Available online at www.sciencedirect.com
ScienceDirect
Journal of Structural Biology 157 (2007) 288–295
Journal of Structural Biology
www.elsevier.com/locate/yjsbi

Implementation and performance evaluation of reconstruction algorithms on graphics processors

Daniel Castaño Díez, Hannes Mueller, Achilleas S. Frangakis *

European Molecular Biology Laboratory, Meyerhofstr. 1, 69117 Heidelberg, Germany

Received 15 June 2006; received in revised form 16 August 2006; accepted 25 August 2006
Available online 1 September 2006

Abstract

The high-throughput needs in electron tomography and in single particle analysis have driven the parallel implementation of several reconstruction algorithms and software packages on computing clusters. Here, we report on the implementation of popular reconstruction algorithms as weighted backprojection, simultaneous iterative reconstruction technique (SIRT) and simultaneous algebraic reconstruction technique (SART) on common graphics processors (GPUs). The speed gain achieved on the GPUs is in the order of sixty (60×) to eighty (80×) times, compared to the performance of a single central processing unit (CPU), which is comparable to the acceleration achieved on a medium-range computing cluster. This acceleration of the reconstruction is caused by the highly specialized architecture of the GPU. Further, we show that the quality of the reconstruction on the GPU is comparable to the CPU. We present detailed flow-chart diagrams of the implementation. The reconstruction software does not require special hardware apart from the commercially available graphics cards and could be easily integrated in software packages like SPIDER, XMIPP, TOM-Package and others.

Keyword: Image processing

1. Introduction

The data amount in electron microscopy increases constantly with the aim to produce images with improved resolution. This increase is boosted by recent hardware developments, e.g. of the CCD cameras (more pixels and larger field of view), as well as the automatization of the data recording, both in electron tomography and single particle analysis (Carragher et al., 2000) (Plitzko et al., 2002) (Zhu et al., 2004). In order to cope with these increasing computational efforts, most of the software packages used in electron microscopy have implemented parallel reconstruction algorithms to be used on distributed computing systems (computing clusters) (Frank et al., 1996) (Sorzano et al., 2004). Electron microscopy data are typically well suited for parallelization, since both images and processes can be treated independently, e.g. electron micrographs can be backprojected in a common three-dimensional (3D) image independently, or the angular parameters of a 3D search of an atomic structure in a density map can be distributed on various processors. The parallelization of these applications therefore results in an almost linear decrease of the computational time, and a linear increase in the purchasing costs.

The new (fourth) generation of graphical processing units (GPUs) shows an impressive performance in certain vertex manipulating operations; the GPUs are capable of performing float-point operations and most importantly they provide both vertex-level and pixel-level programmability (Fernando and Kilgard, 2003). This level of programmability opens up the possibility of performing complex

* Corresponding author.
E-mail addresses: castano@embl.de (D. Castaño Díez), hamuelle@embl.de (H. Mueller), frangak@embl.de (A.S. Frangakis).

1047-8477/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jsb.2006.08.010

calculations on the GPU instead on the central processing unit (CPU), a technique known as GPGPU (General Purpose Computing on Graphics Processing Units). The GPU is a vector processor, which can perform certain operations significantly faster than the CPU. Vector processors were common in the scientific community, being the fastest computers through the 1980s into the 1990s. However, general needs for an increase of the flexibility of the CPU resulted to an almost disappearance of the vector processors, which currently are included as some small processing elements into the common general-purpose CPUs. The architecture of modern GPU relies heavily on vector processors. The main architectural difference between the CPU and the GPU is that the CPU is pipelining only the instructions for processing a data set, while the GPU is pipelining both the data and the instructions (Fernando and Kilgard, 2003). The principle thereby is that the same instruction is being applied on the whole data set, without being decoded for every data point as in the CPU. This makes the calculation of data that can be represented in a pipeline way (or in a vectorized way) extremely effective and therefore fast, however with no flexibility. Therefore, next to the GPU a CPU is necessary, in order to run the operating system and application programs, as well as to control the GPU.

