18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

High-performance tomographic reconstruction using graphics processing units

Ya.I. Nesterets and T.E. Gureyev

CSIRO Materials Science and Engineering, Victoria, Australia
Email: yakov.nesterets@csiro.au

Abstract: Computed Tomography (CT) has become a routine tool for three-dimensional (3D) visualization of the internal structure of objects which are opaque to visible light. X-ray CT is widely used in materials science, biomedical applications and elsewhere. Different CT reconstruction algorithms have been developed including the well-known Filtered-Back-Projection (FBP) algorithm in the parallel beam geometry and the Feldkamp-Davis-Kress (FDK) algorithm applicable to cone-beam geometry with a circular trajectory of a small X-ray source and a flat two-dimensional (2D) detector. Recent progress in detector technology resulted in the availability of commercial 2D Charge-Coupled-Devices (CCDs) with linear dimensions of the order of 4k pixels or more. The amount of memory required for storing a 3D volume of floating-point data of such linear size is 256GB or more, which significantly exceeds the typical amount of RAM found not only in high-end desktop computers but also in small computer clusters.

The most computationally intensive step in the FBP and FDK CT reconstruction algorithms is the so-called back-projection operation that takes up to 99% of the total reconstruction time in a typical CPU-based implementation. The 3D computer graphics capabilities of general-purpose graphics processing units (GP-GPUs) utilized e.g. via OpenGL or DirectX have been used for the back-projection operation for the last ten to fifteen years. With the recent increase in the size of reconstructed volumes as mentioned above, standard approaches which usually store the whole reconstructed volume in GPU memory have become problematic. Different algorithmic approaches are required for the effective use of GPUs for CT reconstruction of large data volumes.

We have developed new CPU-based and GPU-based implementations of the FBP and FDK algorithms for X-ray CT. These implementations take into account the following principles:

- Use as little RAM and/or GPU memory for the reconstruction of each axial slice of the object as possible. E.g., for the GPU-based back-projection code, memory for only a single reconstructed slice is allocated on the GPU; this allows the reconstruction of volumes with linear dimension of up to 16k using top-end GPUs (with more than 1GB memory onboard);
- Reconstruction of each axial slice should be as independent from the others as possible. This allows for parallel reconstruction of different slices using CPU/GPU multithreading capabilities. As a result, the total reconstruction time reduces with the number of CPU cores and/or GPUs.

According to our tests, the GPU-based implementations of the back-projection operation result in up to two orders of magnitude speed-up of the back-projection itself and more than an order of magnitude speed-up of the total CT reconstruction compared to the corresponding results for a single CPU core.

Keywords: *Computed Tomography (CT), High-Performance Computing (HPC), Graphics Processing Unit (GPU)*

1. INTRODUCTION

Computed Tomography (CT) has been a valuable tool for non-destructive investigation of the internal structure of objects since the pioneering work of R.N. Bracewell, A. Cormack and G. Hounsfield in the second half of the 20th century (Herman, 1980). From the very beginning of its applications to problems in astronomy, medicine and material sciences, the use of CT was constrained to a large extent by the processing power of computers available at the time. More recently, several research groups and commercial companies have implemented CT reconstructions using Field-Programmable Gate Arrays (FPGAs) and Cell Broadband Engine (Cell BE) (Kachelrieß et al., 2007). However, most of the specialized hardware implementations suffered from the lack of an easily accessible development platform which could be used by programmers skilled in general-purpose languages such as C or Fortran. The situation changed dramatically a few years ago when NVidia released its C-like language, CUDA, suitable for programming its latest GPUs. This created an explosion in scientific and technological applications of GPU programming, which included simplified codes for CT reconstruction on GPUs, see e.g. (Xu and Mueller, 2007). Examples of CUDA-based applications can be found at the official NVidia website, http://www.nvidia.com/object/cuda_home.html.

