Comparative Performance Evaluation of Multi-GPU MLFMM Implementation for 2-D VIE Problems

Carl Pearson, Mert Hidayetoğlu, Wei Ren, Weng Cho Chew, and Wen-Mei Hwu

Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
hidayet2@illinois.edu

***Abstract*— We compare multi-GPU performance of the multi-level fast multipole method (MLFMM) on two different systems: A shared-memory IBM S822LC workstation with four NVIDIA P100 GPUs, and 16 XK nodes (each is employed with a single NVIDIA K20X GPU) of the Blue Waters supercomputer. MLFMM is implemented for solving scattering problems involving two-dimensional inhomogeneous bodies. Results show that the multi-GPU implementation provides 794 and 969 times speedups on the IBM and Blue Waters systems over their corresponding sequential CPU executions, respectively, where the sequential execution on the IBM system is 1.17 times faster than on the Blue Waters System.**

I. Introduction

The multilevel fast multipole method (MLFMM) computes pairwise electromagnetic interactions between each pixel pair in the discretized geometry by hierarchically clustering the radiating and incoming fields into a spatial quad-tree. In the near-field phase, spatially close interactions are computed within the lowest level of the MLFMM tree. The aggregation and disaggregation phases propagate interactions up and down through the higher-levels of the tree structure, resectively, and the translation phase propagates the long-range interactions within each level. In this way, the N^2 interactions among N pixels are evaluated with $\mathcal{O}(N)$ operations and memory requirement [1]. Even with the low computational complexity, parallel MLFMM is required to take advantage of high-performance computing resources.

In order to achieve an efficient implementation on GPUs, these four MLFMM phases are formulated as matrix multiplications. Common operators are pre-computed on CPU, moved to the GPU, and reused as needed to avoid host-device data transfer. The matrix operations optimized with shared memory tiling, register tiling, and thread coarsening. The MLFMM tree structure is partitioned among message passing interface (MPI) processes, where each of them employs a single GPU for performing partial multiplications. MPI communications are overlapped with GPU kernels to achieve high multi-GPU parallel efficiency. During the MLFMM multiplications, GPUs have to communicate through their owning MPI processes by moving the data from GPUs to central processing units (CPUs), CPUs to CPUs (through MPI routines), and then from CPUs to GPUs. To hide this communication cost, we overlap the MPI communications with GPU kernels. This strategy completely hides the communication cost and provides 96%, MPI-parallelization efficiency on up to 16 GPUs.

TABLE I
Execution Times on XE and S822LC Nodes

	Application	MLFMM	Portion
Blue Waters XE (32 Threads)	120 s	86 s	72%
IBM S822LC (160 Threads)	59 s	49 s	83%

II. MLFMM Performance Results

A. Computational Environments

We compare the implementation performance on two systems: the Blue Waters supercomputer [3], and an IBM S822LC workstation [4]. There are two types of Blue Waters nodes, where each is a two-socket system: the XE node has two AMD Opteron 6276 CPUs, each with eight floating-point units, hardware support for 16 executing threads, and 32 GB of RAM. The XK node replaces one of these CPUs with an NVIDIA K20X GPU and 6 GB of RAM. These XE and XK nodes are representative of the compute capabilities of current-generation clusters and supercomputers. The IBM S822LC represents a next-generation accelerator-heavy supercomputing node. It has two IBM Power8 CPUs, each with ten floating-point units, support for 80 executing threads, and 256 GB of RAM. It also has four NVIDIA P100 GPUs with 16 GB of RAM each.

B. MLFMM Contribution to Application Time

MLFMM is implemented for solving two-dimensional scattering problems involving highly-inhomogeneous bodies, through solution of a volume-integral equation. A BiCGStab solver invokes MLFMM many times until the iterative solution converges. As shown in Table I, MLFMM forms the core computational kernel of the application, and its performance dominates that of the full numerical solver in CPU-parallelized execution on XE and S822LC (72% and 83% respectively) nodes, justifying further targeted acceleration of MLFMM.

C. Overlapping MPI Communications with GPU Computations

To provide good multi-GPU parallel efficiency, we overlap the MPI communications with GPU computations via multi-stream GPU executions. Fig. 1 shows a timeline of a particular MLFMM execution with 16 GPUs in 16 XK nodes. The MLFMM tree structure is divided among many GPUs, creating the short kernel times and the quite communication

between the GPUs. However, the required communication time is substantially smaller than the long-running near-field kernel that it is overlapped with. This method fully hides the communication cost even for faster GPUs or slower inter-process communication.

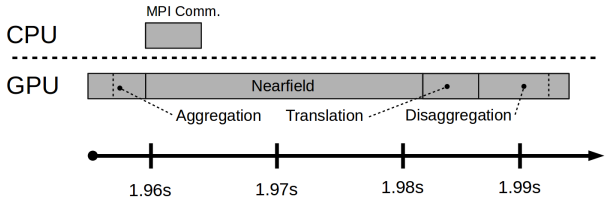

Fig. 1. Representation of MPI communication overlap. After the aggregation stage, the data required for the translation stage is communicated among GPUs during the near-field stage.

