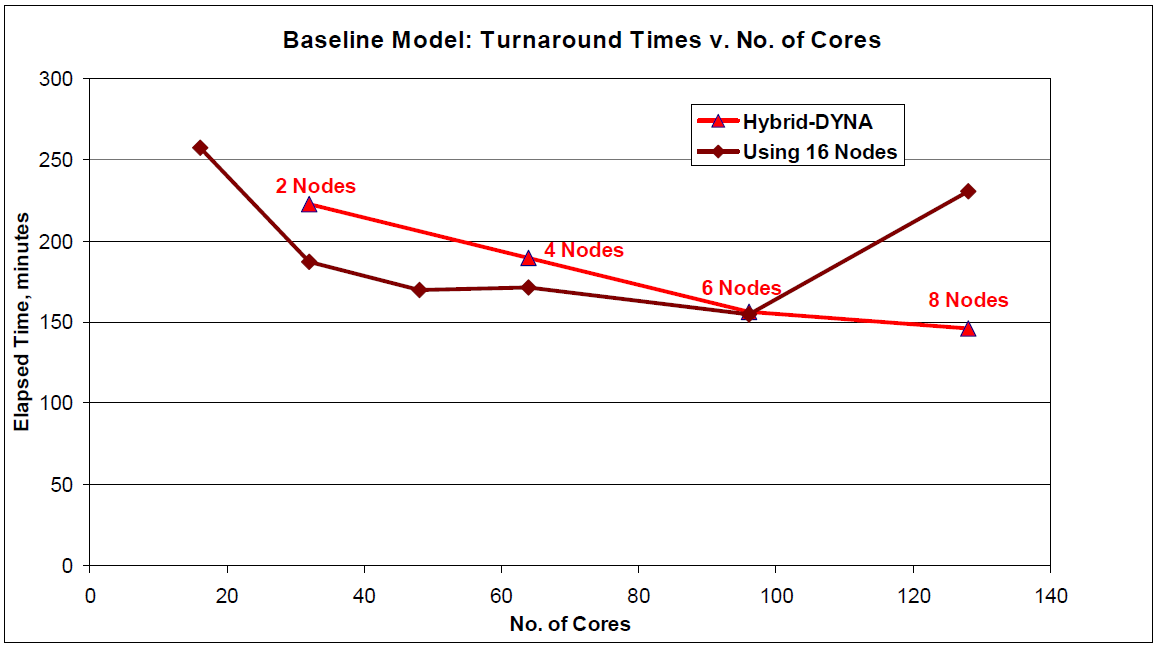

Fig. 4: Baseline Test Model: MPP- and HYBRID-DYNA Run Times and Computing Resources Used

CYL1e6 MODEL

CODE	No. of Nodes Used	Elapsed Time, Mins.
Hybrid DYNA	4	109.8
	6	97.2
	8	82.3
	10	72.5
	12	85.8
MPPDYNA	16	91.7

Table 1 : MPP- and Hybrid-DYNA: Comparison of Computing Resources Used in Analysis

No. of Nodes Reserved	No. of Cores Reserved	No. of Cores Used Per Node	Total No. of Cores Actually Used	Memory Used for MPPDYNA per Core, GB	Total Memory Specified for MPPDYNA, TB	Turnaround Time for 1 Loadstep, Minutes
50	800	2	100	24	2.4	321
100	1600	2	200	24	4.8	176
150	2400	2	300	24	7.2	142
200	3200	2	400	24	9.6	124
300	4800	2	600	24	14.4	110

Table 2 : Computing Resources used by MPPDYNA to run Therm_CYL6e6 Analyses

8 References

1. Kalsi, G.S. 2009. Modelling the Creep Response of a Polymer Bonded Explosive. NAFEMS World Congress, June 16-19, Crete, Greece.

2. Wang, J. 2011. Hybrid (MPP+OpenMP) version of LS-DYNA. LS-DYNA Update Forum in Filderstadt, Germany, Oct. 12-13.

3. Lin, Y.Y. 2012. A Comprehensive Study on the Performance of Implicit LS-DYNA. 12th International LS-DYNA Users Conference, Dearborn, Michigan, USA.

9 Summary

MPPDYNA and Hybrid-DYNA were used to reproduce large, complex implicit analyses previously carried out using LSDYNA3D and have reduced turnaround times by factors between 10 and 20. This is very welcome, but it was found that MPPDYNA makes increasingly inefficient use of computing resources as model sizes increase. Hybrid-DYNA is generally faster and very significantly more efficient, and is recommended for the simulation of large, contact-dominated implicit models in preference to MPPDYNA.