Our group at Commonwealth Scientific and Industrial Research Organisation (CSIRO) has been developing software for X-ray image processing and analysis, including CT reconstruction, for more than 10 years. Our main software called X-TRACT (http://www.ts-imaging.net/Services/AppInfo/X-TRACT.aspx) is capable of performing a large number of different processing operations on X-ray images, including CT reconstruction and simulation in conventional and phase-contrast modes. In the last year we have been developing and testing different implementations of conventional CT reconstruction algorithms (FBP and FDK) on high-end NVidia GPUs. In this paper we report the results of the tests of our CT reconstruction routines running on these GPUs and compare them with the performance of similar codes running on CPUs. The results appear to be very encouraging and seem to confirm that the appearance of the CUDA platform has changed the field of scientific programming and made the use of GPUs for scientific computations a very attractive option.

2. PARALLEL-BEAM AND CONE-BEAM COMPUTED TOMOGRAPHY

Schematic representation of the imaging geometry for the parallel-beam and cone-beam tomography is shown in Figure 1. Here S designates an X-ray source (in the case of the parallel-beam tomography the source is located at a large distance from the object or the incident beam is collimated by some additional device), O is an object and D is a flat-panel detector (for example a CCD camera).

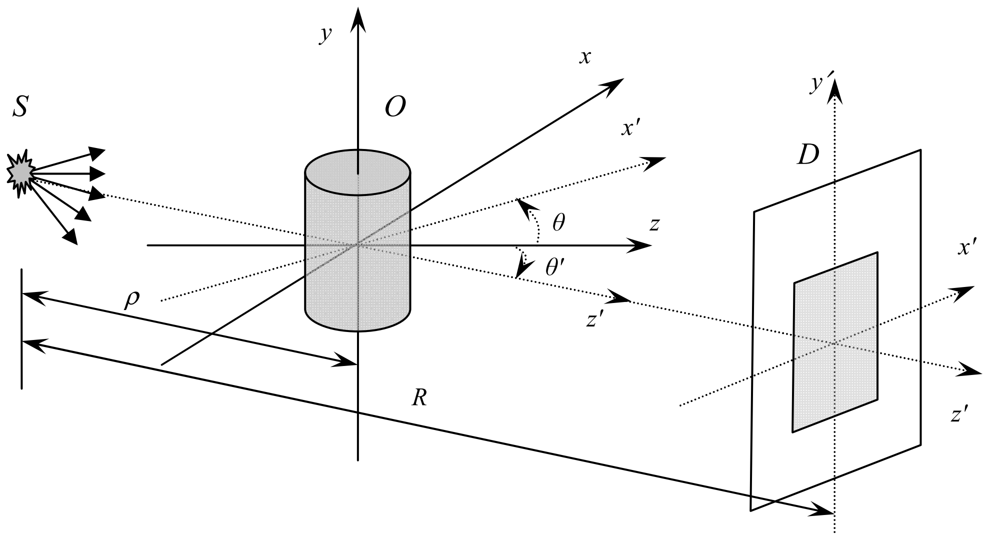

Figure 1. Schematic diagram of the imaging system used for computed tomography.

Let the object be described by a 3D distribution of some physical parameter, $f(x, y, z)$. In parallel-beam geometry, projection of the object, $(\mathbf{P}_\theta f)(x', y')$, at an angular position θ, is mathematically expressed by the following linear integral,

$$(\mathbf{P}_\theta f)(x', y') = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y', z)\,\delta(x' - x\sin\theta - z\cos\theta)\,\mathrm{d}x\mathrm{d}z\,. \quad (1)$$

If projections are collected at multiple angular positions of the object in the interval $\theta \in [0, \pi)$, then the 3D distribution of the object function f can be calculated using the well-known filtered back-projection (FBP) reconstruction algorithm (Herman, 1980; Natterer, 1986),

$$f(x,y,z) = \int_0^\pi \left(\mathbf{F}_1^{-1}\left[\,|\xi'|\,\mathbf{F}_1(\mathbf{P}_\theta f)(x',y')\right]\right)\Big|_{\substack{x'=x\sin\theta+z\cos\theta\\ y'=y}} d\theta\,, \tag{2}$$

where $\mathbf{F}_1$ is the one-dimensional (1D) Fourier transform with ξ' dual to x'.

