Optimising Run Times for Sheet Metal Forming Simulation

Trevor Dutton
Dutton Simulation Ltd

Annika Weinschenk
Hexagon MI

Abstract

Many practitioners of sheet metal forming simulation work in the lower tiers of the manufacturing sector where they tend to rely on PC technology (such as desktop workstations or laptops) to solve their models. Fast turnaround of analyses is critical to success during the process engineering and tool design stage so optimising the model run time is very important.

Most PC hardware now provides multi-core chip technology as standard so this paper examines the differences in run times with different numbers of solver cores across a range of model sizes in order to establish best practice, and indirectly best value for money when investing in both simulation software and hardware.

The paper considers a selection of LS-DYNA® SMP and MPP solvers, examining the scalability across different numbers of cores (up to 16 maximum) and the interaction between solver scalability and the use of adaptivity – the differences from running with a fixed, uniform, small element size vs. an initial large element size mesh with varying levels of adaptivity are described.

The method of mass scaling (standard vs selective) is also examined, to determine the influence of time step on both run time and results. A number of other factors that can influence run time are also reviewed, including adaptivity parameters such as fusion settings, and the type of storage disk hardware used (HDD vs SSD).

A number of recommendations are offered, based on model size and available hardware, in the hope that workers in the field of sheet metal forming will be able to efficiently apply the LS-DYNA solver for this important and widespread application.

Background

Sheet metal forming simulation has become an important application for LS-DYNA. This type of simulation work is undertaken by a wide range of companies, from large organizations, such as vehicle manufacturers, down to very small enterprises involved in tool design and manufacture. While the larger companies may have access to extensive computing resources, enabling them to run simulations in a short period of time using tens or even hundreds of cpu cores in parallel, this may not be the case for the smaller companies.

In addition, there are a number of competitor solutions in this field, some of which make a particular virtue of a quick solution time. This may appeal to the hard-pressed tooling engineer, keen to have their design signed off as quickly as possible, but we should bear in mind that international benchmarks such as NUMISHEET [1] have regularly shown that the result accuracy provided by LS-DYNA is higher. How then can we provide the fastest simulation turnaround time for such smaller organizations using PC technology to allow them to get the best possible results using LS-DYNA?

This paper sets out to explore the key considerations affecting run time when simulating sheet metal forming with LS-DYNA using a modest compute resource – perhaps a small workstation with one or two chips with four to eight cores per chip, or even an i7 laptop with no more than four cores in total – all running on MS Windows platforms. For many tooling engineering companies, the investment in hardware required for more powerful systems (as well as the greater expertise needed to run, say, a Linux cluster) may be a limiting factor – rather than the software licence fee which is structured to encourage use of multiple cores per job.

In order for companies with limited computing resource to make use of the full power of the LS-DYNA solver we would like to determine which approach to parallel processing is the most effective for sheet metal forming simulation. We need to explore the interaction between parallel processing with options regularly used in sheet metal forming simulation such as adaptive re-meshing and mass-scaling. We also need to consider the range of model sizes to be run – what may be an optimal solution for a small, shallow panel may not be optimal for a very large, deep, complex panel. Hence, we hope to be able to establish optimal settings and provide guidelines for efficient – but still suitably accurate – solutions across the range of potential problems faced by the sheet metal forming simulation engineer.

The following sections of this paper detail the investigations carried out to attempt to provide a set of optimal settings to run LS-DYNA efficiently in this application. The paper considers the interaction between SMP and MPP running on a number of cores (up to 16 maximum) with the use of adaptivity, including key settings such as number of levels and the initial element size. These options also interact with the other main parameter used for reducing run time – mass scaling; both standard and selective mass scaling were considered. Velocity scaling is also used – i.e., the tools move faster than they would do in reality – but this was not varied during the studies reported here. Other factors were also reviewed – including the use of adaptive fusion and also the effect of running the model on different storage hardware (traditional spinning hard disk drive (HDD) vs solid state drive (SSD)).

Software & Hardware

A number of versions of LS-DYNA were used for the tests. Four builds of MPP/LS-DYNA were run – R10.0, R10.1, R10.2 and R11.0.0 – chiefly to see if there were any notable differences between them in terms of both runtimes and results. These versions were the most recent available for testing at the time of this study (although not all were available commercially). In all cases the Intel MPI builds were used for MPP runs. Just one version of SMP DYNA was used – R10.1. All builds (MPP and SMP) run were single precision.

