Performance Study of In Core Adaptivity in LS-DYNA®

Houfu Fan, Brian Wainscott, Li Zhang and Xinhai Zhu
LST, an ANSYS company

Abstract

Adaptivity is a key technique in metal forming applications, to save computation time while maintaining the accuracy of the result at locations of interest. Although very useful, the traditional implementation of adaptivity in LS-DYNA has certain inefficiencies in its execution. A new approach for adaptivity has been being developed for a couple of years in MPPDYNA. Recently load rebalance was also added into the code right after each adaptive step, which dramatically improves the computation efficiency. A performance study on the current status of this work is presented.

Introduction

In metal forming applications, one of the key techniques is adaptivity. The idea of adaptivity is to allow the user to input a coarsely meshed blank at the beginning, which automatically gets refined at locations of interest as simulation goes.

In order to perform an adaptive step, the traditional implementation of adaptivity in LS-DYNA needs to write the all the current solution into disk, create a new input file with a refined mesh, perform MPP domain decomposition and resume the computation with the previous information. The time spent in the I/O and re-initialization as well as MPP domain decomposition could be huge, since these are repeated at each adaptive step. In fact, for large model, this part takes most of the simulation time.

The new approach was first reported in 2018, named as in core adaptivity [1]. In core adaptivity has been being developed for a couple of years in MPPDYNA, which can execute in parallel with out exiting the solution loop. New elements, nodes and related arrays are dynamically allocated within the code. At the time when in core adaptivity was first introduced in 2018, there is still a bottle-neck in this approach. During each adaptive step, each processor shall generate new elements, nodes and related arrays at its own rate (density), according to the local requirement. This creates very poor load distributions overall all the participating cores, leading to very inefficient use of the computation power occupied. This part is now resolved by the lately developed load rebalance process right after each adaptive step. It is expected that right now the in core adaptivity would lead to dramatically time saving, especially when the model is large.

Test results of two models of different sizes, with different parameter settings are presented. Adaptivity time (percentages), the total runtime and total time saving percentages for the two models are presented.

In the rest of the paper, the traditional adaptivity approach will be referred as out of core adaptivity, in comparison to the new approach named in core adaptivity.

Test results

Two model problems, each with formability and springback settings are presented here. Formability and springback settings refer to two different set of simulation parameters, both of which are accepted as general metal forming application rules for the purpose of formability analysis and springback analysis, respectively. Stamping simulations with springback setting generally take more computational cost, because of high accuracy requirements in the upcoming springback analysis.

For the sake of confidentiality, details on neither of the two models will be provided here. Model 1 is a small stamping model with 6818 elements in the blank at the beginning of the simulations. With formability setting, after 76282 cycles and 246 adaptive steps, the blank ended up with about 320K elements. With springback setting, after 193,046 cycles and 232 adaptive steps, the blank also ended up with about 320K elements. Model 2 is a medium stamping model with 8186 elements in the blank at the beginning of the simulations. With formability setting, after 151,304 cycles and 435 adaptive steps, the blank ended up with about 1010K elements. With springback setting, after 338,756 cycles and 411 adaptive steps, the blank also ended up with about 1010K elements.

Each simulation was run on 12, 24, 36, 48, 96 and 192 cores. All the results are obtained with the single precision development version of MPP LS-DYNA r146785 on April 16, 2020, running on 768 core cluster with infiniband interconnect. All the timing data was taking from the table at the bottom of the d3hsp file from each run. The adaptivity time for an out of core simulation consists of three parts: “Keyword Processing”, “MPP Decomposition” and “Initialization”, which are all repeated each adaptive step. The adaptivity time for an in core simulation refers to the summation of six parts: “Keyword Processing”, “MPP Decomposition”, “Initialization”, “Adaptive Check”, “Adapt Shells” and “Rebalance”, in which the first three only appear once for each simulation. “Adaptive Check” refers to the time spent on determining which shell to adapt and “Adapt Shells” denotes the time spent on modifying the meshes, which are both repeated each adaptive step. “Rebalance” represents the time spent on shuffling elements and related data between different processors so as to balance the computation loads in all evolving cores, which occurs only at adaptive steps with significant load imbalance.