In the cone-beam geometry, projection of the object in the detector plane (x', y') can be presented as follows,

$$(\mathbf{P}_\theta f)(x',y') = \int_{-\infty}^{\infty} f[\mathbf{s}(\theta) + \mathbf{p}(\theta)\,t\,/\,|\,\mathbf{p}(\theta)\,|]\,dt\,, \tag{3}$$

where $\mathbf{s}(\theta) = (\rho\cos\theta, 0, -\rho\sin\theta)$ describes position of a point source with respect to the object, and $\mathbf{p} = (x'\sin\theta - R\cos\theta, y', x'\cos\theta + R\sin\theta)$ is a vector connecting the source and a point (x', y') in the detector plane. Given knowledge of the object projections at angular positions in the interval $\theta \in [0, 2\pi)$, we can reconstruct the object function f using the well-known Feldkamp-Davis-Kress (FDK) reconstruction algorithm (Feldkamp *et al.*, 1984; Yan and Leahy, 1991),

$$f(x,y,z) = \frac{R^2\rho}{2}\int_0^{2\pi}\frac{1}{p_3^2}\left[\mathbf{F}_1^{-1}\left(|\xi'|\,\mathbf{F}_1\left[\frac{(\mathbf{P}_\theta f)(x',y')}{\sqrt{R^2 + x'^2 + y'^2}}\right]\right)\right]_{\substack{x'=Rp_1/p_3\\ y'=Rp_2/p_3}} d\theta\,, \tag{4}$$

where $(p_1, p_2, p_3) = (x\sin\theta + z\cos\theta, y, \rho - x\cos\theta + z\sin\theta)$ is a Cartesian coordinate system rotated by an angle $\theta' = \pi/2 - \theta$ about the y axis with respect to z, that is centered on the X-ray source.

From the algorithmic point of view, FBP reconstruction can be divided into two main steps:

- 1D ramp filtering of the object's projections in the Fourier space, which is expressed in Eq.(2) by the direct Fourier transform of the projections, subsequent multiplication by the ramp filter and inverse 1D Fourier transform;
- back-projection, integral over the rotation angle θ in Eq.(2).

In the case of the FDK reconstruction, additional operations are introduced, including pre-weighting of the projections preformed prior to ramp filtering and post-weighting performed during the back-projection step.

It is the back-projection step of the CT reconstruction that takes most of the reconstruction time (see Section 3 below). Fortunately, back-projection can be efficiently parallelized due to the fact that this step of CT reconstruction is essentially independent for different voxels in the object (although needs different input data) and can be done in parallel. In our trivial CPU-based implementation of the above-mentioned CT reconstruction algorithms, different axial slices are reconstructed from the sinograms by different CPU cores (using multi-threading capabilities of the CPUs and job distribution capabilities available in computer clusters). In contrast, our GPU-based implementations of the FBP and FDK reconstruction algorithms perform parallelized back-projection of each axial slice, using NVidia's CUDA programming tools (http://www.nvidia.com/object/cuda_get.html) for creating GPU-oriented execution code (kernels) for the back-projection operations. Each thread of the back-projection kernel calculates the object property in a single pixel in the reconstructed slice using the corresponding sinogram(s). In both implementations, filtering (weighted filtering in FDK) of the sinograms is done using CPU.

3. PERFORMANCE OF THE CT RECONSTRUCTION ALGORITHMS

We carried out comparative analysis of the performance of the CPU-based and GPU-based implementations of the FBP and FDK CT reconstruction algorithms. For the results presented in this section, CT reconstructions have been carried out in a single-thread execution mode, using a high-performance Dell Precision T7400 workstation running the Microsoft Windows XP x64 operation system and equipped with a quad-core Xeon E5420 processor (2.5 GHz), 1333 MHz front-side bus and 16 GB of RAM (666 MHz). Also, for the GPU-based calculations we used a GeForce GTX260 GPU (with 192 processor cores and 896 MB memory onboard) connected via PCI-E 2.0 $\times$16 bus.