The workstation used was in fact quite an old machine (purchased in 2013) but perhaps not too different from that typically used in tooling engineering companies. The workstation had two Intel Xeon E5-2687W chips running at 3.1GHz and offering up to eight cores per chip, giving a maximum of 16 cpu available for the tests. A total of 32GB of DDR3 RAM was installed. The computer operating system was Windows 7 Professional (64 bit) – hyperthreading was not used.

The test parts were run on 1, 2, 4, 8 & 16 cores; however, not every combination of LS-DYNA build and core count was run for every case, particularly when it came to the largest of the three panels due to run time constraints.

Simulation Models

Three parts were selected for the study. These covered a range of sizes and depths of form as well as relative complexity in terms of geometry features as these are the main aspects controlling the solve time, directly affecting the number of elements, their size and hence the acceptable timestep. All three panels are made using a draw process. The models were created using FTI's FAST Incremental interface for LS-DYNA.

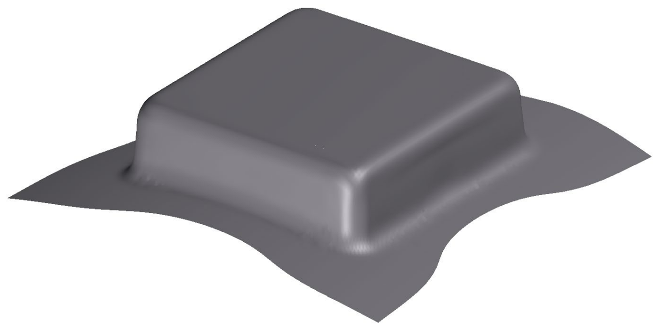

Figure 1 Part 1

Part 1 (see Figure 1, tools blanked for clarity) was a relatively small and shallow box geometry with an initial blank size of 140mm square, thickness 1mm. The final depth was approximately 25mm and the die and punch radius were reasonably generous. The forming simulation for this part ran in less than five minutes on a single cpu core, even without use of symmetry, with an initial element size of 2.2mm with 2 level adaptivity and 45 cycles. The panel formed with no formability issues.

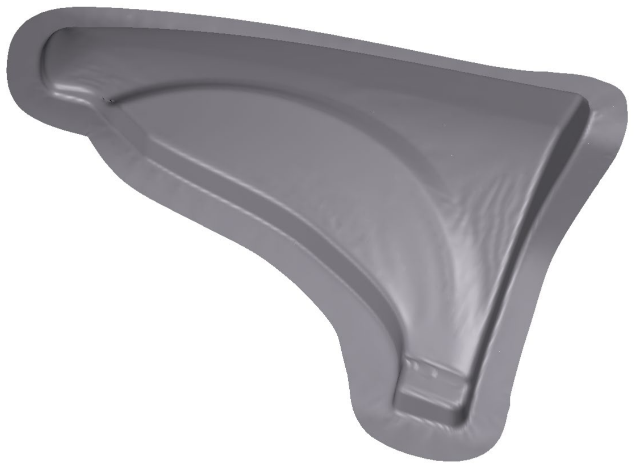

Figure 2 Part 2

Part 2 (see Figure 2) was a front fender panel formed using a developed blank size with maximum dimensions of 1330mm x 1065mm, thickness 0.9mm. The final depth was just under 100mm at the deepest point. While this panel was much larger than Part 1, it was run with a large initial mesh size (20mm with 3 level adapt and 68 cycles), representing an early feasibility development model rather than a final tool design, so still had a run time of under twenty minutes on a single cpu; it should also be noted that the initial binder wrap was modelled using the FTI one step method to save time (a useful option with FAST Incremental). This panel had some wrinkling and a split which created some instability in the results.

