Computer Physics Communications 186 (2015) 39–47

Contents lists available at ScienceDirect

Computer Physics Communications

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

Application of GPU processing for Brownian particle simulation

Way Lee Cheng, Ali Sheharyar, Reza Sadr*, Othmane Bouhali

Texas A&M University at Qatar, Doha, Qatar

ARTICLE INFO

Article history:
Received 1 January 2014
Received in revised form
24 July 2014
Accepted 15 September 2014
Available online 22 September 2014

Keywords:
Nanofluids
GPU
Simulations
CFD
Multiphase flows

ABSTRACT

Reports on the anomalous thermal-fluid properties of nanofluids (dilute suspension of nano-particles in a base fluid) have been the subject of attention for 15 years. The underlying physics that govern nanofluid behavior, however, is not fully understood and is a subject of much dispute. The interactions between the suspended particles and the base fluid have been cited as a major contributor to the improvement in heat transfer reported in the literature. Numerical simulations are instrumental in studying the behavior of nanofluids. However, such simulations can be computationally intensive due to the small dimensions and complexity of these problems. In this study, a simplified computational approach for isothermal nanofluid simulations was applied, and simulations were conducted using both conventional CPU and parallel GPU implementations. The GPU implementations significantly improved the computational performance, in terms of the simulation time, by a factor of 1000–2500. The results of this investigation show that, as the computational load increases, the simulation efficiency approaches a constant. At a very high computational load, the amount of improvement may even decrease due to limited system memory.

1. Introduction

Effective cooling is one of the most challenging research subjects in modern day engineering, as the power density in many industrial applications continues to increase. Researchers have recently reported that suspending nano-sized particles in a base fluid, forming a nanofluid, improves the thermal conductivity of the fluid [1]. The anomalous conductivity enhancement of nanofluids may be instrumental in further enhancements of the cooling systems in many industrial and engineering technologies [2,3]. This step is essential in further developing small-scale devices due to their higher power density requirements. However, the mechanisms leading to the improved heat-transfer properties of nanofluids are not well understood.

Various theories have been proposed in the literature, but there is no solid conclusion to date [4], and the topic is a controversial subject yet to be resolved [5]. Further investigations are therefore required to better understand these anomalous behaviors of nanofluids. It has been suggested that the random movements of suspended nanoparticles in the fluid assist heat transfer and therefore improve the thermal conductivity of nanofluids. Several authors [6,7] have reported that Brownian motion has a strong effect on the thermal conductivity of nanofluids. Chon et al. [8] and Li and Peterson [9] both determined that Brownian motion has a significant effect on the higher thermal conductivity in nanofluids.

Numerical modeling has been a popular tool to study thermal-fluid transport problems. Simulations can provide a simple approach to better understand the physics of the thermal conductivity improvement of nanofluids by examining the effects of various parameters on the problem. Different numerical approaches have been applied for nanofluid computations, such as molecular dynamics [10], single-phase models using effective properties [11,12], and discrete particle modeling [13–15]. However, due to the scale of the dimensions and the non-linear nature of the problems in these studies, the simulations are computationally expensive. Cheaper and more efficient computations are the two keys to further promoting the use of simulations in nanofluid studies. These aims can be achieved by using more efficient algorithms and/or high-performance computing applications, such as supercomputing and/or graphic processing units.

Graphic processing units (GPUs) are multi-threaded parallel units containing hundreds of cores, originally designed for the highly parallel process of graphic rendering. They have been introduced as cost-effective alternatives to expensive high-performance computing. Within the last decade, GPUs have been converted into a low-cost, highly capable computing solution in scientific research such as computational fluid dynamics (CFD). Double precision capability was introduced in the last five years, which has further enhanced GPU applications in the field. Meanwhile, general purpose GPU programming is maturing,

* Corresponding author. Tel.: +1 4044185420.
E-mail address: reza.sadr@qatar.tamu.edu (R. Sadr).

http://dx.doi.org/10.1016/j.cpc.2014.09.013
0010-4655/

Nomenclature

C	Arbitrary constant
D	Diffusivity of particles
Fo	Fourier number
k_B	Boltzmann constant ($1.3806488 \times 10^{-23}$ J/K)
m	Mass
P	Pressure
r	Radial distance
R	Particle radius
Re	Reynolds number
ΔS	Displacement of particles
t	Elapsed time since particle movement
Δt	Size of the time-step
T	Temperature
U	Velocity of the discrete phase
V	Velocity of the continuous phase