In our implementation, CT reconstruction (either FBP or FDK) consists of the following four main steps:

- reading a single sinogram file (multiple sinograms, in the case of the FDK) from hard drive;
- ramp-filtering the sinogram (sinograms) in the Fourier space using fast Fourier transform;
- back-projection, resulting in reconstruction of a single axial slice of the object;
- writing the resultant axial slice to hard drive as a single file.

Total reconstruction times together with the duration times for each reconstruction step are summarized in Tables 1-4. Table 1 and Table 2 correspond to the CPU-based FBP and FDK reconstructions respectively, while Table 3 and Table 4 present results for the corresponding GPU-based calculations. The 512×512×512 volume has been reconstructed from 180 (360) projections in the parallel-beam (cone-beam) case, with 512×512 pixels in each projection. Reconstruction of a volume with twice the linear size needed a twice the number of projections, with a twice the linear size of the projections.

Analysis of Table 1 and Table 2 shows that the back-projection step takes most of the time in the CPU-based CT reconstruction. The fraction of the back-projection step to the total reconstruction time increases with the volume size while the corresponding fractions of the read/write and filtering operations reduce with the volume size. Such behavior is explained by the corresponding complexities of the above operations. Designating N the linear size of the reconstructed volume, the complexities of the back-projection, read/write and filtering operations are expected to be $O(N^4)$, $O(N^3)$ and $O(N^3\log N)$, respectively.

Table 1. CPU-based (quad-core Xeon E5420 @ 2.5GHz) FBP CT reconstruction times performed on a local machine (Dell Precision T7400).

Reconstructed volume	Total time, s	Back-projection time, s	Sinograms reading time, s	Slices writing time, s	Total read/write time, s	Other operations time, s
512×512×512	369.6	352.7 (95.4%)	9.6 (2.6%)	3.2 (0.9%)	12.8 (3.5%)	4.1 (1.1%)
1024×1024×1024	7,220.2	7,123.1 (98.7%)	41.5 (0.6%)	22.8 (0.3%)	64.3 (0.9%)	32.8 (0.4%)

Table 2. CPU-based (quad-core Xeon E5420 @ 2.5GHz) FDK CT reconstruction times performed on a local machine (Dell Precision T7400).

Reconstructed volume	Total time, s	Back-projection time, s	Sinograms reading time, s	Slices writing time, s	Total read/write time, s	Other operations time, s
512×512×512	1,936.8	1,907.7 (98.5%)	13.3 (0.7%)	3.1 (0.2%)	16.5 (0.9%)	12.6 (0.6%)
1024×1024×1024	32,762	32,564 (99.4%)	61.5 (0.19%)	22.5 (0.07%)	84.0 (0.26%)	113.5 (0.34%)

Table 3. GPU-based (NVidia GeForce GTX260) FBP CT reconstruction times performed on a local machine (Dell Precision T7400).

Reconstructed volume	Total time, s	Back-projection time, s	Sinograms reading time, s	Slices writing time, s	Total read/write time, s	Other operations time, s
512×512×512	35.1 (10.5×)	10.0 (28.5%) (35.3×)	16.2 (46.3%)	4.6 (13.1%)	20.8 (59.4%)	4.3 (12.1%)
1024×1024×1024	207.1 (34.9×)	84.6 (40.9%) (84.2×)	57.4 (27.7%)	30.9 (14.9%)	88.3 (42.6%)	34.2 (16.5%)
2048×2048×2048	2,732.3	1,286.4 (47.1%)	258.2 (9.4%)	617.1 (22.6%)	875.3 (32.0%)	570.6 (20.9%)

Table 4. GPU-based (NVidia GeForce GTX260) FDK CT reconstruction times performed on a local machine (Dell Precision T7400).

