Blucher Mechanical Engineering Proceedings
May 2014, vol. 1 , num. 1
www.proceedings.blucher.com.br/evento/10wccm

10th World Congress on Computational Mechanics
8-13 July 2012 • São Paulo • Brazil

PARALLEL COMPUTING EFFICIENCY FOR REAL WORLD PROBLEMS

Dietmar Carl

Bundeswehr Technical Center for Protective- and Special Technologies WTD 52, Schneizlreuth, Germany

Abstract. *The numerical simulation of a high speed short duration event like a detonation or a projectile penetration usually requires the use of sophisticated explicit computer codes. These codes apply small time steps for every computational cycle. Combined with the need to model large structures with millions or hundreds of millions of nodes, these computations can take from days to weeks to run. To shorten the computation time, modern commercial explicit codes support the use of parallelized computer systems. This approach frequently claims to reduce computation times: The use of eight processors instead of one is supposed to reduce computation time by a factor of six or seven for example. While that might be true for simple problems, it is not necessarily true for complex problems. This paper describes some real world applications and corresponding numerical simulations involving Euler and Lagrange computation techniques. Single and parallel computing times are compared.*

Keywords: *Numerical Simulation, parallel, Euler, Lagrange*

1. INTRODUCTION

The numerical simulation of transient events using hydrocodes dates back to the mid 50's [1]. Since then, the codes as well as the computers used to run them have evolved remarkably which is reflected by the appearance of the first commercial supercomputer CDC 6600 (1964) , the Star-100 with 100 MFlops up to today's (2012) machines like the Fujitsu K Computer with a peak speed of about 10 PFlops. Together with the improvement of hard- and software, the demand to model problems closer to reality has increased. The combination of parallel supercomputers and efficient parallelized solving algorithms now allows the handling of problems with hundreds of millions of cells.

However, such large problems are still a challenge for the hard- and software as well as for the engineer using them. Problems concerning computation time arise during set up of the problem and while solving and post processing the results.

2. HARD AND SOFTWARE USED DURING THE TEST PHASE

The computed problems described in this report were designed and run with the following tools:

Software:

- ANSYS® Autodyn version 14.0 (hereinafter referred to as Autodyn): This software is designed to solve highly dynamic non-linear problems. At WTD 52, it is used for solving problems related to weapon effects and protective systems.
- Operating system: Windows 7 professional, 64 bit. For more than 8 tasks SUSE Linux version 11, service pack 1 was used.

Hardware:

The computations shown in this report were run on computers (one or more depending on the use of Windows or Linux) with the following specifications: Processor Intel® Xeon® X5677, double processor quad core, 3.46 GHz, 144 GB RAM.

3. PROBLEMS

In this report, several problems are shown which were selected to test the codes' efficiency when run in parallel. These test problems can be divided into two groups:

- Uncoupled problems
- Coupled problems

The uncoupled problems are employed as a kind of reference where one would expect maximum efficiency of parallel computations.

The coupled problems are used to test the efficiency of the codes when employed to simulate more or less real world applications.

3.1. Uncoupled problems

The uncoupled problems were used as a reference for the computation time of the Lagrangian and Eulerian (multi material solver) solvers. They were kept as simple as possible and therefore cubes were used as mesh geometries. The dimensions of the cubes were 100 mm x 100 mm x 100 mm with 215 nodes in every direction resulting in approx. 10 million nodes. With these cubes computations were carried out using single as well as 2 - 16 processors. The speed up and efficiency were calculated as follows:

$$S = t_s/t_p \,. \tag{1}$$

$$E=t_s/(t_p \cdot n) \,. \tag{2}$$

With: S: Speed up
t_s: Computation time per cycle, single task
t_p: Computation time per cycle, parallel
E: Efficiency
n: Number of tasks in parallel

The following table shows the computation time per cycle, the speed up and the efficiency for these computations:

Table 1. Computation times, speed up and efficiency for uncoupled problem, 10 million nodes

Solver	No. of tasks	Time per cycle (s)	Speed up	Efficiency
Euler	1	67	1.00	1.00
Euler	2	36	1.86	0.93
Euler	4	21	3.19	0.80
Euler	7	17	3.94	0.56
Euler	8	15	4.47	0.56
Euler/Linux	14	11	6.09	0.44
Euler/Linux	16	8	8.38	0.52
Lagrange	1	27	1.00	1.00
Lagrange	2	13	2.08	1.04
Lagrange	4	6	4.50	1.13
Lagrange	7	4	6.75	0.96
Lagrange	8	6	4.50	0.56

As can be seen from table 1:

- As expected, the computation time is decreasing with increasing number of tasks unless one utilizes 8 tasks in the Lagrangian case.
- The efficiency drops significantly for the Euler computation when using more than four tasks.
- For the Lagrange computation, efficiency remains above 0.9 until applying 8 tasks when it decreases to 0.56.

