Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **6**, No. 1, pp. 1-23

DOI: 10.4208/aamm.2013.m87
February 2014

# Numerical Study of Geometric Multigrid Methods on CPU–GPU Heterogeneous Computers

Chunsheng Feng[1], Shi Shu[2,*], Jinchao Xu[3] and Chen-Song Zhang[4]

[1] *Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Xiangtan University, Xiangtan 411105, China*
[2] *Key Laboratory of Intelligent Computing and Information Processing of Ministry of Education, Xiangtan University, Xiangtan 411105, China*
[3] *Department of Mathematics, Pennsylvania State University, PA, USA*
[4] *NCMIS and LSEC, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100190, China*

Received 12 January 2013; Accepted (in revised version) 18 July 2013

Available online 13 December 2013

**Abstract.** The geometric multigrid method (GMG) is one of the most efficient solving techniques for discrete algebraic systems arising from elliptic partial differential equations. GMG utilizes a hierarchy of grids or discretizations and reduces the error at a number of frequencies simultaneously. Graphics processing units (GPUs) have recently burst onto the scientific computing scene as a technology that has yielded substantial performance and energy-efficiency improvements. A central challenge in implementing GMG on GPUs, though, is that computational work on coarse levels cannot fully utilize the capacity of a GPU. In this work, we perform numerical studies of GMG on CPU–GPU heterogeneous computers. Furthermore, we compare our implementation with an efficient CPU implementation of GMG and with the most popular fast Poisson solver, Fast Fourier Transform, in the cuFFT library developed by NVIDIA.

**AMS subject classifications**: 65M10, 78A48

**Key words**: High-performance computing, CPU–GPU heterogeneous computers, multigrid method, fast Fourier transform, partial differential equations.

## 1 Introduction

Simulation-based scientific discovery and engineering design demand extreme computing power and high-efficiency algorithms [7,8,31,33,34]. This demand is one of the main

*Corresponding author.
*Email:* spring@xtu.edu.cn (C. S. Feng), shushi@xtu.edu.cn (S. Shu), xu@math.psu.edu (J. Xu), zhangcs@lsec.cc.ac.cn (C.-S. Zhang)

forces driving the pursuit of extreme-scale computer hardware and software during the last few decades. It has become increasingly important for algorithms to be well-suited to the emerging hardware architecture. In fact, the co-design of *architectures*, *algorithms*, and *applications* is particulary important given that researchers are trying to achieve exascale ($10^{18}$ floating-point operations per second) computing. Although the question of what is the best computer architecture to achieve exascale or higher remains highly debatable, many researchers agree that hybrid architectures make sense due to energy-consumption constraints. There are already quite a few heterogeneous computing architectures available, including the Cell Broadband Engine Architecture (CBEA), Graphics Processing Units (GPUs), and Field Programmable Gate Arrays (FPGAs) [15,18,50].

A GPU is a heterogeneous co-processor that can be accessed and controlled by CPUs. The Intel/AMD CPU accelerated by NVIDIA/AMD GPUs is probably the most commonly used heterogeneous high-performance computing (HPC) architecture at present. GPU-accelerated supercomputers feature in many of the top computing systems in the HPC Top500 [46] and the Green500 [25]. Some "old" supercomputers, such as JAGAUR (now known as TITAN) of the Oak Ridge National Laboratory, are being redesigned in order to incorporate GPUs and thereby achieve better performance. GPUs have evolved from fixed-pipeline application-specific integrated circuits into highly programmable, versatile computing devices. Under conditions often met in scientific computing, modern GPUs surpass CPUs in computational power, data throughput, and computational efficiency per watt by almost one order of magnitude [16,20,40].

Not only are GPUs the key ingredient in many current and forthcoming petaflop supercomputers, they also provide an affordable desktop supercomputing environment for everyday usage, with peak computational performance matching that of the most powerful supercomputers of only a decade ago. General-purpose graphics processing units (GPGPU), as a high-performance computational device are becoming increasingly popular. NVIDIA Kepler GPU and the Intel Many Integrated Core (MIC) architecture are the most promising heterogeneous co-processors with high energy-efficiency and computation power. The Intel Xeon Phi Coprocessor 5110P (60 cores, MIC architecture) is capable of delivering 1 Teraflop operation in double precision per second, whereas the peak performance of Tesla K20X (2688 cores) is 1.31 Teraflop operations in double precision per second.

One of the most discussed features of the MIC architecture is that it shares the x86 instruction set such that users often assume that they do not need to change their existing codebase in order to migrate to MIC. However, this assumption is subject to argument as even if legacy code can easily be migrated, whether the application it is then used for is able to achieve the desired performance is highly questionable. Achieving scalable scientific applications in the exascale era is our ultimate goal. Hence, software, more importantly algorithms, must adapt to unleash the power of the hardware. Unfortunately, none of the processors envisioned at present will relieve today's programmers from the hard work of preparing their applications. In fact, power constraints will actually cause us to use simpler processors at lower clock rates for the majority of our work. As an