In the last years, the development of cheap (in the order of some hundred Euros) GPUs was mainly driven by the gaming industry. The result was an unparalleled increase in the performance of the GPUs surpassing the constant redoubling of the number of transistors integrated in the CPU every 24 months by far (known as the Moore's law) (Moore, 1965). A modern GPU has 4 times as many transistors as a modern CPU (Fernando and Kilgard, 2003). With the advent of the programmable GPUs, some algorithms, e.g. backprojection, Fast Fourier Transform (FFT), etc. have been implemented on the GPU, mainly for applications in medical computed tomography, with significant running time gains (Xu and Mueller, 2005) (Schiwietz et al., 2006). However, most of them were commercial implementations also relying on custom hardware, with no flexibility for the researcher to manipulate the code (e.g. Visage RT Image Reconstruction by Mercury Computer Systems Inc. which can use not only the GPU, but also the CPU and FPGAs if available http://www.mc.com/products/view/index.html). In the meanwhile also some programming languages have appeared, namely Cg (from Nvidia and Microsoft), Brooke for GPUs, and Sh, making the implementation of individual programs on commodity GPUs easily accessible (Fernando and Kilgard, 2003), http://graphics.stanford.edu/papers/brookgpu/, (McCool and Du Toit, 2004). Due to good documentation and similarity to C/C++, Cg is easily applicable. However, Cg is different from C/C++ because it is very specialized. This type of language is called a shading language, which can be used on the new programmable GPUs to implement various non shading tasks as e.g. a Fourier transformation or a reconstruction technique. Here, we report on the implementation of backprojection methods, simultaneous iterative reconstruction techniques (SIRT) and simultaneous algebraic reconstruction techniques (SART) on the GPU (Kak and Slaney, 1988) (Carazo et al., 1999).

The quality achieved by these reconstruction techniques and more importantly the speed-gain compared to the CPU is presented. Detailed flowchart and pseudo-code diagrams present the implementation tailored for the needs of electron microscopy on the GPU. Limitations of the implementation such as the instruction limit present on the GPUs, and the reasons why the algebraic reconstruction technique (ART) can not be implemented efficiently on the GPU are also discussed in particular. The paper is organized as follows: In Section 2, the basic structure of the algorithms is discussed. In Section 3, the iterative algorithms are presented and in Section 4 the implementation of the algorithms on the GPU is explained. Finally in Section 5 the results in terms of quality as well as performance are compared to the CPU

2. Structure of the implemented algorithms

Before we discuss the implementation strategy of the reconstruction techniques, it is worthwhile examining which algorithms are well suited for the GPU. In the following, we assume that each image used for the reconstruction is aligned to a common origin by an arbitrary automatic or interactive technique (Frank, 1992). The reconstruction algorithms used in electron microscopy can be classified in two groups: (a) the weighted backprojection methods (WBP), with a weighting with an appropriate function, e.g. ramp filter, exact weighting function etc. and the backprojection of the image (Radermacher, 1992), and (b) the algebraic iterative techniques, e.g. algebraic reconstruction technique (ART), simultaneous iterative reconstruction technique (SIRT), and simultaneous algebraic reconstruction technique (SART) (Kak and Slaney, 1988) (Sorzano et al., 2001). Even though weighted-backprojection methods appear to be more common for reconstructing 3D images, maybe due to their algorithmic simplicity and computational speed, it is worth to notice that iterative methods could have the potential to outperform backprojection methods; fundamentally due to their flexibility to enrich the model with physical information just by adding side conditions into the mathematical problem (Skoglund et al., 1996). However, their large demand for computation time has prevented the exploration of these capabilities. In this view, the possibility of performing reconstructions by iterative methods in acceptable computing time becomes especially relevant.

In the following, we concentrate on the iterative methods, because from the algorithmic point of view the backprojection step required for a reconstruction by filtered