A second series of computations similar to the mentioned above was carried out using only 1 million nodes. The following table shows the results for this problem:

Table 2. Computation times, speed up and efficiency for uncoupled problem, 1 million nodes

Solver	No. of tasks	Time per cycle (s)	Speed up	Efficiency
Euler	1	3.89	1.00	1.00
Euler	2	2.60	1.50	0.75
Euler	4	1.56	2.49	0.62
Euler	7	1.94	2.01	0.29
Euler	8	1.35	2.88	0.36
Lagrange	1	2.39	1.00	1.00
Lagrange	2	1.31	1.82	0.91
Lagrange	4	0.67	3.57	0.89
Lagrange	7	0.38	6.29	0.90
Lagrange	8	0.33	7.24	0.91

The following figure shows the efficiency for the above mentioned cases:

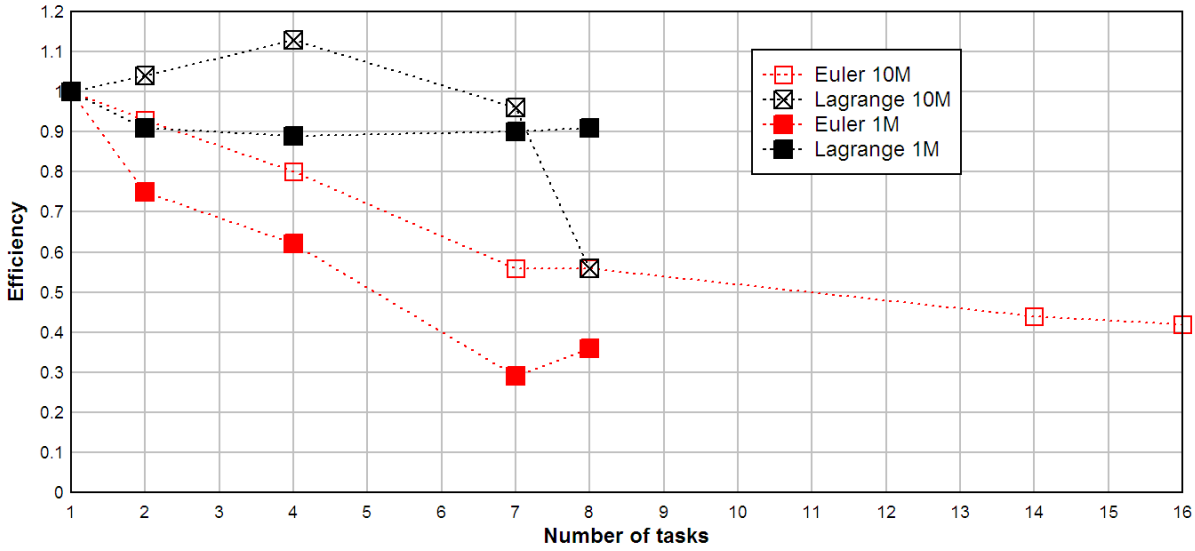


Figure 1. Efficiency for uncoupled computations.

As can be seen from figure 1:

- The efficiency for the Lagrange problems remains at 0.9 or higher except for the 10 million node case with 8 tasks where it drops to 0.56.
- The efficiency for the Euler problems is lower than for the Lagrange problems and decreases with increasing number of tasks.
- The number of Euler nodes influences the efficiency with a lower number resulting in lower efficiency.

