Stack-oriented memory allocation using space filling curves for parallel PDE-solvers

Michael Bader <bader@informatik.tu-muenchen.de>
Miriam Mehl <mehl@informatik.tu-muenchen.de>
Christoph Zenger <zenger@in.tum.de>

Institut für Informatik, Technische Universität München
Boltzmannstrasse 3, D-85748 Garching, Germany

From the point of view of numerical efficiency, application programs for numerical simulation have to work with adaptive grids and multigrid methods to be competitive. However, these methods induce rather complicated -- and irregular -- data structures with numerous mutual dependencies of data points. This is an obstacle for efficient parallelization by message passing strategies or shared memory architectures, causes increased memory requirements and, last but not least, it deteriorates the cache efficiency on single processors due to the non-locality of data access. If we look at the extremely high computing times of fluid dynamical applications even on the biggest and fastest computers, it becomes obvious that this problem is not negligible. In practice, regular grids without adaptivity are still sometimes preferred because the loss of hardware and, in particular, storage efficiency caused by the use of adaptive codes may overcompensate the gain in the numerical efficiency.

In this paper, we describe a memory access strategy for adaptive finite element or finite difference algorithms where the access to memory is organized by a fixed and moderate number of stacks. In a stack, the access to memory is purely local as the only 'jumps' allowed in this concept are jumps to direct neighbours (in the sense of the order of the data structure). The main difference to conventional programs, where data remain on the same location, is the fact, that after reading data from memory (in our strategy from top of one of the stacks) data are stored back to a different location (in our case to another stack). This relocation of data has to be done even in the case that the values of the data remain unchanged. The writing of data back to memory can in principle be done in parallel to computation and should not cause an overhead in computing time if the computer architecture allows this parallelization.

The basic idea of our approach -- allowing the construction of our stacks in the first place -- is the ordering of the data along space filling curves. It was already observed that space filling curves of arbitrary type improve the locality of data access and allow an efficient balanced parallelization of PDE-solvers. However, the respective codes still comprise jumps in memory access. We overcome this problem by using special space filling curves and a tricky memory organisation leading to a small and fixed number of stacks. Peano curves and Sierpinsky curves turned out to be well suited for our approach whereas Hilbert curves seem not to be useful at least in the three-dimensional case.
First implementations of the concept already using multigrid acceleration techniques show that the approach works and that we in addition achieve a substantial reduction of the memory requirements. This is due to the structuredness of the underlying grids, the fixed ordering of grid cells and our cell-oriented algorithm for operator evaluation that enables us to do without pointers to neighbours as known from conventional tree implementations and without any data needed for identification of data as needed for hashing for example. On the other hand, the whole potential in saving computation time could still not be observed in practical computations. This requires further optimizations and adaptations of the code to the underlying computer architecture which is subject to our current work.