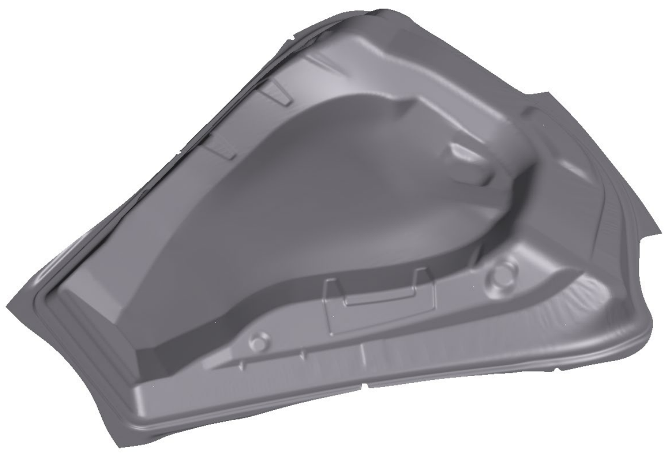

Figure 3 Part 3

The final panel, Part 3 (see Figure 3), was the largest and most complex of the three with a number of local fine details that required use of more accurate settings (8mm initial element size, 4 adapt levels and 180 cycles). The panel was formed using a developed blank with maximum dimensions 1120mm x 1020mm, 0.8mm thickness; maximum final depth was ~80mm. The panel was actually designed to make left and right hand parts double-attached so was symmetric, but to push the limits of the simulation it was run as a full model and did not take advantage of symmetry in this case – clearly, in practice one would always do so with a panel like this. The run time was therefore many hours for this panel.

Scalability Results

The run times for each part are presented in the following paragraphs. Results for (in most cases) and a selection of MPP codes as well as SMP R10.1, running on various combinations of cores from 1 to 16 are shown, all based on the original model setup (initial mesh size, adaptivity settings, etc.).

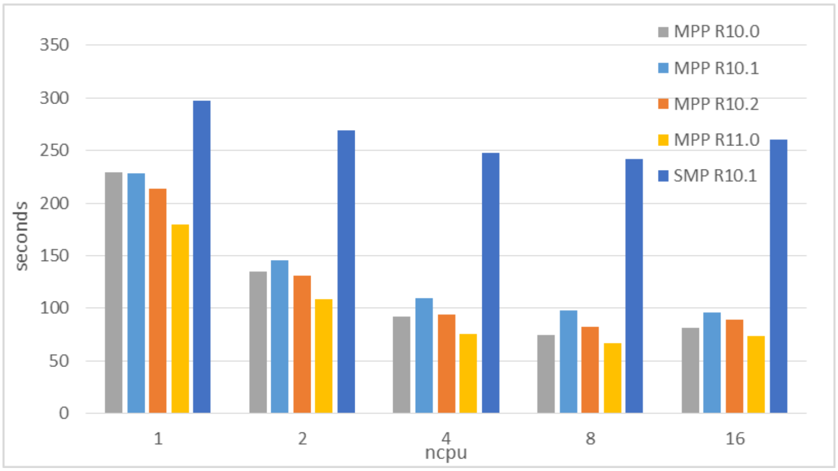

Figure 4 Run times for Part 1

Run times for Part 1 are shown in Figure 4. For the four MPP solvers, good speed up can be seen moving from 1 core to 2, and reasonable speed up from 2 to 4. There is only a small improvement going to 8 cores, and with 16 the times actually increase again. Of the four MPP codes, R11 always shows the fastest run times (yellow bars).

SMP run times (dark blue bars) are in all cases slower than MPP, even on one core, and the scalability on SMP is also much worse.

The cpu percentage usage for this small model shows that a lot of solver time is spent on initialization and decomposition and also on input and output, and relatively little on element calculations. As the model adapts, the analysis must be stopped with the current state written to disk and then re-initialized, including a new decomposition to try to maintain good load balancing between domains. All of this is done on a single core. This likely explains why scalability tails off with increasing number of cpu. In this case the blank element count increased from about 4,000 to just over 10,000 at the end – even with just two levels of adaptivity.

