Future Generation Computer Systems 56 (2016) 759–765

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A hybrid solution method for CFD applications on GPU-accelerated hybrid HPC platforms

Xiaocheng Liu [a,1], Ziming Zhong [b,*,1], Kai Xu [a]

[a] College of Information System and Management, National University of Defense Technology, Changsha, Hunan, China
[b] Complex Aviation Systems Simulation Laboratory, Beijing, China

HIGHLIGHTS

- We propose a hybrid solution method for CFD applications for CPU+GPU platforms.
- We propose a domain decomposition method based on the functional performance model.
- We evaluate the proposed methods with the lid-driven cavity flow application.

ARTICLE INFO

Article history:
Received 9 April 2015
Received in revised form
16 July 2015
Accepted 4 August 2015
Available online 28 August 2015

Keywords:
Computational fluid dynamics
GPU-accelerated multicore system
Performance modeling
Hybrid computing
Data partitioning

ABSTRACT

Heterogeneous multiprocessor systems, where commodity multicore processors are coupled with graphics processing units (GPUs), have been widely used in high performance computing (HPC). In this work, we focus on the design and optimization of Computational Fluid Dynamics (CFD) applications on such HPC platforms. In order to fully utilize the computational power of such heterogeneous platforms, we propose to design the performance-critical part of CFD applications, namely the linear equation solvers, in a hybrid way. A hybrid linear solver includes both one CPU version and one GPU version of code for solving a linear equations system. When a hybrid linear equation solver is invoked during the CFD simulation, the CPU portion and the GPU portion will be run on corresponding processing devices respectively in parallel according to the execution configuration. Furthermore, we propose to build functional performance models (FPMs) of processing devices and use FPM-based heterogeneous decomposition method to distribute workload between heterogeneous processing devices, in order to ensure balanced workload and optimized communication overhead. Efficiency of this approach is demonstrated by experiments with numerical simulation of lid-driven cavity flow on both a hybrid server and a hybrid cluster.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Computational Fluid Dynamics (CFD) is the analysis of systems involving fluid flow, heat transfer, and associated phenomena by means of computer-based numerical simulation [1]. Over the past few decades, CFD has become a practical cornerstone of most fluid and mechanical engineering applications, such as aerodynamics of aircraft and vehicles.

The governing equations of fluid motion are partial differential equations, which are difficult to solve analytically. Therefore, numerical methods are usually used. CFD simulations commonly require a significant amount of computational resources for accurate solutions, especially for complex simulation scenarios such as transonic or turbulent flows. With high performance computing (HPC) platforms, faster and better solutions can be achieved. Therefore, parallelization of numerical simulations of fluid dynamics is an attractive research topic from the very beginning in this area.

The current trend in gaining high computing power is to incorporate specialized processing resources such as graphics processing units (GPUs) in multicore systems, thus making a computing system heterogeneous. Over recent years, a large number of critical scientific software have been ported to multicore and GPU architectures. Meanwhile, there has been extensive research on utilizing hybrid GPU-accelerated multicore platforms. However, most of the state-of-the-art CFD packages, e.g. OpenFOAM [2], are designed for facilitating the implementation of CFD applications on traditional parallel platforms based on CPU processors. Although

* Corresponding author.
E-mail address: zzm_nudt@163.com (Z. Zhong).
[1] Both authors contributed equally to this work.

http://dx.doi.org/10.1016/j.future.2015.08.002
0167-739X/© 2015 Elsevier B.V. All rights reserved.

many efforts have been invested in improving the performance of CFD simulations on traditional HPC platforms, more is required for hardware-accelerated hybrid HPC platforms.

In this work, we focus on the design and optimization of CFD applications on GPU-accelerated parallel platforms. In a CFD application, the performance-critical part is its linear equation solvers. In order to exploit the computational power of target platforms, we propose to develop hybrid linear equation solvers. A hybrid linear equation solver consists of both one CPU version and one GPU version of code required for solving a linear equations system. The CPU version and GPU version of code are implemented using the programming languages and optimized numerical libraries for CPU and GPU architectures respectively. When a hybrid linear equation solver is invoked during the simulation, the CPU portion and the GPU portion will be run on corresponding processing devices respectively in parallel according to the execution configuration. During the simulation, a portion of the workload will be offloaded to GPU to accelerate the computation, and the result will be uploaded the host when the computation is finished.