3.2. Coupled problems

In the next step, more complicated coupled problems were generated and tested as single and parallel computations. Two types of coupled problem were used:

- Lagrange-Lagrange coupling
- Euler-Lagrange coupling

3.2.1. Lagrange-Lagrange Coupling, case A

The Lagrange-Lagrange coupling is typically applied when problems like the penetration of a bullet into a hard target have to be computed. Here, a simpler set-up is used: Two identical cubes made of steel having the same speed but with opposing directions collide. The following figure shows the problem:

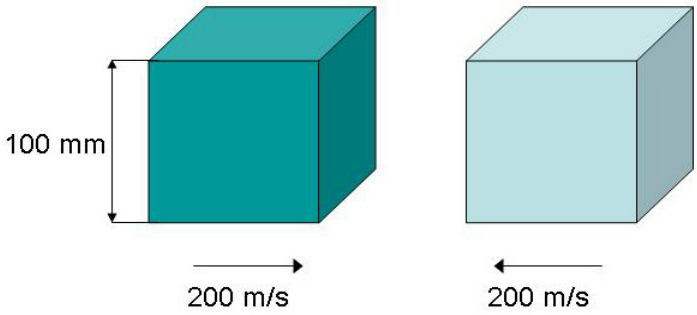


Figure 2. Sketch of Lagrange-Lagrange problem, case A (not to scale).

The dimensions of each cube were 100 mm x 100 mm x 100 mm, the distance between the cubes was 1 mm. Each grid had 5 million nodes with equal spacing. The cubes were filled with a linear elastic material. Lagrange-Lagrange interaction was turned on using the external gap option. The following figure shows the decomposition for two and four tasks:

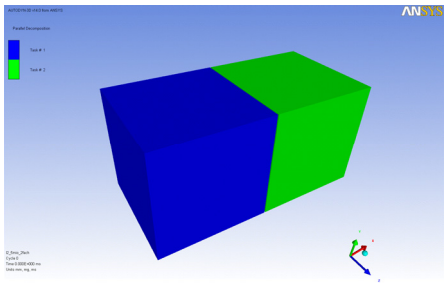
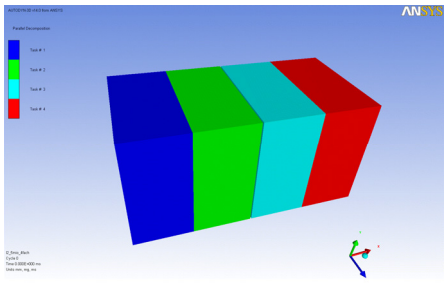
Tasks	Decomposition	Tasks	Decomposition
2		4	

Figure 3. Decomposition for Lagrange-Lagrange computations

The following table shows the computation times for these set-ups:

Table 3. Computation times, speed up and efficiency for Lagrange-Lagrange coupled problem, case A

No. of tasks	Time per cycle (s)	Speed up	Efficiency
1	23.4	1.00	1.00
2	13.0	1.80	0.90
4	7.3	3.21	0.80
6	5.3	4.42	0.74
7	7.0	3.34	0.48
8	5.5	4.26	0.53

As can be seen from table 3:

- The efficiency decreases with increasing number of tasks and remains above 0.7 for up to 6 tasks. For 7 and 8 tasks, the efficiency drops to about 0.5.
- The efficiency in the coupled set-up is smaller than that for the uncoupled Lagrange computation.

3.2.2. Lagrange-Lagrange Coupling, case B

In this model a penetrator hits a target, like for example a high speed projectile shot from a tank impacting the armour of another tank. The following figure shows the set-up:

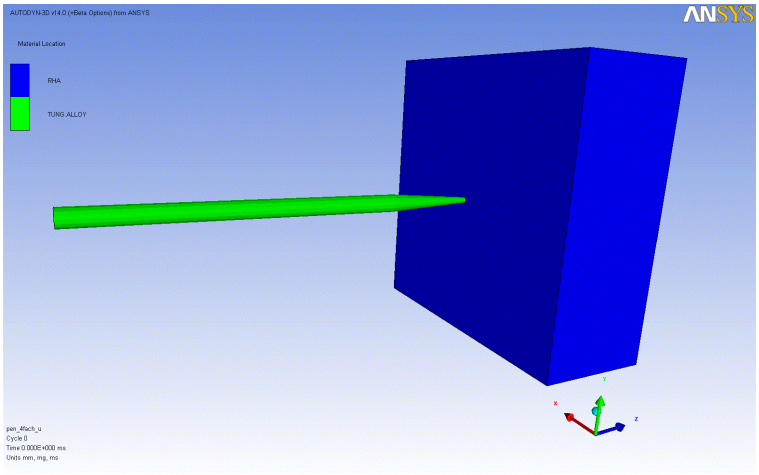


Figure 5. Lagrange-Lagrange problem, case B.

