Sparse Matrix Multiplication Method on Multiple Graphics Processing Units

Sidi Fu* [(1)(2)], Shaojing Li [(1) (2)], Ruinan Chang [(1) (2)], and Vitaliy Lomakin[(1) (2)]
(1) Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA
(2) Center for Magnetic Recording Research, University of California at San Diego, La Jolla, CA 92093 USA

Numerical solvers composed of fast algorithms have made it possible to efficiently simulate large scale electromagnetic (EM) problems. Among them, sparse matrix multiplication method (SpMVM) is an important component, as they allow reducing memory and computational time. There are several storage formats for sparse matrices. Here, we adopt the common compressed sparse row format (CSR).

While there is a hardware limitation for a further frequency improvement of Central Processing Units, massively parallel systems like Graphics Processing Units (GPU) has emerged and are being widely utilized. For example, the latest version of general-purpose GPU GTX 680 has 1536 streaming processors and has a performance of 3 TFLOPs and nearly 200 GB/s memory bandwidth. The idea of single instruction, multiple data (SIMD) has made it possible to accelerate the speed of some algorithms more than 100 times. And the price of $500 allows it to be used in a simple desktop.

In this work, we present a parallel SpMVM implementation on GPUs using the CUDA programing environment. For further acceleration purpose, we also developed a multi-GPU code with good parallelization efficiency by evenly assigning the computation and memory transfer work. The ability to achieve high acceleration rates requires a careful handling of the GPU memory, including special sorting procedures as well as the utilization of the GPU memory hierarchy.

We have tested our implementation with a 2 million sized problem on a workstation with two GTX 690, both dual GPU devices. The problem was solved within 7.95 ms on a single GPU, including 6.29 ms for computation and 1.64 ms for memory transfer. This corresponded to a 53x GPU-CPU acceleration comparing to our CPU implementation of SpMVM on INTEL Xeon E5-1650 at 3.2G Hz. For the multi-GPU implementation the same problem took 2.97 ms.