11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
July 20–25, 2014, Barcelona, Spain

USING GPU ACCELERATORS FOR CRYSTAL PLASTICITY SIMULATIONS

Ylva Mellbin[1]*, Håkan Hallberg[1] and Matti Ristinmaa[1]

Division of Solid Mechanics, Lund University, Lund, Sweden. E-mail: Ylva.Mellbin@solid.lth.se

Key words: *Crystal plasticity, Graphics processing unit, CUDA, GPGPU.*

In order to model large deformations in metals accurately, it is necessary for the model to be able to describe the changes occurring in the microstructure. One possible approach is by using a crystal plasticity model. Using the common Taylor assumption, the stresses in the material are calculated by averaging the results from several hundreds of crystalline grains subjected to the same deformation. In each of the grains the slip has to be calculated, which can be done by solving a set of stiff differential equations.

The main drawback with crystal plasticity models is that they become computationally very demanding, due to the ill conditioned problem of finding the slip and due to the large number of grains required. Since there is no coupling between grains, there exists a large potential for parallelization. Some implementations have taken advantage of this, but those rely on the availability of large computer clusters to reduce the computation time.

Recently it has been made possible to use graphical processing units (GPUs) for accelerating non-graphical applications. Originally produced to suit the gaming industry, GPUs are cheap and easily available, while optimized to handle large numbers of floating point operations in parallel. By utilizing GPUs it is possible to get considerable parallel throughput even in an ordinary desktop computer, making this approach attractive when compared to CPU-based cluster solutions. The present implementation of crystal plasticity takes advantage of this.

Different strategies for solving the equations associated with the slip have been investigated. The first one is an implicit Euler method commonly used for crystal plasticity simulations. Another one is an explicit Runge-Kutta formulation, which while less stable is also less memory demanding than the implicit method, and thus more suitable for GPU implementation due to the limited amount of memory available on the graphics card. The final approach, aiming to keep the stability while reducing the memory requirements even further, is a semi-implicit operator split methodt.

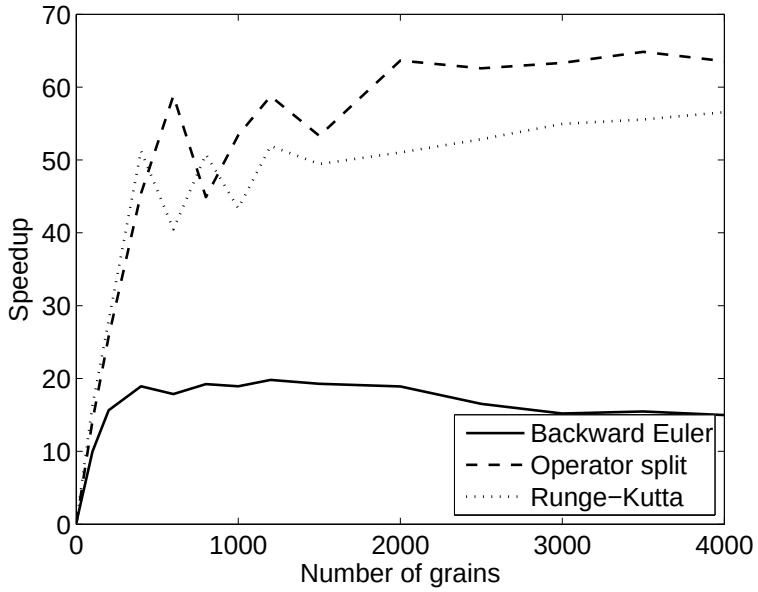

Figure 1: Speedup achieved with the GPU implementation compared to the CPU implementation for each of the three different numerical schemes, for different numbers of grains.

The speedup achieved when running the different methods on a GPU, compared to a CPU implementation of the same method, is shown in Figure 1, using different numbers of grains in the simulations. As is obvious from the graph, a large number of grains is required to make optimal use of the GPU, which is no problem considering that polycrystal plasticity simulations usually uses 400-1000 grains per integration point. Due to the larger memory requirements, the implicit method achieves less speedup than the two methods better suited for GPU implementation, but still a significant improvement. Which method is actually most favorable to use depends on many different factors, e.g. the loading rate and time step used, as well as on whether the formulation of the main program is such that the calculation of a tangent stiffness is required.

Even though the limited amount of memory available on the GPU remains a major concern, it is shown that significant speedup can be achieved through taking advantage of the computational capacity of GPUs. The approach makes it feasible to run crystal plasticity simulations even on a standard desktop computer equipped with a GPU, waiving the need for large clusters.