# Integration and Optimization of a 64-core HPC for FEM- and/or CFD Welding Simulations

Per R. M. Lindström*, Andreas de Blanche**

* DNV Materials Laboratory, Det Norske Veritas AS, Høvik, Norway /
*Department of Engineering Science, University West, Trollhättan, Sweden*

** Department of Engineering Science, University West, Trollhättan, Sweden

## 1 Introduction

This document describes the selection and integration of a computational platform (hardware and operative system) intended for the sake of CWM-analyses.

Computational Welding Mechanics, CWM, can not only be used for RSDP, Residual Stress and Distortion Prediction [1]. It can also be used for a broad range of applications such as remaining life time analyses of reactors, pressure vessels, boilers, furnaces, tube- and pipe systems. As well as the Manufacturing Engineering process of integrated metallic structures that is the engineering discipline that by the application of physics principles deals with the R&D and optimisation of manufacturing processes and systems.

In a scientific thermo-mechanical FEM-simulations to determine the temperature and stress fields present during welding have been performed for about forty years. The FE-technology and the computer capacity have developed to a state that Computational Welding Mechanics, CWM, is an established and mature process [2] [3] and CWM can be used as reliable tool in structural design, for example in optimization and parametric studies, [4].

A successful commercial CWM implementation requires that the actual industrial organisation has gained the knowledge and understanding of the following related topics:

- Computational platforms that comprises the selection of hardware, operative system and FEM-code as well as suitable pre- and post-processing tools
- Welding Engineering with an emphasis on the weld process parameters and its thermodynamics
- Test weld quality control such as calibration, validation, DAQ and documentation etc.
- Transient thermo-mechanical coupled FE-analyses and constitutive modelling

## 2 HPC Computer Hardware

During the last fifteen years the industrial HPC-systems have been dominated by low-end SMP (symmetrical multi-processor) computers or Linux based cluster solutions. This was changed by the introduction of multi core processors to the mass market. The multicore computers sparked the renaissance of the MPP-architecture (Massively Parallel Processor) which is a single computer, controlled by one operating system instance, containing a large number of processor cores and a shared memory often implemented in a NUMA (Non-Uniform Memory Architecture) [5].

Our computational platform has four 16-core AMD Opteron 6284SE processors with a total of 64-cores, these are the same processors as Cray use in their XE6 supercomputer [6]. The cores run at 2.7GHz and are equipped with AMD's Turbo CORE [7] functionality that adjusts the speed of the cores depending on the number of cores in use. Each 16-core processor can be connected to two NUMA memory regions with each region connected to 125GB of memory; however our computer is equipped with a total of 264GB DDR3 memory.

The system is running CentOs version 6.4, and we are connecting to and controlling it through ssh and ThinLinc [8]. ThinLinc is a remote desktop software built upon the VNC protocol and it basically processes all graphics on the server and sends the screen-image to the client. The desktop communication is performed through an encrypted tunnel and the computer can be reached from anywhere in the world and is impossible to listen to. The remote desktop solution works perfectly well and the graphical experience (with a good internet connection) is better than the built in graphics card connected to a monitor. Fig. 1 shows a screenshot of our CWM-model in LS-PrePost [9] from a Windows 7 laptop.