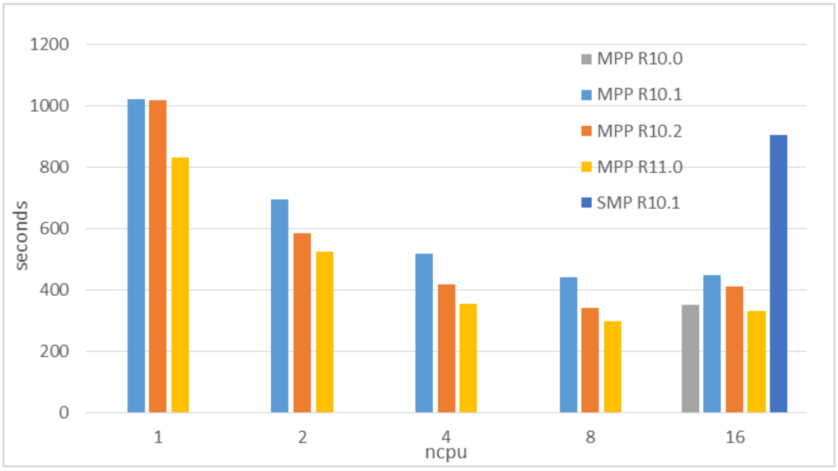

Figure 5 Run times for Part 2

Run times for Part 2 are shown in Figure 5. This model takes four to five times longer to run than Part 1 – it actually starts with fewer elements (~2,500) but increases to almost 25,000 by the end of the form. The results for the MPP solvers are similar to Part 1 with good scalability initially that tails away with increasing number of cpu. Version R11 of the MPP solver was again the fastest of those used; and SMP (only run for the 16 cpu case) was again considerably slower.

The likely reasons for the trend in these results are similar to those for Part 1 – inefficiencies in the initialization and re-balancing of the model at each adaptive cycle. However, in both this case and for Part 1 it is worth noting that the overall run time is quite short so if the analyst chose 8 cores instead of 4, or even 16 cores instead of 8, they would not be too troubled by the poor scalability, given that they would still get their results in a short amount of time.

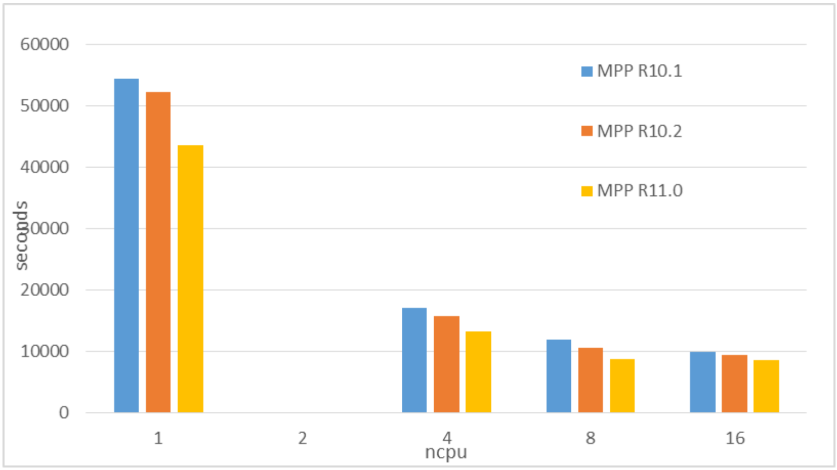

Figure 6 Run times for Part 3

For Part 3, the run times were considerably longer (see Figure 6), with even the fastest options used requiring around 2.5 hours to solve, so fewer cases were run because of the cpu time required. The model started with ~11,000 blank elements but this increased to over 400,000 by the end of the form. It is immediately clear that the scalability for the MPP solvers is better than with the smaller/faster running models, especially with 4 cores (scale factor = ~3.3 here); even 16 cores shows some scaling compared with 8. SMP was not run on this part.

The result for Part 3 suggests that, for larger (and arguably more typical) sheet metal forming models, there is a definite benefit in using 8 or even 16 cores. MPP R11 is once again fastest although for this version the 8 to 16 core scaling is modest; the best scaling is with R10.1, with a factor of 1.2 – which is still not large compared with a theoretical factor of 2. The reasons for reduced scalability are again thought to be initialization cost from adaptivity. However, even a factor of 1.2 equates to a time saving of more than 30 minutes for this model.

In summary from the initial results for all three parts, we can say that:

- MPP appears to scale better than SMP, which appears to be slower in all cases
- Scalability is better with larger models than smaller ones, although small models run quickly enough to make the scaling reduction less of a concern
- Of the solvers used the R11 MPP version was the fastest

Mesh Size and Adaptivity

While it is clear from these tests that the MPP/LS-DYNA offers the best run times and reasonable scalability across multiple cpu cores, it is worth considering if the run time performance could be further enhanced by model settings and options – in particular relating to blank mesh size and the use of adaptive re-meshing.