Greek Alphabets

δ	Distance from the particle surface
Λ	Sample size
η	Speed-up factor
μ	Dynamic viscosity
ν	Kinematic viscosity
ρ	Density
σ	Standard deviation
τ	Relaxation time
$\Delta\tau$	Size of the sub-time-step
φ	Polar angle
ψ	Stream-function
ω	Random number (normally-distributed)

Subscripts

0.05	Location of 5% of the free-stream velocity from the particle surface
D	Diameter
fluid	Fluid
max	Maximum
part	Particle
r	Radial distance
SS	Steady-state
TS	Transient
θ	Azimuthal angle
φ	Polar angle

Superscripts

*	Dimensionless quantity
$\bar{\alpha}$	Mean of quantity α

which is essential for designing various numerical GPU solvers. One of the most common GPU programming environments to date is Compute Unified Device Architecture (CUDA) [16]. Early computational works used GPU computations in fluid dynamics with a finite volume solver [11,17] or finite element methods [18]. Garland et al. [19] reviewed CUDA implementations in a variety of computational problems. The authors reported that, depending on the mesh density used in the calculation, a single GPU can be 16–22 times faster than with conventional serial CPU calculation when solving the compressible two-dimensional Euler equation. Several recent studies have used multi-GPU implementations to solve multiphase incompressible flow with Navier–Stokes equations. Cohen and Molemaker [20] demonstrated one of the first multi-GPU implementations of a buoyancy-driven turbulence problem and reported an eight-fold improvement in the computation speed. Thibault and Senocak [21] proposed the first GPU-cluster implementation of the Navier–Stokes equations and reported the possibility of reducing the computational time up to 100 fold when using a multi-GPU cluster to solve the three-dimensional lid-driven cavity flow problem. Jacobsen et al. [22,23] proposed a distributed-memory parallel implementation in solving the Navier–Stokes equations with the multi-grid method.

There are also recent developments in extending GPU applications for multiphase problems. Kelly [24] and Kuo et al. [25] proposed parallel algorithms implemented on a single GPU for solving two-phase flow problems. Shigeto and Sakai [26] proposed an algorithm for a discrete element method of multi-thread parallel computing on multi-core processors for powder dispersion in a cylindrical drive. They reported up to a 3.5-fold improvement in the computation time and noted that the limited improvement was attributed to their hardware restrictions. Griebel and Zaspel [27] proposed a multi-GPU parallel implementation to compute an air bubble rising in water. They reported an 11-fold improvement in the computation time over conventional serial CPU computation when using a single GPU for the computation. Griebel and Zaspel [27] also demonstrated a 70-fold improvement in the simulation time when using an 8-GPU cluster for the same problem and later reported an additional 30% reduction in the computation time by further refining their code [28]. Chen et al. [29] proposed an algorithm for molecular dynamic simulation using GPU and reported 20- to 60-fold faster computations.

Most nanofluid simulations are computationally expensive. It is very likely that GPU usage can also significantly reduce the time consumption for complicated multiphase computations such as nanofluid simulations. However, GPU applications aimed at complex multiphase-fluid (including nanofluids) simulations are still in their infancy, and only a limited number of studies have been published on this subject [29–31]. In particular, most of these works are focused on liquid–liquid or solid–gaseous systems that might not be applicable to solid–liquid, two-phase systems such as nanofluids.

In this study, nanofluid simulations are conducted using both conventional serial CPU and parallel GPU implementations, and their computational performances are compared. The hydrodynamic characteristics of the nanofluids are examined from a statistical perspective, using a discrete-phase model. An algorithm is developed for solving the nanofluid model on GPUs. The computational performance of the GPU for different computational loads is demonstrated by comparing the speed-up factor (a metric for measuring the computation speed) between the GPU and conventional CPU algorithms. The powers and limitations of the GPU algorithm implementations are then discussed.

2. Mathematical setup

Hydrodynamics of moving particles in a quiescent viscous fluid

A nano-sized particle of radius R, moving at a constant velocity U_{part}, through a quiescent viscous fluid generates a flow field in the surrounding fluid, at constant temperature, governed by the Stokes equation. This induced fluid flow, also known as creeping flow, is characterized by a small Reynolds number ($Re \to 0$). Happel and Brenner [32] derived the steady-state solution, in spherical coordinates, to this flow field in terms of the stream-function, ψ:

$$\psi = -\frac{1}{4} U_{part} \cdot r^2 \cdot \sin^2\varphi \cdot \left[\left(\frac{R}{r}\right)^3 - 3\left(\frac{R}{r}\right) \right]. \quad (1)$$