Preparing Algebraic Multigrid Solvers for Future Supercomputers

Ulrike Meier Yang

Lawrence Livermore National Laboratory
yang11@llnl.gov

ABSTRACT

Algebraic Multigrid (AMG) solvers are an essential component of many large-scale scientific simulation codes. Their continued numerical scalability and efficient implementation is critical for preparing these codes for future computer architectures. Multi-core processors are now standard on commodity clusters and high-end supercomputers alike, and core counts are increasing rapidly. With exascale machines expected to have hundreds of millions or billions of tasks and hundreds of tasks per node, programming models will necessarily be hierarchical, with local shared-memory nodes in a larger distributed-memory message-passing environment.

AMG has demonstrated good weak scalability in distributed-memory environments, but our investigation of its use on multicore architectures has shown that non-uniform memory access (NUMA) latency between sockets, deep cache hierarchies, multiple memory controllers, and reduced on-node bandwidth can negatively affect its performance. To achieve high performance on exascale machines, we will need to ensure numerical scalability and an efficient implementation as core counts increase, memory capacity per core decreases, and on-node cache architectures become more complex. Some components of AMG that lead to very good convergence do not parallelize well or depend on the number of processors.

We examine the effect of high level parallelism involving large numbers of cores on one of AMGs most important components, smoothers. We also develop a performance model of the AMG solve cycle to better understand AMGs performance bottlenecks, and use it to evaluate new AMG variants. Since our investigations show that the increasing communication complexity on coarser grids combined with the effects of increasing numbers of cores lead to severe performance bottlenecks for AMG on various multicore architectures, we investigate two different approaches to reduce communication in AMG: the the development of AMG variants, which require less messages, and the use of a hybrid MPI/OpenMP programming model.