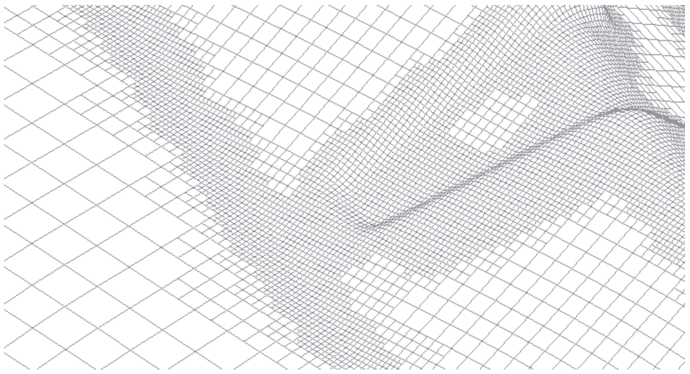

Figure 7 Adapted blank mesh

Adaptive re-meshing has been a feature in LS-DYNA since the 1990s. It was introduced as a method to speed up sheet metal forming simulations by allowing an initially large element size to be used for the blank which would then be locally refined as necessary based on the curvature of the approaching tool. On detecting imminent contact, based on a look-ahead distance, the solver would pause and divide any blank elements based on an angle value to ensure that they would form over the tool shape more accurately (Figure 7).

One element can be divided into four smaller elements at each adaptive cycle. The number of levels of adaptivity (two meaning one division, three meaning two divisions, etc.), the minimum element size below which no more adapting takes place, as well as the angle tolerance and frequency of checking are all defined using a control card.

This approach gives a reasonably efficient approach to running sheet metal forming simulation problems and can be very effective, especially with large relatively flat panels with small local areas of finer detail. However, adaptive re-meshing is not without its problems. If the elements are created slightly late then the new nodes may receive a higher contact force leading to bumps or spikes in the blank, or simply some unevenness in the predicted stresses. Also, with an aggressive mass-scaling timestep a lot of extra mass can quickly be added in the refined mesh areas, contributing to the uneven stresses. On the other hand, areas that remain flat (or at least below the adaptive angle threshold) will run throughout with the larger initial element size; this may inhibit the development of wrinkles, such as in the part of the blank held on the binder. In addition, the fact that the simulation must be re-initialized at every adaptive cycle leads to scaling inefficiencies as noted above.

For all these reasons adaptive re-meshing might be considered something of a compromise which is only sustained in order to achieve practical simulation run times. But what if the improvements in parallel processing in LS-DYNA, along with more affordable hardware and options to licence extra cpu cores, make this compromise no longer necessary?

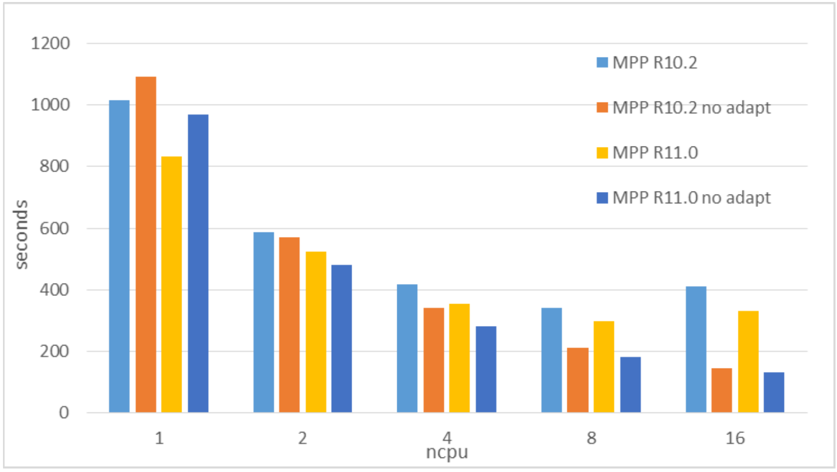

Figure 8 Part 2 rerun with no adaptivity

Part 2 was re-run with a blank meshed entirely with 5mm elements from the start, and with adaptive re-meshing turned off – the run times with MPP/LS-DYNA R10.2 and R11 are shown in Figure 8, comparing the results presented earlier with the refined non-adapting model. The run time on 1 cpu with no adaptivity is slightly slower for both versions but in all other cases the model runs quicker, and furthermore shows much improved scalability for 8 and especially 16 cores (where in the original model scalability was reversed).

The "no adapt" model achieves a speed up factor of ~5.3 with 8 cores and ~7.4 with 16 cores – much better than the original model which gave factors of ~2.8 and ~2.5 respectively. Some minor differences in results were observed but it can be argued (for reasons noted earlier) that the "no adapt" results might be the more accurate.

