Abdulrahman Manea

A Massively Parallel Semicoarsening Multigrid for 3D Reservoir Simulation on Multi-core and Multi-GPU Architectures

Green Earth Sciences Building
Room 065
Stanford University
Stanford
CA 94305-2220
amanea@stanford.edu
Tchelepi Hamdi

In this work, we have designed and implemented a massively parallel version of the Semicoarsening Black Box Multigrid Solver [1], which is capable of handling highly heterogeneous and anisotropic 3D reservoirs, on a parallel architecture with multiple GPUs. For comparison purposes, the same algorithm was also implemented on a shared-memory multi-core parallel architecture using OpenMP. The parallel implementation exploits the parallelism in every module of the original Multigrid algorithm, including both setup stage and solution stage, without modifying the original algorithm basic steps. The benefits of this approach are twofold: maintaining the inherent strong linear convergence of the serial Multigrid algorithm, and making advantage of the shared-memory architecture to minimize the need for communication.

The design of the algorithm uses a combination of plane relaxation and semicoarsening to efficiently handle anisotropies in 3D, [2]. Sense the z-direction in most reservoir models is a direction of strong-coupling compared to the x- and y- directions, semicoarsening is employed in the z-direction, and plane relaxation is used for relaxation on x-y planes. Besides solving 2D-systems for plane-relaxation, during the setup stage, a set of 2D systems must be also solved on each multigrid level to get an approximate representation of the exact prolongation operator described in [1]. For solving both types of 2D systems, we used a parallel version of the 2D standard-coarsening operator-induced multigrid [3]. To be able to handle problems involving high anisotropies in the x- and y- directions, we used alternating line-relaxation with zebra ordering to parallelize across multiple line solves. For the coarsest-solve, we use multicolor Gauss-Seidel algorithm, with four colors to handle the nine-point stencils of the coarsest structured grid.

There are several differences in the 2D-solver requirements between the setup-stage systems and the solution-stage systems. The setup-stage systems are independent and directly parallelizable. On the other hand, for parallelizing

the solution stage systems, i.e. plane-relaxation, red-black coloring must be used. Since the convergence of the overall semicoarsening multigrid algorithm is very sensitive to the quality of the prolongation weights computed by the 2D-solves during the setup stagewhich are in turn are used to build the coarse-grid operators in the standard Galerkin approach the setup stage 2D-solves are carried out to tight residual tolerances using several multigrid cycles. On the other hand, for plane relaxation, the accuracy requirements are much looser, where a single multigrid cycle is found to be good enough for smoothing most of the high-frequency error modes.

To have coalesced global memory access pattern on the GPU, the Diagonal Sparse-Matrix Format [4] was used to store the system matrices at all the levels. In both the 3D Semicoarsening Multigrid and the 2D Multigrid plane relaxation, V-cycling was used to avoid spending more time at coarser levels and thus affecting the parallel efficiency. To minimize the expensive communication through PCIe between the host and a GPU and amongst GPUs, every 2D solve is explicitly handled by a single GPU, where we avoid splitting a single 2D solve between multiple GPUs.

Due to the inherent granularity difference between the GPU threads and the Open-MP threads, each GPU thread is typically assigned one computational point, whereas every relatively coarse OpenMP thread is typically assigned several computational points. In addition, this inherent granularity difference also affects the design of the tridiagonal solves during the line-relaxation. Since OpenMP threads are coarse (as they are mapped to one physical core on the host), it is suffices to assign a line or a small group of lines to a single thread, where the tridiagonal systems can be efficiently handled using the serial Thomas Algorithm. However, on the GPU, this approach would be very inefficient, as it involves relatively large serial computational tasks, and would not generate enough parallel work to utilize most of the available hardware resources. Thus, the approach taken was to assign every computational point to a thread, where threads operates in two stages, namely, a setup stage and a solution stage. In the setup stage, the threads work on preparing the tridiagonal systems of one red/black zebra line sweep, in parallel. Then, in the solution stage, the threads solve those tridiagonal systems in one batch in parallel using NVIDIA's CUSPARSE LIBRARY [5], which in turn uses a combination of both the Cyclic Reduction and Parallel Cyclic Reduction algorithms.

The two versions were tested using various highly heterogeneous problems derived from SPE10 Second Dataset and varying in size from tens of millions of cells to hundreds of millions of cells. For problems with sizes large enough to saturate the GPU resources, the Multi-GPU implementation is found to be faster than the OpenMP implementation running on 12 Intel Xeon X5650 2.66GHz cores. In addition, the inherent serial nature of multiplicative multigrid, along with the approach taken to minimize the communication through PCIe, is found to limit the scalability beyond 3-4 GPUs.

References:

1. S. Schaffer. A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients.SIAM J. Sci. Comput, 20(1):228242, 1998.

2. Dendy, Jo E., et al. "Multigrid methods for three-dimensional petroleum reservoir simulation." Tenth SPE Symposium on Reservoir Simulation. 1989.

3. Alcouffe, R. E., et al. "The multi-grid method for the diffusion equation with strongly discontinuous coefficients." SIAM Journal on Scientific and Statistical Computing 2.4 (1981): 430-454.

4. N. Bell, M. Garland, "Efficient Sparse Matrix-Vector Multiplication on CUDA", NVIDIA Technical Report, NVR-2008-004, (2008).

5. NVIDIA Corporation, "NVIDIA CUDA CUSPARSE Library", [Online]. Available: http://docs.nvidia.com/cuda/index.html