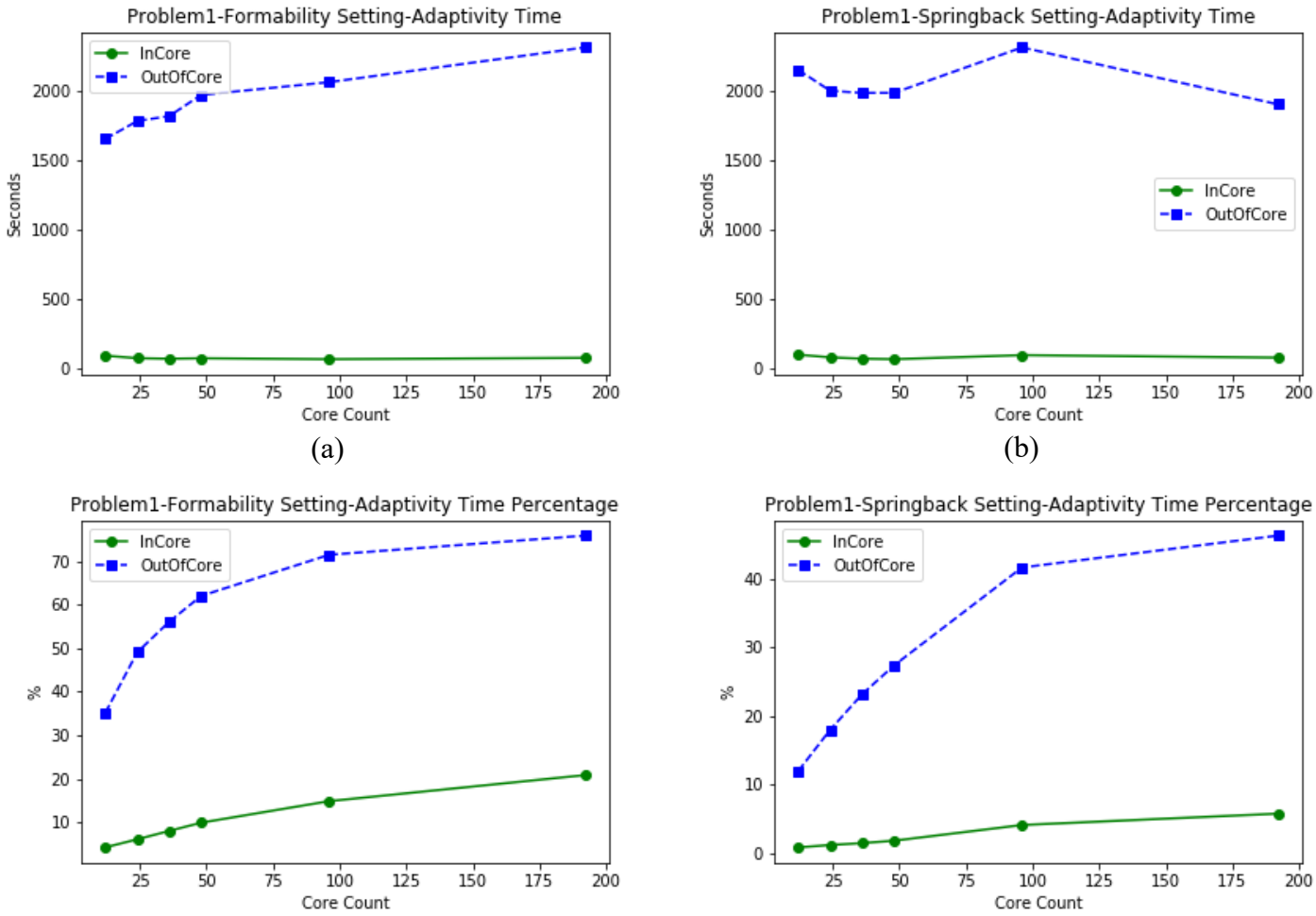


Figure 1, Problem 1-Adaptivity Time (Percentage) versus Core Count

First, the adaptivity time spent in out of core/in core simulations of problem 1 and 2 with different settings will be presented. Figure 1 shows the adaptivity time (percentage) with formability and springback settings on problem 1. With formability setting, the out of core spent between 1654 and 2312 seconds in adaptivity (Figure 1(a)), contributing a total percentage of simulation time ranging from 35% to 76% (Figure 1(c)). Correspondingly, the in core spent between 97 to 81 seconds in adaptivity (Figure 1(a)), contributing a total percentage of simulation time ranging from 4% to 21% (Figure 1(c)). With springback setting, the out of core spent between 2150 and 2310 seconds (Figure 1(b)), contributing to a total percentage of simulation time ranging from 11% to 46% (Figure 1(d)). The in core spent between 102 to 73 seconds, contributing a total percentage of simulation time ranging from 1% to 6%.