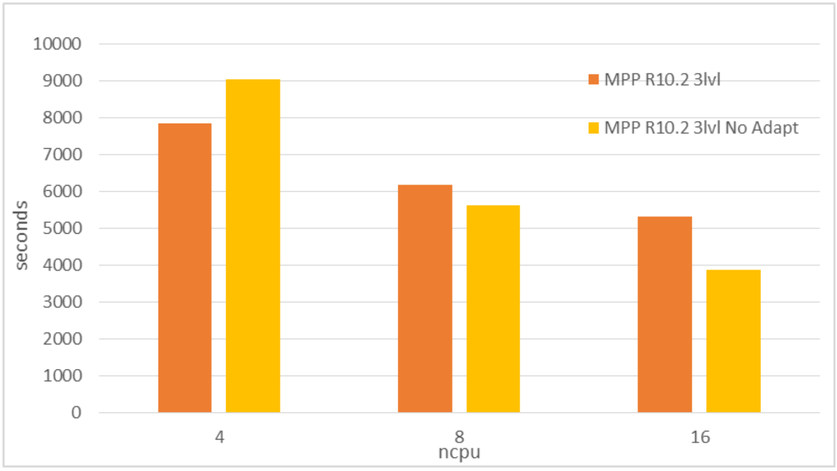

Figure 9 Part 3 rerun with no adaptivity

Part 3 was also re-run with no adaptivity and meshed with a 2.2mm element size – this was compared to a re-run of the original model with three (instead of four) adaptivity levels so also having a smallest final element size of 2.2mm.

The model was run using MPP/LS-DYNA R10.2 on 4, 8 & 16 cores. The run time comparisons are shown in Figure 9.

With 4 cores the "no adapt" model is slower than the adaptive model but on 8 and especially 16 cores the "no adapt" result is faster; on 16 cores it is ~27% faster, a saving of almost 24 minutes. As with Part 2, the difference in the results (e.g., thickness distribution) was minor. N.B., as with Part 2, the time step for mass scaling was left unchanged.

Based on these tests, running both a small, fast model and a large, slower model, it would appear that it is better to run with a refined blank mesh with no adaptive re-meshing to get the best run times when running the MPP solver on 8 or more cpu.

One other point to note is that LST are working on In Core Adaptivity [2], a method using dynamic memory allocation to handle the change in array sizes as the model adapts. The work to re-code the solver to use this is substantial, including the need to re-balance domains dynamically, but if the need to stop and re-initialize (on a single cpu) every adaptive cycle can be eliminated then an overall reduction in run time should be achievable.

Time Step for Mass Scaling

Another key solver feature in achieving an efficient run time while maintaining an acceptable level of accuracy is mass scaling – the technique of adjusting the element density to maintain the specified time step without breaking the Courant time step criterion. This allows a model to run with an acceptable time step even if some (deformable) elements in the model are too small for the criterion based on the original material density. However, an excessively high time step setting can distort the results due to the additional inertial forces in areas with a lot of adaptivity and hence many, small elements. For this reason, our usual approach has been to set the time step for mass scaling based on the smallest expected element size after the prescribed level of adaptive re-meshing so that only a few isolated elements have additional mass, to give an acceptable degree of accuracy within a reasonable run time.

An alternative mass scaling option is selective mass scaling (SMS). This works differently from conventional mass scaling (CMS) in that it limits the spurious dynamic effects by solving a set of linear equations that to some extent relates to implicit methods, even though it is done using an explicit approach. However, it is more challenging to set the time step correctly; too low and the model would run much slower (due to the extra computation to solve the linear equations), too fast and the solution could grind to a halt due to conditioning problems in the matrix solution. This case is now caught in recent releases of the solver so it is now easier to choose a good value for SMS.

Based on tests with the three parts discussed earlier (and indeed many others not reported here), we believe that generally speaking, a value of 3x the CMS minimum time step gives a "break even" run time while a SMS value of 6 – 8x the CMS value gives optimum results in terms of both run time and model accuracy. For example, in Part 1 the CMS timestep for a 1.1mm smallest element size would be 0.2E-06s with a run time of 138s; with the SMS set to 6x this the model runs in 96s, 30% faster. And with Part 2, the CMS step would be 1.0E-06s running in 426s, but with SMS 7x faster the run time was 367s, 15% faster. With the larger Part 3 with more levels of adaptivity, the CMS result with 0.2E-06s ran in 23083s whereas the SMS run with 8x this timestep solved in just 7306s, almost 70% faster, and with only very minor detectable differences in the results.