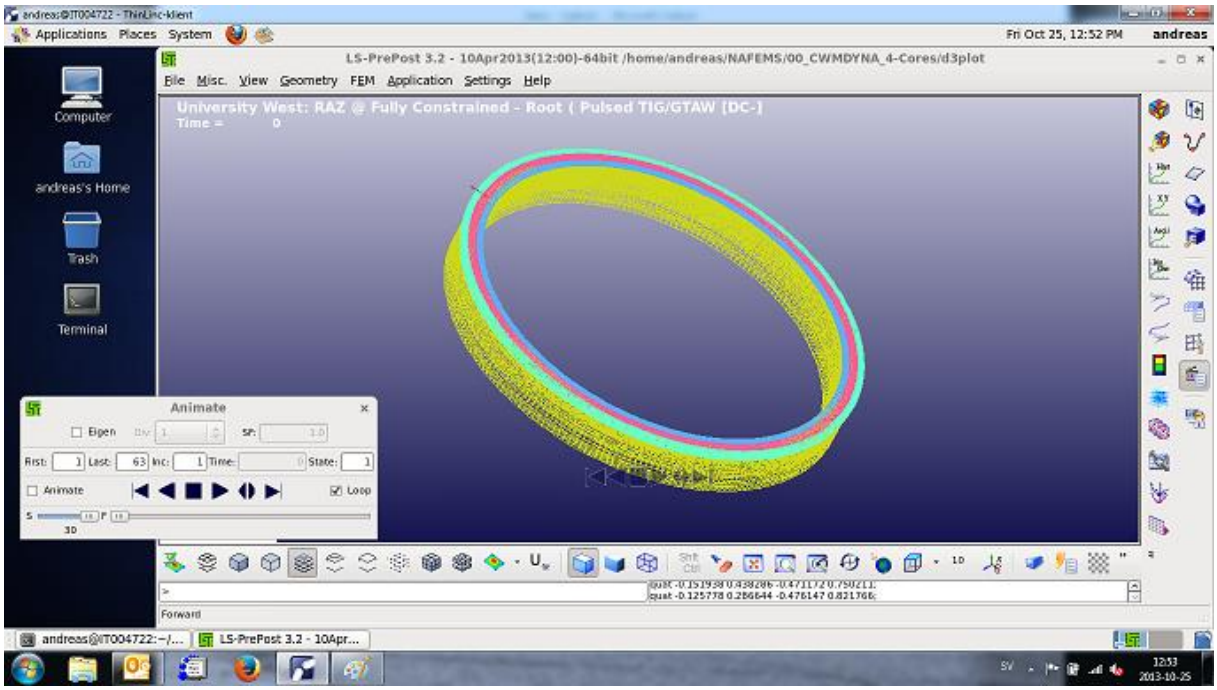


**Fig. 1. Screenshot of ThinLinc and LS-PrePost taken on Windows 7, with our benchmark model.**

All in all, the system that we use for CWM had a total cost of SEK 100 000 :- and has almost instant communication between all processes. This should be compared with a cluster consisting of something like 8 nodes with 8 cores each and an Infiniband solution. The cluster has a several orders of magnitude slower process communication, require almost eight times the space, and administration time as well as energy consumption is higher.

## 3 Experiments

In our first series of experiments we executed the simulation serially as well as in parallel distributed into: 4 ; 8 ; 12 ; 16 and 20 processes. To get results as close to a production system as possible all experiments were carried out on a fully loaded computer, i.e. all 64-cores were utilized during all experiments. All scheduling and allocation decisions were made by the underlying Linux kernel and the partitioning was done on-the-fly by the LS-Dyna CWM solver [10] [11] [12].

Turning to the results, Fig. 2 contains the "wall clock"-time (top) and the "perfect speedup"-time (bottom) lines. The "wall clock" time is how long we have to wait to get the answer from the simulation and "perfect speedup" is a best case estimate of how fast the simulation could execute according to Amdahl's law [13] with a serial component of 0, which is a better-than-possible estimate.

