INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 4, Number 4, Pages 394–412



# GPU COMPUTING FOR MESHFREE PARTICLE METHOD

M. PANCHATCHARAM[⋆,⩘], S. SUNDAR[⋆], V. VETRIVEL[⋆], A. KLAR[⩘], AND S. TIWARI[⩘]

**Abstract.** Graphics Processing Units (GPUs), originally developed for computer games, now provide computational power for scientific applications. A study on the comparison of computational speed-up and efficiency of a GPU with a CPU for the Finite Pointset Method (FPM), which is a numerical tool in Computational Fluid Dynamics (CFD) is presented. As FPM is based on the point cloud, it is so expensive when the number of particles are in millions. We have demonstrated the application of the FPM using a single-GPU (Nvidia Tesla M2050) and Intel CPU (Dual Xeon). Importance of the GPU is realized by the FPM since GPU yields a computational speed-up of 70× for the Poisson equation with various boundary conditions.

**Key words.** Finite Pointset Method(FPM), CUDA, GPU, Bi-CGSTAB.

## 1. Introduction

Nowadays, computational methods and related hardware are really inseparable. The hardware architecture progress leads the numerical methods that can be used with a reasonable computational cost. To increase the computational ability of CPU, a large and expensive cache is integrated and a many-core design has been employed [10]. The small scale problems of CFD can be solved on a PC of multi-core CPU and shared-memory parallel programming. For large scale problem, a PC with few cores cannot offer enough computational capability, and a cluster with many CPUs (or cores) is needed. Nevertheless, the memory bottleneck which appears in the form of bandwidth limitation and fetching latency, has restricted the performance of the many-core systems. In the meantime, Graphics Processing Units (GPUs), having recently turned into general-purpose programmable units, can provide a different solution to the memory access problem.

Initially driven by the gamer market, GPUs recently became suitable for high performance computing applications. A GPU is a multi-threaded, many core processors which was originally developed for graphics processing. However, in recent years, the so-called General Purpose GPU (GPGPU) has been used widely for computation in different fields because it has a high computing ability and a relatively low cost. The main advantage of GPUs is their ability to perform significantly more floating point operations (FLOPS) per unit time than a CPU. One of the market leaders, NVIDIA, developed a parallel computation architecture called CUDA (Compute Unified Device Architecture) [7]. CUDA is an extension of C language which allows us to program the NVIDIA GPUs in an easy way. Other than NVIDIA, there are several ways to realize the GPGPU computing: Computer Graphics with OpenGL [6], OpenCL [12], Stream (ATI Corporation) [1]. But according to Du et al. [5], at this moment CUDA is more efficient on the GPU than OpenCL. Hence, in our study, the NVIDIA GPU with CUDA platform is chosen because CUDA is used in a variety of different fields of scientific computation such as graphics, biology, linear algebra, PDE solvers and computational physics. Especially in

Received by the editors October 31, 2012 and, in revised form, August 31, 2013.
2000 *Mathematics Subject Classification.* 65Y05, 65Y20, 35Q30, 76D05.
This research was supported by DAAD.

the applications of computational mathematics and computational physics, it obtained surprising speed-up. Harada et al. [8] obtained 28× speed-up on GPU for Smoothed Particle Hydrodynamics (SPH) solver using 262,144 particles. Rossinelli et al. [18] present a GPU-accelerated solver for simulations of bluff body flows in 2D using a re-meshed vortex particle method and 30× speed-up was obtained. Knibbe et al. [9] implemented a Krylov solver on GPU with a 30× speed-up. Luo et al. [11] obtained 71× speed-up on GPU for 1D shock tube problem using CESE (Conservation Element and Solution Element) method.

The Finite Pointset method (FPM) [20] is a mesh free method for solving fluid dynamic equations. It is a Lagrangian particle method, in which computational domain is filled with a finite number of particles. Particles move with fluid velocities and they carry the fluid quantities, like the density, the velocity, the pressure and so on. Similarly, boundaries are also approximated by a finite number of boundary particles and boundary conditions are prescribed for them. Since it is a Lagrangian particle method, the distribution of particles can be arbitrary. The FPM has more advantages on fluid flow problems with moving boundary in time, where the remeshing is not required. The disadvantage of the FPM is the constant particle management at each time step of a flow problem, otherwise, it may lead to inaccuracy or instability. Drum et al. [4], Tiwari and Kuhnert [21] have applied the FPM for various fluid flow problems. Our main goal is to solve the incompressible Navier-Stokes equations by the FPM in GPU, however, we restrict ourselves to solve the Poisson equation, since the pressure Poisson equation is the essential part of solving the incompressible Navier-Stokes equations. Therefore, in this paper, we focus on the iterative solvers for the Poisson equation in two dimensions on GPU using CUDA.

The particles generated in CPU and transferred to the GPU for neighbor search computations. In this paper, we implemented the non-recursive merge-sort algorithm to sort the neighbors.

Finally, the computation in CPU produces a system of equations, where the matrix is sparse and has no blocks. This system is solved in GPU using the Bi-CGSTAB solver from *cusp* library.

The rest of the paper is organized as follows: Section 2 is the overview of the Finite Pointset method. In section 3, key facts of CUDA are provided. The specific aspects of the GPU implementation of Krylov methods such as Bi-CGSTAB using *cusp* library and non-recursive merge-sort algorithm are considered in detail in Section 4. Section 5 will represent the numerical results for some benchmark problems and analyze the computational performance. Finally conclusions are drawn in Section 6.

## 2. The Finite Pointset Method

In this paper, we restrict ourselves for solving the Poisson equation in $\bar{\Omega}(\subset R^2)$. Let us consider the following boundary value problem

$$\Delta u = f \quad \text{in} \quad \Omega \tag{1}$$

$$a_0 u + b_0 \frac{\partial u}{\partial \vec{n}} = g \quad \text{on} \quad \partial\Omega, \tag{2}$$

where $a_0, b_0, f$ and $g$ are given functions and $\vec{n}$ is the unit normal on boundary pointing inside the domain. Here, $(a_0, b_0) = (1, 0)$ denotes the Dirichlet boundary