D. Performance Results

All evaluations are done on a problem with 16 million pixels (unknowns). Fig. 2 shows MLFMM performance scaling on various configurations on Blue Waters nodes and on the IBM workstation. On the XK nodes (Fig. 2(a)), each node runs a single MPI process. 1T and 32T are single-thread and 32-thread OpenMP executions on a single XE node. On the S822LC workstation (Fig. 2(b)), the 4 MPI ranks run on a single machine to utilize the 4 GPUs. 160T is a 160-thread OpenMP executions on S822LC. 1 GPU is a GPU-accelerated execution on a single XK node or using one GPU on S822LC. 4 GPU and 16 GPU are multi-GPU executions with a corresponding number of MPI ranks.

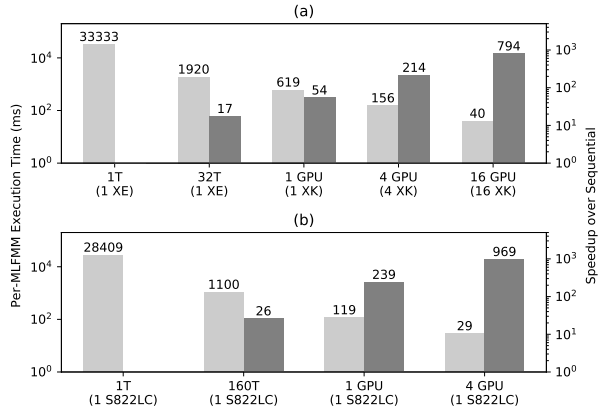

Fig. 2. MLFMM execution times and speedup over single-thread executions on (a) Blue Waters XE and XK nodes and (b) IBM S822LC. Light bars represent execution time (left axis). Dark bars show speedup normalized to the sequential execution on the respective system.

Both XE and S822LC achieve more CPU speedup than they have floating-point units (17 times speedup with 32 threads on 16 units for XE, 26 times speedup with 160 threads on 20 units for S822LC). When floating-point units are oversubscribed, they are more fully utilized.

The CUDA implementations leverage hybrid shared-memory and register tiling, and thread coarsening [2]. In both systems, using a GPU for MLFMM provides substantial speedup (additional 3.2 and 9.2 times speedups on XE/XK and S822LC nodes, respectively) over fully utilizing the CPUs. Furthermore, nearly linear scaling when using multiple GPUs is also achieved thanks to overlapping strategy of required MPI communication with GPU computation. This corresponds to a reduction in execution time from approximately 33 seconds to 40 milliseconds on XK nodes, and 28 seconds to 29 milliseconds on S822LC.

Despite the 5-year gap between deployment of the Blue Waters and IBM S822LC systems, the baseline sequential execution is only 1.2 times faster on S822LC than on an XE node. This reflects the slow pace of single-threaded CPU performance improvement. On the other hand, the P100 GPU in S822LC provides 4.4 times speedup over the K20X in XK. On a per-node basis the four GPUs in S822LC provide 17.9 times speedup over the single GPU in XK.

The average kernel-execution speedup moving from K20X to P100 is 5.3 times, and the disaggregation kernel speedup is the largest, at 8 times. On both K20X and P100, this kernel's performance is limited by the amount of CUDA shared memory it requires. In S822LC, the newer Pascal GPU architecture provides 64 KB of shared memory per thread-block rather than the 48 KB on XK, which allows more thread-blocks to run concurrently and provide the disproportionate speedup on that machine.

III. Conclusions

This paper compares multi-GPU MLFMM performance on two types of computer systems: 16 nodes of the Blue Waters supercomputer and an IBM S822LC workstation. MLFMM operations are realized as matrix operations for excellent performance on GPUs. Significant CPU speedup on both systems is achieved with OpenMP, and further eclipsed by CUDA implementations that take advantage of well-understood matrix optimization techniques. A speedup of 969 times over single-threaded CPU execution is achieved on S822LC, bringing execution times from seconds to milliseconds for large problems. This speedup justifies the significant time investment for the multi-GPU implementation.

Acknowledgments

This work was supported by the NVIDIA GPU Center of Excellence, the NCSA Petascale Improvement Discovery Program, and the IBM-Illinois Center for Cognitive Computing Systems Research (C3SR).

References

[1] W. C. Chew, J.-M. Jin, E. Michielssen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*. Boston: Artech House, 2001.

[2] W.-M. W. Hwu, *GPU Computing Gems Emerald Edition*. Elsevier, 2011.

[3] National Center for Supercomputing Applications, Urbana, IL, USA "Blue Waters System Summary," [online] Available: https://bluewaters.ncsa.illinois.edu/hardware-summary, Accessed on: May 8, 2017.

[4] International Business Machine Corporation, New York, USA, "IBM Power System S822LC for High Performance Computing," [online] Available: http://www-03.ibm.com/systems/power/hardware/s822lc-hpc, Accessed on: June 4, 2017.