One can see that the time spent on doing adaptivity for out of core is huge, as compared to the corresponding time spend by in core. In fact, from the result in Figure 1, one can quickly calculate and get a factor (out of core / in core) of roughly 17 to 30 for all the simulations involved. One could imagine the benefit of time saving by slaughtering this part.

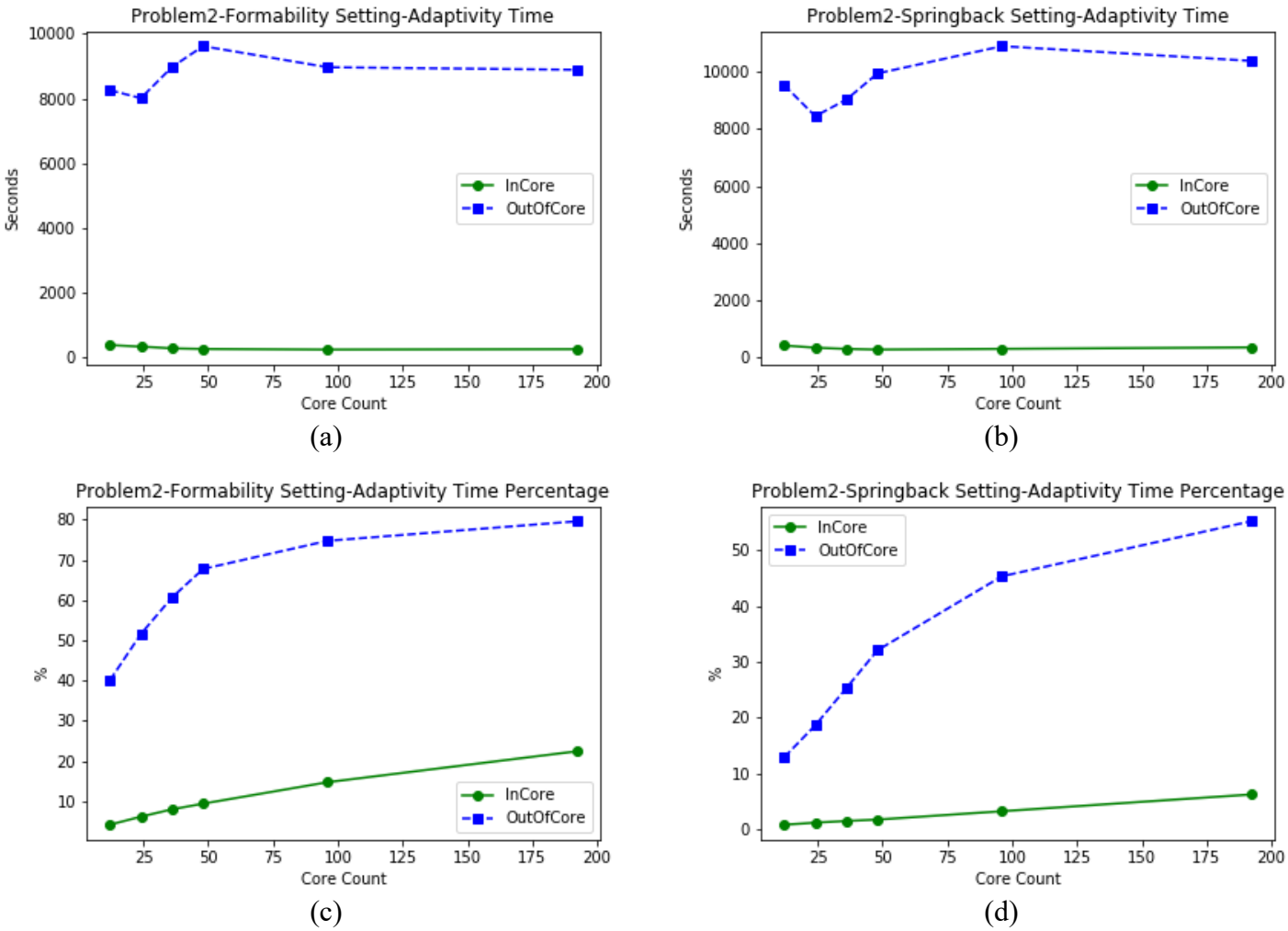
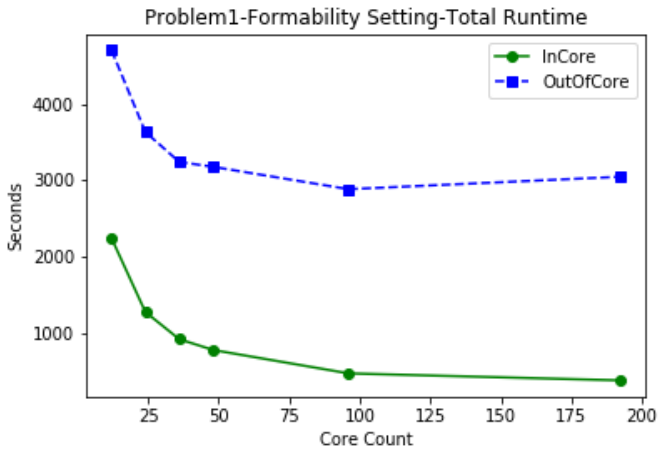