Reconstructed volume	Total time, s	Back-projection time, s	Sinograms reading time, s	Slices writing time, s	Total read/write time, s	Other operations time, s
512×512×512	52.8 (36.7×)	16.0 (30.3%) (119.2×)	19.6 (37.2%)	4.7 (8.9%)	24.3 (46.1%)	12.4 (23.6%)
1024×1024×1024	372.3 (88.0×)	179.2 (48.1%) (181.7×)	56.4 (15.2%)	23.9 (6.4%)	80.3 (21.6%)	112.8 (30.3%)
2048×2048×2048	4,831.0	2,543.8 (52.7%)	487.8 (10.1%)	604.5 (12.5%)	1,092.2 (22.6%)	1,195.0 (24.7%)

One can notice, by analyzing Table 1 and Table 2, that reconstruction of a relatively small volume, 1024×1024×1024, takes hours when carried out using our CPU-based algorithms. For comparison, Table 3 and Table 4 present reconstruction times for our GPU-based implementations of the FBP and FDK algorithms, using the same input data as above in the CPU-based tests. Producing virtually identical results, the back-projection speed-up is in the range from several tens to more than a hundred times, compared to the corresponding CPU-based calculations. This results in the ten to ninety times speed-up of the total CT reconstruction. As an example, FDK reconstruction of the 1024×1024×1024 volume, that took more than 9 hours on a single CPU core, takes only slightly more than 6 minutes on the same CPU when carried out using GPU for back-projection, giving an 88-fold speed-up.

4. SCALABILITY OF THE CT RECONSTRUCTION

There are at least two aspects of scalability of the CT reconstruction algorithms presented in this paper. The first aspect is related to the ability of the algorithms to process large data sets. This ability is limited by the algorithmic demands in memory (for storing all necessary data) and by available RAM and/or GPU memory. From this point of view, our FBP reconstruction algorithm needs to store in host and GPU memory, at any time, of only two arrays of data, one array for the input sinogram and the other array for the reconstructed axial slice. As an example, an axial slice of size 4096×4096 (8192×8192) storing single-precision floating-point numbers, occupies 64MB (256MB) of memory. Thus FBP reconstruction of slices up to 16k×16k pixels can be in principle done even on desktop computers having 2GB of RAM. In contrast, our FDK reconstruction algorithm, at any time, stores in RAM several sinograms necessary for reconstruction of a given axial slice (a fraction of the total set of sinograms needed for reconstruction of a given axial slice strongly depends on the imaging geometry and slice number). Noticeably, for the GPU-based back-projection calculations, memory for only a single reconstructed slice is allocated on the GPU; this allows reconstruction of slices with linear dimension of up to 16k using top-end GPUs (with more than 1GB memory onboard).

Table 5. CPU-based (dual quad-core Xeon X5355 @ 2.66GHz) FBP CT reconstruction times performed on two cluster nodes (Dell PowerEdge 2900). 512×512×512 volume has been reconstructed from 180 projections, 512×512 pixels each.

No. of threads (N)	Total time (T), s	Speed-up
1	410	—
2	207	1.98×
4	110	3.73×
8	60	6.83×
16	35	11.71×

Table 6. CPU-based (dual quad-core Xeon X5355 @ 2.66GHz) FDK CT reconstruction times performed on two cluster nodes (Dell PowerEdge 2900). 512×512×512 volume has been reconstructed from 360 projections, 512×512 pixels each.