Hence, it would appear that SMS is a good option for sheet metal forming simulation to achieve improved run times with no appreciable effect on the results as long as the time step value chosen is between 6 – 8x the minimum based on the smallest element size after mass scaling.

Other Ideas for Optimizing Run Time

Although LS-DYNA is not generally seen as making heavy use of disk read/write operations (at least when running explicit models that don't need large scratch files), it was felt that a check on the effect of disk hardware might be of interest, with the availability of larger and larger solid state drives (SSD) offering an alternative to the traditional spinning disk hard disk drive (HDD). This may be particularly the case when running with adaptive re-meshing as the solver must write the current state to disk, create a new input file and start afresh each time the model adapts.

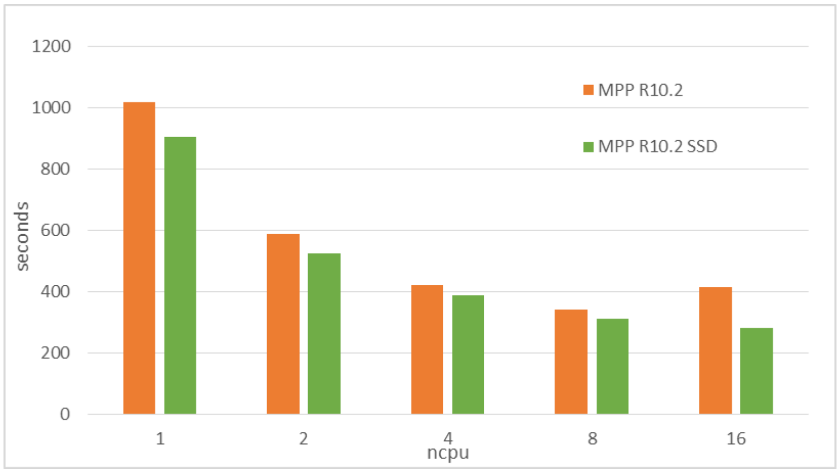

Figure 10 Part 2 rerun with SSD for file input/output

Part 2 was re-run with file input and output on SSD with interesting results, albeit for a model that runs relatively quickly. Figure 10 compares the original R10.2 results with the same model re-run on SSD. The run times on all cores are faster using SSD but what is interesting is that the negative scaling going from 8 to 16 cores with the original model is reversed, with the SSD model showing a small (11%) speed up instead. A similar set of results were produced for Part 1. Of course, the benefits of this may be reduced with less frequent (or no) adaptivity, although other I/O tasks will still be faster.

Another feature in LS-DYNA relating to adaptive re-meshing that can have an influence on run time is adaptive fusion. Here, elements that have been divided by adaptivity fission can be re-combined once they have moved over the high curvature region into a flatter area of the tool. There is a certain amount of care required to set the parameters for this correctly – including the number of fusion cycles cf. fission, the angle tolerance, birth time and initial fission level before fusion is applied. In practice we have found it quite difficult to derive default settings for reduced run time with fusion and in some cases the chosen parameters have not created a speed up at all. This is likely to be model dependent; where a blank is drawn from a flat binder over a tight die radius onto a flat wall then some speed up with fusion may be possible. Note though, that the LS-DYNA manual indicates some potential loss of accuracy when applying fusion, particularly for springback. All in all, and given the earlier indication that running with minimal or even no adaptive fission may be the better option, no further testing of fusion was undertaken.

Conclusions

The tested models would indicate that, where simulation run time is a key consideration, running sheet metal forming simulation on PCs with up to 16 cores the best option is to run the MPP solver and limit the use of adaptive re-meshing as far as possible. Selective mass scaling, based on a timestep of up to eight times the minimum time step for the majority of the model, is also beneficial. In addition, running the model on an SSD is likely to offer further run time improvements.

References

1. "Benchmark 2 – Springback of a Jaguar Land Rover Aluminium Panel Part A: Benchmark Description", Martin Allen et al, Journal of Physics: Conference Series 734 (2016) 022002
2. "In Core Adaptivity", Wainscott & Fan, 15th International LS-DYNA Users Conference, 2018