AmgX: Scalability and Performance on Massively Parallel Platform

Maxim Naumov, Marat Arsaev, Patrice Castonguay, Jonathan Cohen, Julien Demouth, Joe Eaton, Simon Layton, Nikolay Markovskiy, Nikolai Sakharnykh, Robert Strzodka and Zhenhai Zhu

In this presentation we discuss the AmgX iterative solve framework, which includes implementation of two forms of Algebraic Multigrid (AMG), Krylov subspace methods, as well as parallel variants of typical smoothers and preconditioners.

The goal of AmgX is to fully utilize the hardware capabilities of large GPU-accelerated clusters, especially for solving elliptic PDEs. While parallel versions of many suitable numerical methods exist, it is challenging to find algorithms which fully exploit both fine-grained parallelism ("scale-in") within a single node, as well as the coarse-grained parallelism ("scale-out") between nodes. In particular, there are only a few choices when it comes to fault tolerant numerical schemes that retain the robust and fast convergence properties of serial methods when scaling on massively parallel computer platforms.

To address these goals we design a highly parallel version of standard methods such as Jacobi, Gauss-Seidel, diagonal incomplete-LU (DILU) and incomplete-LU with 0 fill-in (ILU0). We use graph coloring to expose fine-grained parallelism in both the factorization and solve phases of these algorithms. The graph coloring itself is performed in parallel on the GPU.

Since many real-world applications require the AMG setup phase to be called more than once, we efficiently parallelize both the setup and solve phases of the method.

The main feature of AmgX is a parallel Un-smoothed Aggregation AMG method. This method supports both scalar as well as coupled block systems, and scales well across multiple GPU enhanced nodes. The key to the fully parallel setup phase is a one-phase handshaking graph matching algorithm. A distinguishing feature is an optimized implementation of the Galerkin coarse matrix calculation taking advantage of the unsmoothed aggregation approach.

We also implement a Ruge-Steuben-style AMG solver that uses the parallel modified independent set (PMIS) coarsening algorithm with optional long-range interpolation. Also, we design the corresponding high performance routines for computing Galerkin projections and general sparse matrix-matrix product which efficiently utilize the memory and arithmetic throughput capabilities of GPUs.

In order to address fault tolerance we implement flexible variants of the standard Krylov subspace methods, such as CG, BiCGStab and GMRES. These schemes allow the preconditioner to vary across different steps and therefore can naturally handle fault tolerance if run as an outer iterative method.

Finally, we use MPI asynchronous communications to achieve coarse-grain parallelism. The data transfers are limited to the halo elements of each partition, and are completely hidden behind local computation on the fine levels of the matrix hierarchy.

To conclude, we put all of these parallel numerical schemes together into a complete software package, where the user is allowed to mix and match different inner and outer methods. The AmgX API allows the library to efficiently interact with sequential and parallel MPI codes. We present numerical experiments that showcase the performance of the library inside the real-world applications on large scale GPU-accelerated clusters.