No. of threads (N)	Total time (T), s	Speed-up
1	1,916	—
2	964	1.99×
4	490	3.91×
8	250	7.66×
16	126	15.21×

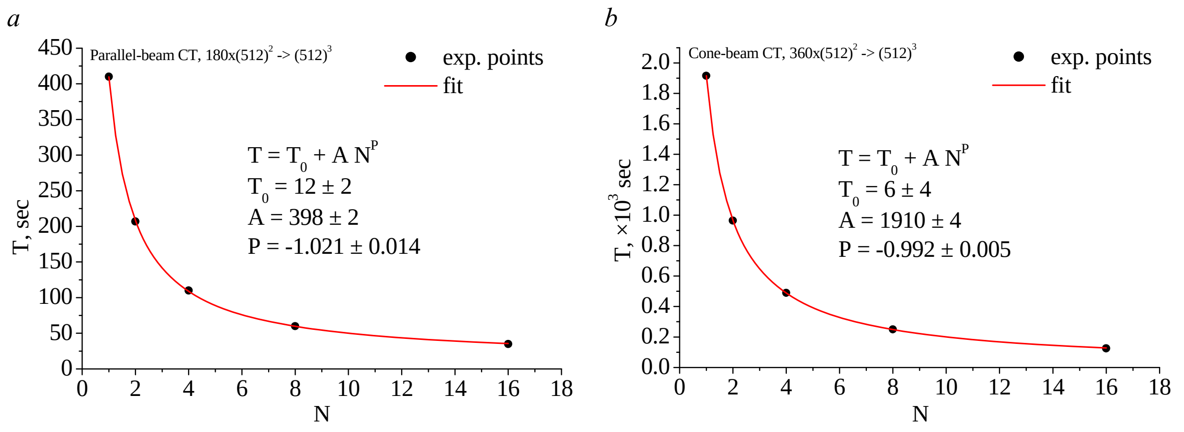

Figure 2. Total CT reconstruction time versus number of threads for the CPU-based implementations of the FBP (a) and FDK (b) algorithms carried out at a computer cluster. Dots - experimental points, solid line – curve fit.

The second aspect of scalability is related to the ability of our implementations of the CT reconstruction algorithms to be executed in parallel by many CPUs/GPUs using, for example, a computer cluster. This is a topical problem, as a single GPU (and all the more, a single CPU) cannot do CT reconstruction of large volumes (4096^3 and larger) in a short time. Indeed, according to Table 3 and Table 4, FBP reconstruction of 2048^3 pixels volume takes about 46 minutes, and the corresponding FDK reconstruction of the same volume takes about 81 minutes. Larger volumes, 4096^3 voxels and above, would require many hours/days of calculations using a single GPU/CPU. As we have already mentioned, our implementations of the FBP and

FDK reconstruction algorithms carry out volume reconstruction in a slice-by-slice manner, where reconstruction of each slice is independent from the others (although needs different input data). This calculation model was chosen intentionally, targeting its utilization with a computer cluster.

We carried out CT reconstructions on a small computer cluster using the same data as in the previous section. There are four nodes on this cluster, each running the Microsoft Cluster Server 2003 ×64 operation system. Three nodes of the cluster are Dell PowerEdge 2900 servers. For the CPU-based calculations we used two identical nodes, each equipped with dual quad-core Xeon X5355 processors (2.66 GHz), 1333 MHz front-side bus and 48 GB of RAM (666 MHz). For the GPU-based calculations we used the other node, a Xenon Nitro Z4 workstation equipped with dual quad-core Xeon X5472 processors (3.00 GHz), 1600 MHz front-side bus, 36 GB of RAM (800 MHz) and two GeForce GTX280 GPUs (each with 240 processor cores and 1 GB of memory) connected via PCI-E 2.0 ×16 bus. In both cases, input data and output data have been stored on the fourth node of the cluster used as a file server. The nodes in the cluster are connected by 1 Gigabit/s Ethernet.

Table 5 and Table 6 show total CPU-based CT reconstruction times for different numbers of running threads. The maximum number of simultaneously running threads was limited by the number of cores on the two nodes used for the calculations. Maximum speed-up achieved in the FBP (FDK) reconstruction was 11.7× (15.2×), as compared to the corresponding single thread execution, and reached 35 s (126 s) for the 512^3 voxels object. Figure 2 shows the measured reconstruction times versus number of threads (dots) together with the results of their fitting using a function of the following form, $T(N) = T_0 + AN^P$. Fitting parameters are shown in Figure 2. This form of the fitting function implies that there could be at least two components in the reconstruction time, one being independent of the number of threads running (first term in the function $T(N)$), and the other decreasing with the number of threads (second term in the function $T(N)$). Extrapolating the curves to larger values of N, one can notice that the ultimate reconstruction time cannot be less than T_0. Existence of this term can be explained by the finite internode data transfer and hard drives access rates.