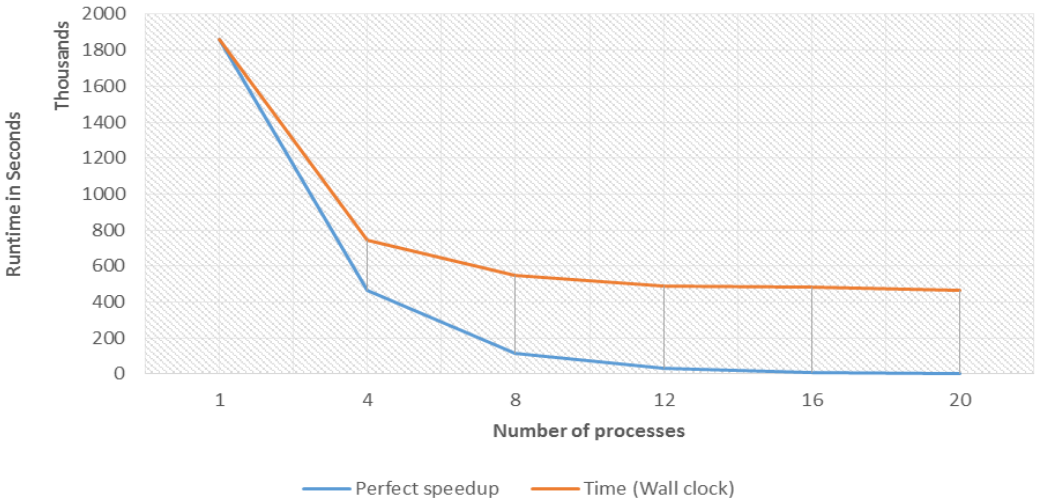


**Fig. 2. Simulation runtimes with 1 to 20 processes on a fully loaded computer.**

The difference between the lines, in Fig. 2, indicates that the incentive for increasing the number of processes decreases the higher we go. Although it is a well-known fact [14] that the speed increase diminishes for every added process until it levels out completely. Knowing how fast the speedup deteriorates can be very useful when determining how to allocate the computational resources. Take Fig. 3 as an example. The dotted line shows the simulation runtimes normalized after the fastest runtime and the solid line is the throughput, i.e. the number of jobs that can be completed in a given amount of time. If we view the throughput/runtime problem as an optimization problem we want to maximize the throughput and minimize the runtime.

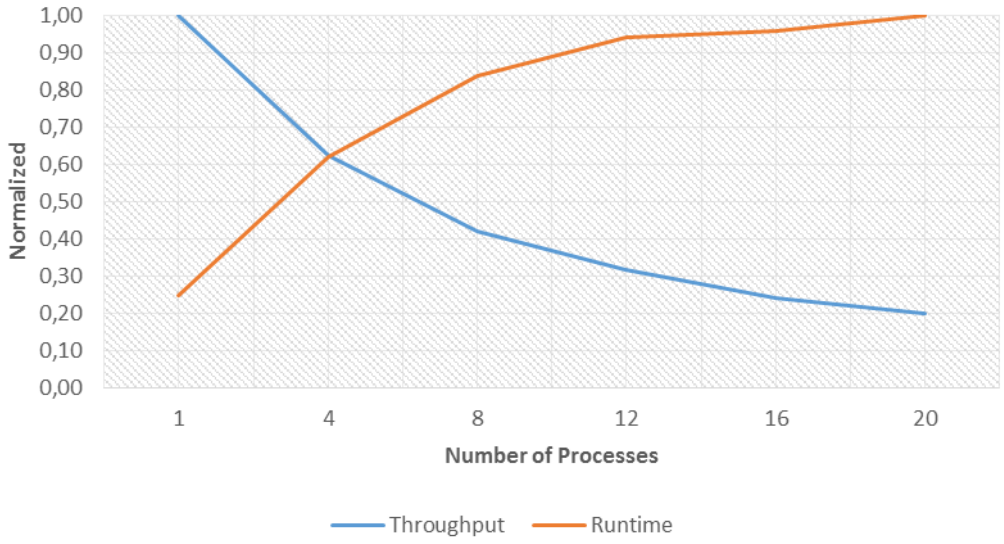


**Fig. 3. The normalized throughput and runtimes for simulation executed in parallel over 1 to 20 cores.**

Given that we have a steady flow of simulation jobs to be executed the most efficient choice (read: most simulations per euro spent) is to maximize the throughput. However, that also means that we will have to wait the longest for the simulation to finish. On the other hand if we have a tight deadline we should use 20 cores (or maybe more) to be able to finish the project on time. Consequently, executing only 4 processor jobs would give the best tradeoff between throughput and runtime.

**Example:** We want to make a parametric study and evaluate five parameters with ten discrete levels each. For this experiment we will have the entire computer at our disposal. The results in Fig. 4 clearly show that the throughput is extremely important when the systems are under heavy load. It should be mentioned that the calculations are based on whole jobs and since 64 is not even dividable by 12 or 20 those setups only utilize 60 cores.

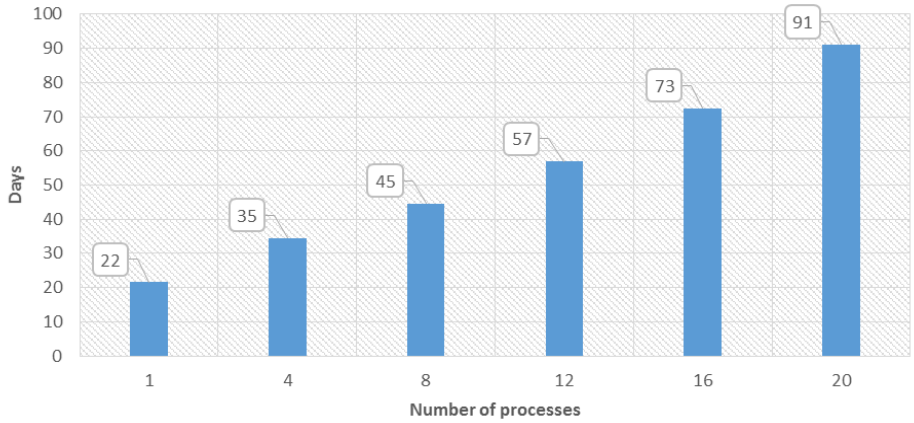


**Fig. 4. Total runtime for a parametric study with 50 simulations when utilizing different degrees of parallelization.**