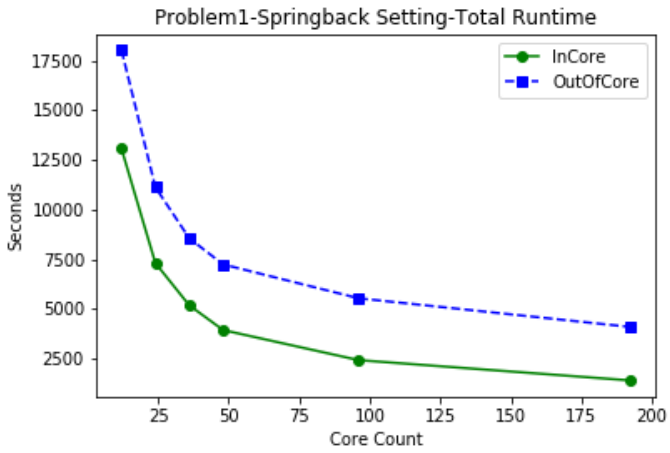
Figure 2, Problem 2-Adaptivity Time (Percentage) versus Core Count

Figure 2 shows the adaptivity time (percentage) with formability and springback settings on problem 2. The results are similar to that of problem 1. With formability setting, the out of core spent between 8263 and 9620 seconds in adaptivity (Figure 2(a)), with the percentage of time spent doing adaptivity ranged from 40% to 80% (Figure 2(c)). Correspondingly, the in core spent between 386 to 259 seconds in adaptivity (Figure 2(a)), which ranged from 4% to 22% out of the corresponding total run time (Figure 2(c)), indicating an increase of rebalance with the number of cores increasing. With springback setting, the out of core spent between 9528 and 10375 seconds (Figure 2(b)), contributing to a total percentage of simulation time ranging from 13% to 55% (Figure 2(d)). The in core spent between 424 to 355 seconds, contributing a total percentage of simulation time ranging from 1% to 6%.

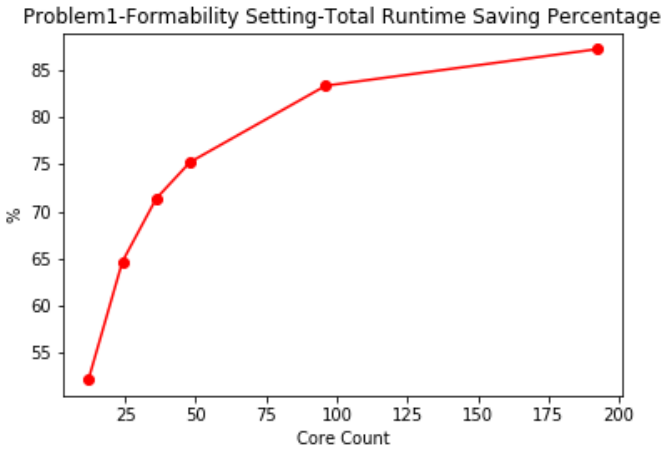
From the results shown in Figure 1 and 2, one can see that in core adaptivity time is quite consistent for problems with formability and springback settings. With formability setting, the percentages of time spent doing adaptivity ranged from 4% to 22% for problem 1, and ranged 4% to 21% for problem 2. With springback setting, the corresponding percentages ranged from 1% to 6% for both problem 1 and problem 2.