Similarly, Tables 7-10 present GPU-based CT reconstruction times versus a number of threads. Two volumes have been reconstructed using GPUs. One might notice that cluster-based reconstruction times for a single thread significantly exceed the corresponding times of reconstructions carried out at a local workstation (Tables 3 and 4). This is a result of relatively low data transfer rate between the compute nodes and the file-server node. This connection latency is effectively masked in the multi-threaded regime; the minimum reconstruction times (see results for 8 threads) become noticeably smaller compared to those obtained in local calculations. Figure 3 presents results of fitting the experimental points contained in Tables 7-10 using function $T(N)$ defined above.

Table 7. GPU-based (dual GeForce GTX280) FBP CT reconstruction times performed on a single cluster node (Xenon Nitro Z4). 512×512×512 volume has been reconstructed from 180 projections, 512×512 pixels each.

No. of threads (N)	Total time (T), s	Speed-up
1	90	—
2	50	1.8×
4	30	3.0×
8	25	3.6×

Table 8. GPU-based (dual GeForce GTX280) FDK CT reconstruction times performed on a single cluster node (Xenon Nitro Z4). 512×512×512 volume has been reconstructed from 360 projections, 512×512 pixels each.

No. of threads (N)	Total time (T), s	Speed-up
1	120	—
2	65	1.85×
4	40	3.00×
8	30	4.00×

Table 9. GPU-based (dual GeForce GTX280) FBP CT reconstruction times performed on a single cluster node (Xenon Nitro Z4). 1024×1024×1024 volume has been reconstructed from 360 projections, 1024×1024 pixels each.

No. of threads (N)	Total time (T), s	Speed-up
1	492	—
2	265	1.86×
4	165	2.98×
8	111	4.43×

Table 10. GPU-based (dual GeForce GTX280) FDK CT reconstruction times performed on a single cluster node (Xenon Nitro Z4). 1024×1024×1024 volume has been reconstructed from 720 projections, 1024×1024 pixels each.

No. of threads (N)	Total time (T), s	Speed-up
1	745	—
2	400	1.86×
4	236	3.16×
8	149	5.00×