The main input data for the computation was as follows:

Penetrator:	Length:	650 mm
	Diameter:	30 mm, 10 mm at tip
	Speed:	1000 m/s
	Material:	Tungsten alloy
	Number of cells:	8320
Target:	Thickness:	200 mm
	Width x Height:	500 mm x 500 mm
	Material:	Rolled homogenous armour (RHA)
	Number of cells:	998560

In contrast to the other computations, this was done using unstructured meshes. Also, trajectory contact was used. The parallelization was done automatically by the Autodyn software.* The following table shows the computation times, the speed up and the efficiency:

Table 4. Computation times, speed up and efficiency for Lagrange-Lagrange coupled problem, case B

No. of tasks	Time per cycle (s)	Speed up	Efficiency
1	2.03	1.00	1.00
2	1.58	1.29	0.65
4	1.06	1.92	0.48
8	1.06	1.92	0.24

* In version 14.0 of Autodyn, trajectory contact in combination with parallel computations has beta status.

As can be seen from table 4, increasing the number of tasks to 8 does not speed up the computation when compared to 4 tasks. In this set-up, the efficiency seems to be especially low when compared to the other cases in this report.

3.2.2. Euler-Lagrange Coupling

For the Euler-Lagrange coupling, a set up was chosen where 1 million Euler nodes and 1 million Lagrange nodes were used. The model is shown in the following figure:

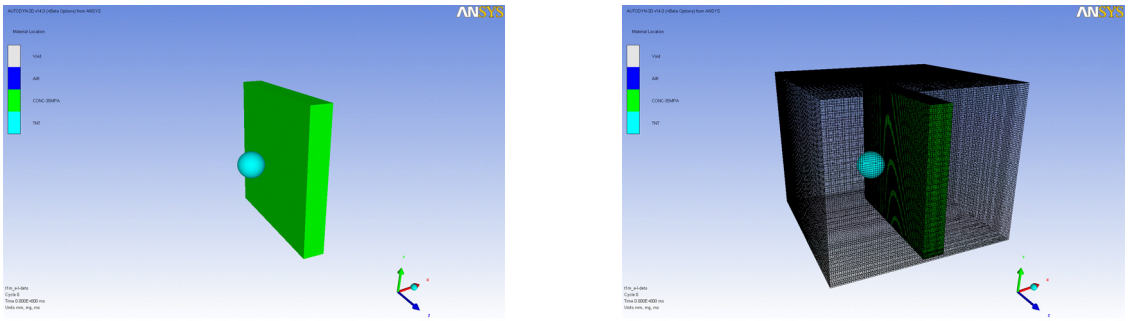


Figure 6. Euler-Lagrange problem, left without, right with grid (air not shown).

A TNT charge is modeled that detonates in front of a concrete wall. The following table shows the computation time per cycle, the speed up and the efficiency for these computations:

Table 5. Computation times, speed up and efficiency for Euler-Lagrange coupled problem

No. of tasks	Time per cycle (s)	Speed up	Efficiency
1	11.6	1.00	1.00
2	11.6	1.00	0.50
4	7.7	1.51	0.38
6	7.0	1.66	0.28
8	6.5	1.79	0.22

As can be seen from table 5:

- For one and two tasks, the computation times are identical, parallelization does not increase the speed up.
- Using 4, 6 or 8 tasks reduces the computation time. The efficiency drops from 0.38 for 4 tasks to 0.22 for 8 tasks.
- The efficiency is smaller than the efficiencies for Euler or Lagrange or coupled Lagrange-Lagrange computations: Using four processors, it reaches 0.38 while for the simple cube problems it is between 0.8 and 1.13 (see table 1).

4. SUMMARY

This paper describes single and parallel computations using the codes ANSYS® Autodyn 14.0. Most of the computations were done on Windows systems, whilst some were run on a Linux operating system. The aim of the computations was to compare run times and efficiency associated with parallelization for simple problems (cubes, no interactions) with the speed up for simplified real world problems including coupling and detonation.

The comparison has shown that the speed up and the efficiency for coupled applications are significantly lower than for the simple cube problems. This is shown in the following table for the cases with 4 tasks:

Table 6: Efficiencies for 4 tasks

Lagrange	1.13
Euler	0.8
Lagrange-Lagrange, A	0.8
Lagrange-Lagrange, B	0.48
Euler-Lagrange	0.38

While the results of this report may not to be applicable to all kinds of problems, they support the experience the author has gained over years working with codes like ANSYS Autodyn. However, it should be noted that different hardware configurations might result in different results.

3. REFERENCES

[1] M.L. Wilkins. Computer Simulation of dynamic Phenomena. Berlin: Springer, 1999.