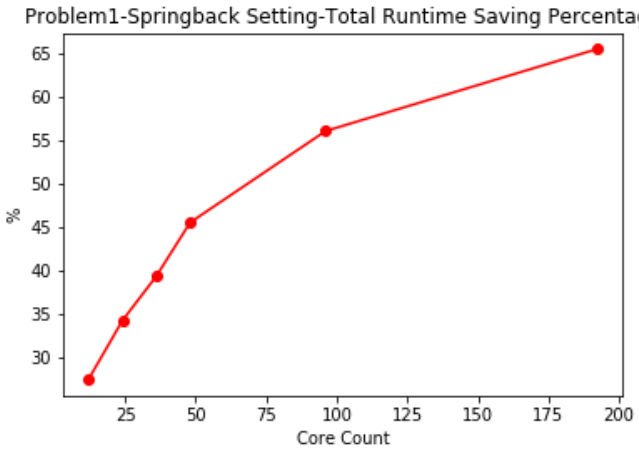
(a)



(b)



(c)

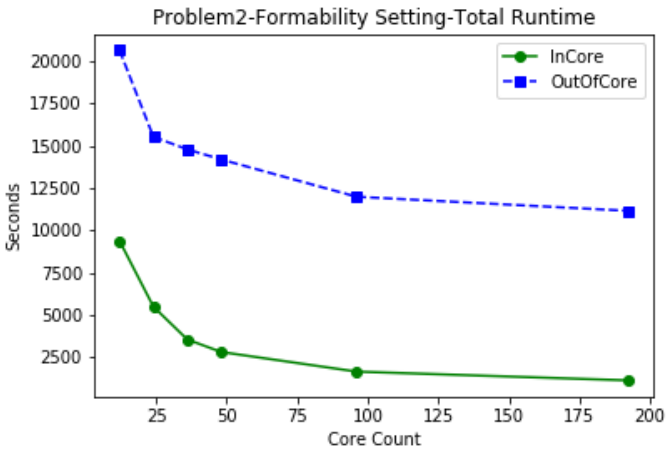


(d)

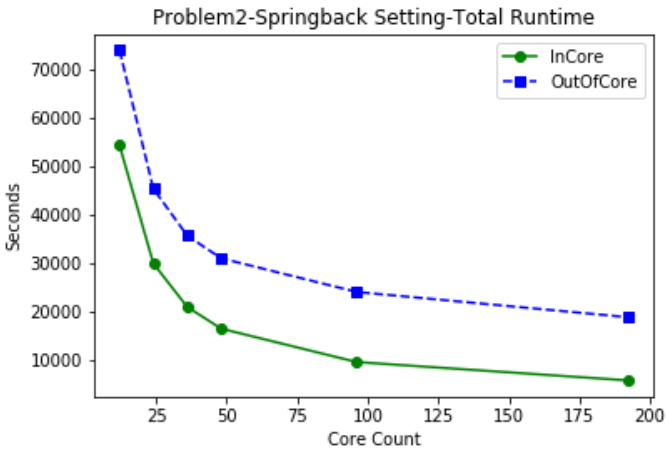
Figure 3, Problem 1-Total Runtime (Saving Percentage) versus Core Count

Next, total runtime spent in out of core/in core simulations of problem 1 and 2 with different settings, as well as the overall time saving percentage of in core total runtime with respect to the corresponding out of core will be presented.