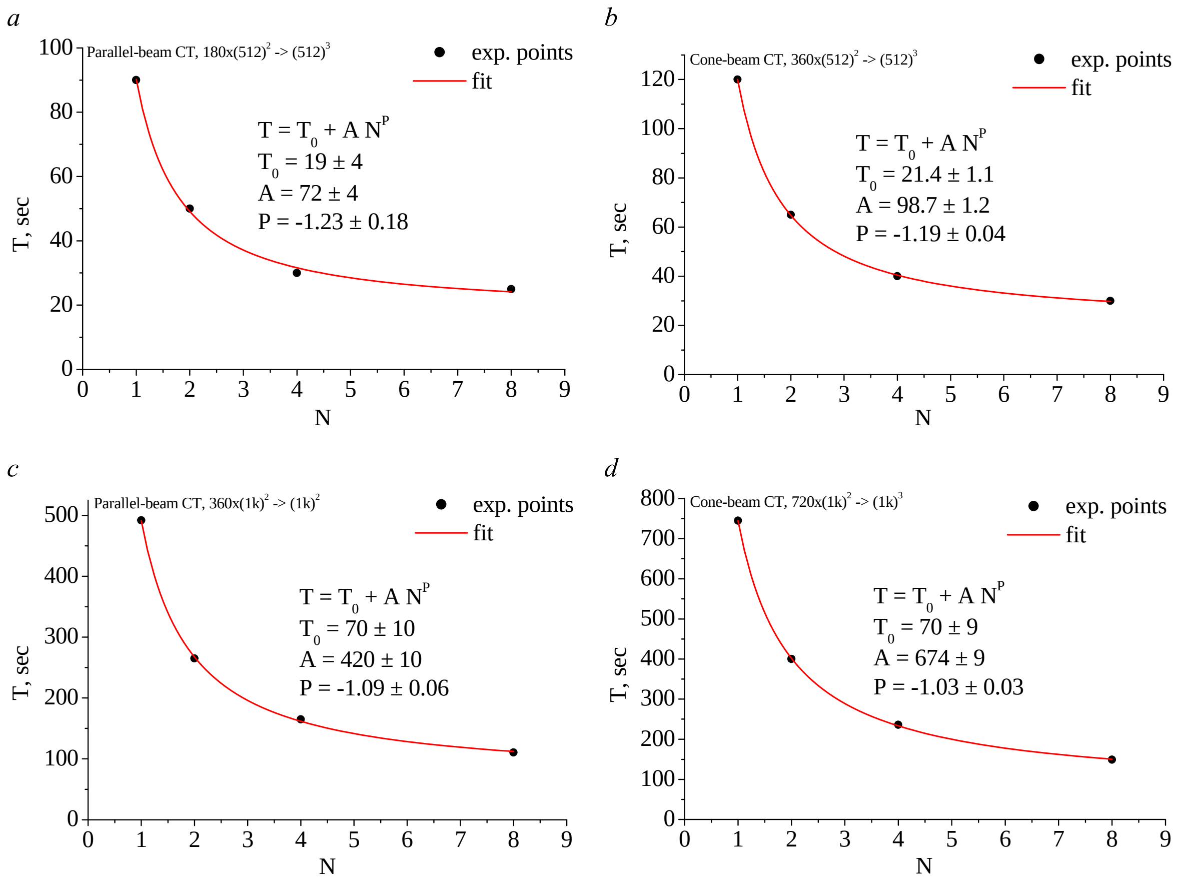

Figure 3. Total CT reconstruction time versus number of threads for the GPU-based implementation of the FBP (a) and FDK (b) algorithms carried out at a computer cluster. Dots - experimental points, solid line – curve fit.

5. CONCLUSIONS

We have shown that after implementing the back-projection operation (this is the most time consuming step in the FBP and FDK CT reconstruction algorithms) for execution on a GPU, the overall performance of a single-threaded reconstruction increases by several tens (up to one hundred) times compared to a comparable CPU-based implementation. This can be sufficient for reconstructing relatively small object volumes in a very short time. For larger volumes, further speed-up of the CT reconstruction is only possible if the job is distributed between many CPU cores (hundreds or even thousands) and/or GPUs (tens or even hundreds). In this case, data transfer between different computer cluster nodes can become a potential bottleneck and can only be overcome by effective use of the existing network capabilities and/or by utilizing higher-speed interconnects.

REFERENCES

Feldkamp, L.A., Davis, L.C., and Kress, J.W., (1984), Practical cone-beam algorithm. J. Opt. Soc.Am. A, 1, 612-619.

Herman, G.T., (1980), Image Reconstruction from Projections. The Fundamentals of Computerized Tomography. Academic Press, New-York.

Kachelrieß, M., Knaup, M., and Bockenbach, O., (2007), Hyperfast parallel-beam and cone-beam back-projection using the cell general purpose hardware. Med. Phys., 34(4), 1474-1486.

Natterer, F., (1986), The Mathematics of Computerized Tomography. Wiley, New York.

Xu, F., and Mueller, K., (2007), Real-time 3D computed tomographic reconstruction using commodity graphics hardware, Phys. Med. Biol., 52, 3405-3419.

Yan, X., and Leahy, R.M., (1991), Derivation and analysis of a filtered backprojection algorithm for cone beam projection data. IEEE Trans. Med. Im., 10, 462-472.