Parallel computing in CFD simulations is usually based on domain decomposition, which is essentially data parallelism [1]. To achieve the maximum performance of CFD applications on hybrid HPC platforms, it is important to balance the workload and minimize the volume of communication among processing devices. To this end, we propose a method of heterogeneous domain decomposition. First, by benchmarking the linear equation solvers of a CFD application, we built the functional performance models [3] of processing devices. The FPMs are constructed empirically, and integrate many important features characterizing the performance of both the architecture and the application [4]. Then, the relative speeds of processing devices are calculated using FPM-based data partitioning algorithms, which have been proved accurate and efficient in [5]. Last, the relative speeds are provided to graph partitioning softwares, such as Metis [6], Scotch [7], as input (weights) to subdivide the domain into multiple subdomains, each assigned to one processing device. The graph partitioning softwares guarantee a balanced workload and an optimized communication overhead between heterogeneous processing devices. The problem will then be solved on the entire domain from problem solutions on subdomains.

In this work, without loss of generality, we experiment with a typical CFD application, namely the numerical simulation of the lid-driven cavity flow, on both a GPU-accelerated multicore node and a cluster of hybrid nodes. We develop the hybrid Conjugate Gradient (CG) [8] and Bi-Conjugate Gradient Stabilized (BiCGSTAB) [9] linear equation solvers for GPU-accelerated computing systems, build functional performance models of the processing devices of the experimental platforms, and decompose the domain based on these performance models. Experimental results demonstrate that the proposed heterogeneous domain decomposition methods based on performance models are able to balance the workload and deliver good performance.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 briefly describes the computational fluid dynamics, the numerical methods and its parallelization. Section 4 presents the proposed hybrid solution method. Section 5 presents the experimental settings and results. Section 6 concludes the paper.

2. Related work

2.1. Computational fluid dynamics

Over the past few decades, a number of CFD packages have been developed, such as OpenFOAM [2], FEniCS [10], PyFR [11]. Most of them are aimed at homogeneous parallel platforms, and use the domain decomposition method to realize process-level parallelism of CFD applications. With the advent of GPU computing [12], many works have been carried out on developing linear equation solvers on GPUs for the speedup of CFD applications [13–15]. The recent progress and challenges in exploiting graphics processors in computational fluid dynamics are reviewed in [16]. Furthermore, there have also been some study on optimizing the CFD applications on GPU-accelerated HPC platforms. For example, [17] presents a high-order flow solver using multi-block structured grids on GPU clusters and propose a workload balancing model for distributing the workload between CPUs and GPUs. [18] proposes methods for designing and optimizing OpenFOAM-based CFD applications on hybrid and heterogeneous HPC platforms.

2.2. Performance models

Performance modeling is a very common technique used for optimization of scientific applications on parallel HPC platforms. A large number of performance models have been proposed for multicore and GPU architectures, such as the Roofline model [19] for multicore processors, and the integrated power and performance prediction model for GPU architecture [20]. The functional performance model (FPM) [5] represents the processor speed by a continuous function of the problem size. The speed is defined as the number of computation units performed per one time unit, and is found experimentally by measuring the execution time. A functional performance model consists of a series of speed measurements taken over a range of problem sizes. In many computationally intensive applications, the performance-critical part that is representative of the whole application in terms of performance can be extracted and used for performance measurement [21].

2.3. Load balancing algorithms

The load balancing algorithms can be classified into two categories, namely *static* and *dynamic* algorithms. Static algorithms, such as data partitioning [22–24], require a priori information about the parallel application and platform. Static algorithms rely on accurate performance models as input to predict the future execution of the application. Static algorithms are particularly useful for applications for which data locality is critical because data redistribution incurs significant overhead. However, these algorithms are unable to balance on non-dedicated platforms, where load changes with time. Dynamic algorithms, such as task scheduling and work stealing [25–30], balance the load by moving fine-grained tasks between processors during the execution. Dynamic algorithms do not require a priori information about execution but may incur large communication overhead due to data migration. A number of FPM-based data partitioning algorithms have been proposed, such as the geometric partitioning algorithm [5] and the dynamic data partitioning algorithm [31]. In this work, we use the FPM-based data partitioning to calculate the accurate relative speeds of processing devices of heterogeneous platforms.

2.4. Graph partitioning

Several software packages for graph partitioning have been developed, which can be used to partition the graph representing a system of sparse linear equations. The goal is to subdivide the graph into smaller subgraphs in order to balance the workload and to minimize the volume of communication among processors. Graph partitioning algorithms implemented in Metis [6], Scotch [7], Jostle [32] reduce the number of edges between the target subdomains, aiming to minimize the total communication cost

Download English Version:

https://daneshyari.com/en/article/424930

Download Persian Version:

https://daneshyari.com/article/424930

Daneshyari.com