Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 5342–5359
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

General purpose molecular dynamics simulations fully implemented on graphics processing units

Joshua A. Anderson [a,*], Chris D. Lorenz [b], A. Travesset [a]

[a] *Ames Laboratory and Department of Physics and Astronomy, Iowa State University, Ames, IA 50011, USA*
[b] *Materials Research Group, Engineering Division, King's College, London Strand, London WC2R 2LS, UK*

Received 21 September 2007; received in revised form 24 January 2008; accepted 29 January 2008
Available online 8 February 2008

Abstract

Graphics processing units (GPUs), originally developed for rendering real-time effects in computer games, now provide unprecedented computational power for scientific applications. In this paper, we develop a general purpose molecular dynamics code that runs entirely on a single GPU. It is shown that our GPU implementation provides a performance equivalent to that of fast 30 processor core distributed memory cluster. Our results show that GPUs already provide an inexpensive alternative to such clusters and discuss implications for the future.

PACS: 02.70.Ns; 61.20.Ja; 61.25.He

Keywords: Graphics processing unit; GPU; NVIDIA; CUDA; Molecular dynamics; Polymer systems

1. Introduction

Fuelled by the dramatic increases in computing power over the years, the impact of computational methods in the traditional sciences has been gigantic. Yet, the quest for new technologies that enable faster and cheaper calculations is more fervent than ever, as there are a vast number of problems that are on the brink of being solved if only a relatively modest increase in computer power were available.

Molecular dynamics (MD) has emerged as one of the most powerful computational tools [2], as it is capable of simulating a huge variety of systems both in and out of thermodynamic equilibrium. During the last decade, general purpose MD codes such as LAMMPS [3], DLPOLY [4], GROMACS [5], NAMD [6] and ESPResSO [7] have been developed to run very efficiently on distributed memory computer clusters. More recently, graphics processing units (GPUs), originally developed for rendering detailed real-time visual effects in computer games, have become programmable to the point where they are a viable general purpose programming platform. Dubbed GPGPU for general purpose programming on the GPU, it is currently getting a lot of attention

* Corresponding author.
E-mail address: joaander@ameslab.gov (J.A. Anderson).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.01.047

in the scientific community due to the huge computational horsepower of recent GPUs, evident in Fig. 1 [8–13]. The use of GPGPU techniques as an alternative to distributed memory clusters in MD simulations has become a real possibility.

Until recently, the only way to make use of the GPU's abilities was to carefully cast the algorithm and data structures to be represented as individual pixels being written to an image via *fragment shaders*. In addition to the cumbersome nature of programming this way, there are various other limitations imposed. Perhaps the most severe is that each thread of execution can only write a single output value to a single memory location in a gathered fashion. Scattered writes to multiple memory locations are important in implementing a number of algorithms, including parts of molecular dynamics. Likely because of this limitation, all the early implementations of MD using GPUs have been based on a mixed approach. The GPU performs those computationally intensive parts of MD that can be implemented in a gather implementation, and the CPU handles the rest [8,9]. A typical MD simulation implemented on the CPU spends only 50–80% of the total simulation time performing these gather operations. So the speedup by using a mixed CPU/GPU approach is limited to a factor of 2 or 3 at most, even though, as shown in Fig. 1, the GPU has the ability to perform significantly more floating point operations (FLOPs) per unit time than a CPU, thus leaving room for a dramatic speedup.

In this paper we provide, to our knowledge, the first implementation of a general purpose MD code where all steps of the algorithm are running on the GPU. This implementation is made possible by the use of the NVIDIA® CUDA™ C language programming environment. CUDA provides low level hardware access, avoiding the limitations imposed in fragment shaders. It works on the latest G80 hardware from NVIDIA, and will be supported on future devices [1]. Algorithms developed for this work will be directly applicable to newer, faster GPUs as they become available. After the initial submission of this paper, it came to our attention that van Meel et al. [13] submitted a similar work nearly simultaneously, where they also used CUDA to put all the steps of MD onto the GPU. The differences in implementation and performance are discussed in Section 2.7.

Early on in the development, it was decided not to take an existing MD code and modify it to run on the GPU. Any existing package would simply pose too many restrictions on the underlying data structures and the order in which operations are performed. Instead, a completely fresh MD code was built from the ground up with every aspect tuned to make the use of the GPU as optimal as possible. Despite these optimizations, every effort was made to develop a *general* architecture in the code so that it can easily be expanded to implement any of the features available in current general purpose MD codes.

This work includes algorithms for a very general class of MD simulations. N particles, in either an NVE or NVT ensemble, are placed in a finite box with periodic boundary conditions, where distances are computed according to the minimum image convention [14]. Because our own interests are the simulation of non-ionic polymers [15], non-bonded short-range and harmonic bond forces are the only interactions currently

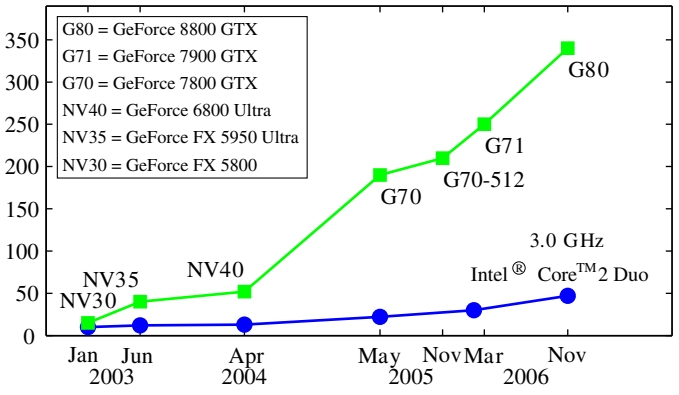

Fig. 1. Performance of CPUs (blue circles) and GPUs (green squares) over the last few years. Figure courtesy of NVIDIA, and adapted from Ref. [1]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)