## 4 The NUMA memory

Currently one of the largest challenges in multicore job-scheduling is to overcome the memory bottleneck of modern multicore computers. To characterize the non-uniform memory architecture of our computational platform we used the STREAM benchmark to evaluate the memory throughput. Fig. 5 shows the aggregated memory bandwidth for our system. We measured the bandwidth 100 times with different, randomly selected, process placements. We noticed that the median value came very close to the best value for all cases but 12 processes.

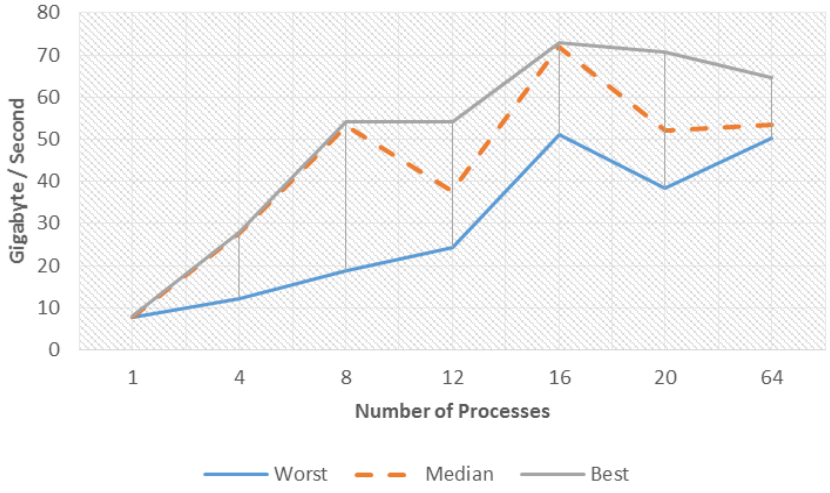


**Fig. 5. Aggregated memory bandwidth as measured by STREAM for [1 … 64] processes.**

Figure Fig. 5 also show that placing 20 processes on 20 cores will basically render the same (median) aggregated memory throughput as when placing processes on all 64 cores. Turning to fig Fig. 6 we can clearly see that the per process memory throughput declines quite rapidly and that the spread between the worst and best cases converges towards 1GB/s when utilizing all 64 cores. However, by utilizing some simple memory based job-scheduling techniques as those proposed in [15] [16] or just manually reordering the job-schedulers allocation scheme it is fully possible to guarantee that he per process available memory bandwidth always will be in the area between the median and best cases.

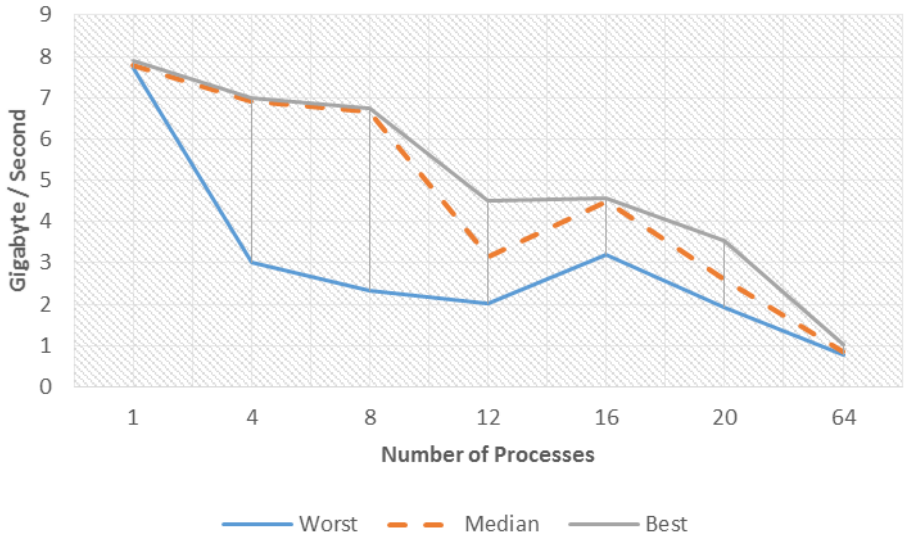


**Fig. 6. Per process memory bandwidth as measured by STREAM for 1 -64 processes**

## 5 Discussion and Conclusions

When trying to get most from a 64-core MPP for computational welding simulations we first ventured into the administrative areas of the system with ease of access and administration, we then turned to the systems performance characteristics. To mimic a production environment our experiments were performed in accordance with the recommendations of DNV Materials Laboratory [17] on a more or less fully utilized system. We used the LS-Dyna MPP-solver [10] and our CWM material models [11] [12] and a downsized fully 3D tube model [18]. All in all we found that:

