How numerical methods benefit from recent many–core processors

Gundolf Haase
Karl–Franzens–Universität Graz
Institut für Mathematik und wiss. Rechnen

Abstract

Recent developments in graphics hardware and associated software development tools enable the adaption of numerical algorithms for many–core processors and clusters of them. We use General Programmable Graphical Processing Units (GPGPU) as experimental platform for our investigations. Although the adaption to the GPU requires a redesign of algorithm components in order to fit into the highly parallel framework of the GPU, the resulting algorithms outperform the fastest single CPU implementation by more than one order of magnitude.

The presentation will introduce the benefits and restrictions of recent GPGPUs in comparison to available CPUs followed by an introduction into the host–device concept of GPGPU programming based on the CUDA platform. This concepts allows an incremental transfer of most time consuming code parts from the CPU to the GPU. This transfer is explained in more detail for a preconditioned CG with an (algebraic) multigrid preconditioner for solving a potential problem on unstructured 3D finite element meshes based on a medical technology project. We achieved an acceleration by 10 on the GPU for this unstructured data problem wrt. the fastest CPU available. A similar problem on tensor product meshes achieves a factor of 35.

For problems with more computation per memory transfer as a multi–component flow in complex domains a speedup between 30 and 100 can be realized, in case of a curve optimization for millions of parameters even more than that. One group of our SFB exploited the performance of a GPU for a radial MRI registration problem resulting in a perfomance improvement between 10 and 50.

We consider the GPGPU as a fast co–processor for a CPU or CPU–core and therefore we use our conventional parallelization strategy via MPI for clustering the GPUs. Except for the last, all algorithms run in parallel on up to 32 GPUs arranged in 4 nodes with 8 GPUs each. Depending on problem size, parallel granularity and algorithm we achieve between good and superlinear speedup on clusters of GPUs.