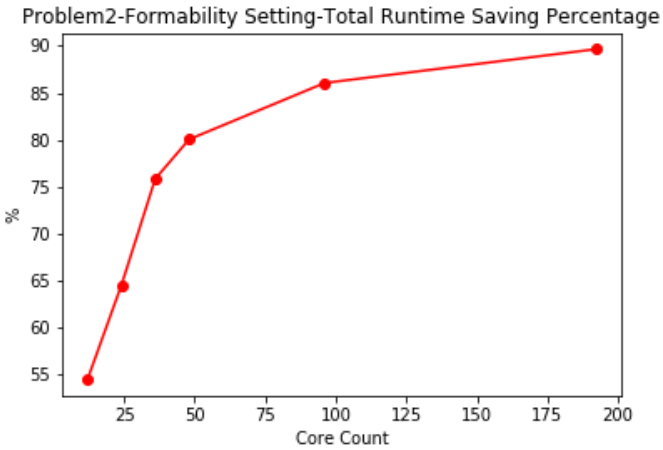
Figure 3 (a-b) shows the total runtime with formability and springback settings on problem 1. With formability setting, the out of core spent between 4702 and 2887 seconds in total runtime (Figure 3(a)). One thing to be noticed is that increasing from 36 cores to 48 cores does not help much in total runtime saving (3234 seconds to 3179 seconds), which clearly indicates that the out of core already starts to saturate (problem 1 is relatively small). In fact, the simulation with 96 cores only takes 2887 seconds while that with 192 cores yields 3049 seconds. Correspondingly, in core spent between 2247 to 390 seconds in total runtime, decreasing monotonically as the number of cores increases. Figure 3 (c) shows the time saving percentage increases from 45% to 90%. Even with 36 cores (out of core not yet saturates), the time saving percentage is 75%. With springback setting, the out of core spent deceases from 18013 to 4104 seconds (Figure 3(b)). The in core spent decreases from 13058 to 1415 seconds. Figure 3 (d) shows that the overall time saving percentage increases from 27% to 65%.



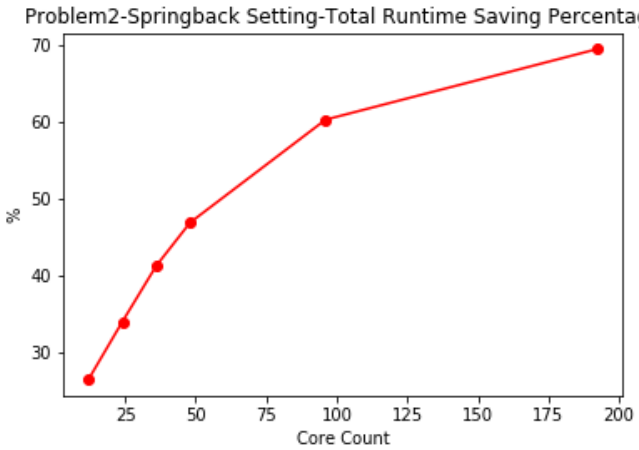
(a)



(b)



(c)



(d)

Figure 4, Problem 2-Total Runtime (Saving Percentage) versus Core Count

Figure 4 (a-b) shows the total runtime with formability and springback settings on problem 2. With formability setting, the out of core total runtime spent decreases from 20652 to 11172 seconds, monotonically (Figure 4(a)). Correspondingly, in core total runtime spent decreases from 9396 to 1157 seconds, monotonically. Figure 4(c) shows the time saving percentage increases from 55% to 90%. With springback setting, as shown in Figure 4(b), the out of core total runtime spent decreases from 74033 to 18796 seconds. The in core spent decreases from 54363 to 5719 seconds. Figure 4 (d) shows that the time saving percentage increases from 27% to 70%.

From the results in Figure 4, one can see that with the same model, time saving percentage is higher with formability setting as compared to springback setting. This can be attributed to two aspects. First, problem 1 with formability setting (246) has slightly higher adaptive steps than that of springback setting (232), which means the former has more time to be "slaughtered" by using in core adaptivity. In addition, the problem with formability setting runs much less number of cycles, requiring less real computational cost than that with springback setting. To sum up, in core with formability setting has more out of core adaptivity time to be "slaughtered" while at the same time, only needs to do less real computational job, as compared to in core with springback setting.

Summary

In this work, a performance study on in core adaptivity in LS-DYNA is reported. Two models of different sizes, with different parameter settings, run with number of cores ranging from 12 to 192, using both out of core and in core approaches. Results show that in core adaptivity can save total computation runtime dramatically. For a small model of approximately 320K elements in the blank at the end of the simulations, the total time savings with formability setting ranges from 45% to 90% and that with springback setting ranges from 27% to 65%. For a medium model of approximately 1010K elements in the blank at the end of the simulations, the total time saving with formability setting ranges from 55% to 90% and that with springback setting ranges from 27% to 70%.

References

[1] In Core Adaptivity, Brain Wainscott and Houfu Fan, 15-th International LS-DYNA users conference