- It is not obvious which combination of solver and mpi-library that will work out-of-the-box on all systems. We had to recompile the MPICH library in order to get the standard solver to work.
- The more cores that are used the faster the simulation will finish, however when there are many jobs in queue decreasing the parallelization increases the throughput radically.
- Performing quite “simple” changes in the job-schedulers allocation algorithm can increase the *worst case* per-process memory performance with between 1% and 185% (average 65%).

## 6 Acknowledgements

Arblos AB is gratefully acknowledged for the financial support of this study.

## 7 References


[1] Janonosch, J. J., 2003, Round Robin Phase II – first 3D modeling results, IIW Doc. No. X-1541-03

[2] Goldak, J., Akhlaghi, M., 2005, Computational Welding Mechanics, ISBN: 9780387232874,Springer-Verlag New York Inc., USA

[3] Lindgren, L.-E., 2007, Computational Welding Mechanics, ISBN: 1845692217, Woodhead Publishing, UK

[4] Pakhamaa, A., Wärmefjord, K., Karlsson, L., Soderberg, R., and Goldak, J., 2012, Combining variation with welding simulation for prediction of deformation and variation of a final assembly, International Journal of Computing and Information Science in Engineering, Vol. 12, to appear.

[5] Yang R; Antony, J.; Rendell, A.; Robson, D.; Strazdins, P., "Profiling Directed NUMA Optimization on Linux Systems: A Case Study of the Gaussian Computational Chemistry Code," Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International , vol., no., pp.1046,1057, 16-20 May 2011.

[6] Vaughan, C.; Rajan, M.; Barrett, R.; Doerfler, D.; Pedretti, K., "Investigating the Impact of the Cielo Cray XE6 Architecture on Scientific Application Codes," Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on , vol., no., pp.1831,1837, 16-20 May 2011.

[7] Branover, A.; Foley, D.; Steinman, M., "AMD Fusion APU: Llano," Micro, IEEE, vol.32, no.2, pp.28, 37, March-April 2012.

[8] http://www.cendio.com/products/thinlinc/

[9] LS-PrePost 3.2, 2012, Livermore Software Technology Corporation, USA

[10] LS Dyna 971, Release 7.0.0, Double Precision MPP Solver, Livermore Software Technology Corporation, USA

[11] Material model: MAT_THERMAL_CWM (MAT_T07), LS Dyna 971, Release 7.0.0, Livermore Software Technology Corporation, USA

[12] Material model: MAT_CWM (MAT_270), LS Dyna 971, Release 7.0.0, Livermore Software Technology Corporation, USA

[13] Hill, M.D.; Marty, M.R., "Amdahl's Law in the Multicore Era," Computer, vol.41, no.7, pp.33, 38, July 2008

[14] Boklund, A.; Namaki, N.; Mankefors-Christiernin, S.; Gustafsson, J. & Lingbrand, M., "Dual Core Efficiency for Engineering Simulation Applications", Parallel and Distributed Processing Techniques and Applications, CSREA Press, pp. 962-968, July 2008.

[15] de Blanche, A.; and Mankefors-Christiernin, S.;, "Method for Experimental Measurement of an Applications Memory Bus Usage", International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA, July 12-15, 2010

[16] Zhuravlev, S.; Blagodurov, S.; Fedorova, A.; "Addressing shared resource contention in multicore processors via scheduling", SIGPLAN. 45, March 2010.

[17] Lindström, P. R. M., 2013, "DNV Platform of Computational Welding Mechanics", IIW Document Number: X-1732-13, IIW 66th Annual Assembly, September 11th – 17th 2013, Essen, Germany

[18] Lindström, P. R. M., Josefson, B.L., Schill, M., and Borrvall, T., 2012, "Constitutive Modeling and Finite Element Simulation of Multi Pass Girth Welds", NAFEMS NORDIC Conference: Engineering Simulation, Gothenburg, Sweden