On the performance of various parallel GMRES implementations on CPU and GPU clusters

E.I. Ioannidis, N. Cheimarios, A.N. Spyropoulos, and A.G. Boudouvis

School of Chemical Engineering, National Technical University of Athens, 15780 Athens, Greece

ch08001@chemeng.ntua.gr, nixeimar@chemeng.ntua.gr, aspyr@chemeng.ntua.gr, boudouvi@chemeng.ntua.gr

Key words: Partitioning, Galerkin/finite element method, sparse matrices, Newton iteration, MPI, CUDA

As the need for computational power and efficiency rises, parallel systems become increasingly popular among various scientific fields. Distributed, shared and hybrid memory multi-core systems were until recently the main computing architectures used for parallel computations while sequential algorithms were transformed to run in those parallel architectures exploiting their high computing capabilities. Nevertheless, the constant need for solving efficiently demanding problems has led to the development of a new kind of parallel computing system: General-purpose Graphics Processing Units (GPGPU). The rapid evolution of general purposes GPU-based architectures takes high performance computing to the next level.

Applying finite element methods in boundary value problems involves the solution of large and sparse systems of linear (or linearized) algebraic equations of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, an expensive computational task that requires most of the total processing time. GMRES, an established iterative solver for large and sparse linear equation sets, is often employed for this task. In this work, different implementations of a parallel version of GMRES is presented, each of them on different computing architectures: From distributed and shared memory core-based to GPU-based architectures. The computational experiments emanate from the dicretization of a benchmark boundary value problem with the Galerkin/finite element method.

The main issue addressed is the method that someone should choose in order to efficiently solve large sparse linear systems of equations depending on the hardware available. Results show that memory access puts a serious limitation to the number of cores that can run concurrently on a shared memory core-based architecture. Distributed memory systems including fast network connections are therefore preferred as there is no noticeable effect of inter-processor communication on parallel speedup. Furthermore, the use of a single GPU device can accelerate the computations several times while using multiple GPUs even greater speedups are achieved.