19th International Congress on Modelling and Simulation, Perth, Australia, 12–16 December 2011
http://mssanz.org.au/modsim2011

Rapid CT reconstruction on GPU-enabled HPC clusters

D. Thompson [a], Ya. I. Nesterets [a], T.E. Gureyev [a], A. Sakellariou [a], A. Khassapov [a], and J.A. Taylor [a]

[a] *Commonwealth Scientific and Industrial Research Organisation (CSIRO), Private Bag 33, Clayton South MDC, VIC 3169, Australia*
Email: darren.thompson@csiro.au

Abstract: Computed Tomography (CT) reconstruction is a computationally and data-intensive process applied across many fields of scientific endeavor, including medical and materials science, as a non-invasive imaging technique. A typical CT dataset obtained with a CCD-based X-ray detector, such as that at the Australian Synchrotron with 4K×4K pixels captured over multiple-view angles, is in the order of 128GB. The reconstructed output volume is in the order 256GB. CT data sizes increase at 1.5 times the number of pixels in the detector, while the data-processing load generally increases as the square of the number of pixels, hence data storage, management and throughput capabilities become paramount. From a computational perspective, CT reconstruction is particularly well suited to mass parallelisation whereby the problem can be decomposed into many smaller independent parts. We have achieved significant performance gains by adapting our XLI software algorithms to a two-level parallelisation scheme, utilising multiple CPU cores and multiple GPUs on a single machine. In turn, where data sizes become prohibitively large to be processed on a single machine, we have developed an integrated CT reconstruction software system that is able to scale up and be deployed onto large GPU-enabled HPC clusters. We present here the results of reconstructing large CT datasets using our XLI software on both the CSIRO GPU cluster and the new MASSIVE-1 cluster located at the Australian Synchrotron. Both of these clusters provide high-end compute nodes with multiple GPUs coupled by high-speed interconnect and IO capabilities which combine to allow rapid CT reconstruction. Provided in this paper are examples of the application of the developed tools to the reconstruction of large CT datasets collected both at synchrotrons and with laboratory-based CT scanners.

Keywords: *CT reconstruction, GPU, high performance computing, clusters, parallel programming*

1. INTRODUCTION

Computed Tomography (CT) reconstruction has become a common non-invasive tool for the visualisation of the internal structure of objects which are opaque to visible light. X-ray CT is widely used across a diverse range of disciplines, from medicine to materials science. It is a computationally and data-intensive process requiring significant computational resources. Our group within the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia, has for over ten years been developing our XLI (http://www-ts.imaging.net/Services/AppInfo/X-TRACT.aspx) application for X-ray image analysis, processing and simulation of which CT reconstruction is a major function.

CT reconstruction belongs to the class of computational problems which are often referred to as "embarrassingly parallel" and described by Forster (1995). Essentially, such problems can be decomposed into smaller independent units and computed in parallel. In recent years, several research groups and commercial enterprises have exploited this property to develop rapid CT reconstruction using specialised parallelised hardware such as Field-Programmable Gate Arrays (FPGAs) and Cell Broadband Engines (Cell BE), see Leeser, et al. (2005), Kachelrieß, et al. (2006) and Scherl, et al. (2007). However, such implementations can be limited due to the expense of the hardware and the complexity in developing software. In this period, Graphics Processing Units (GPUs) have become widely popular in both mass-market and High Performance Computing (HPC) systems. Originally, GPUs were designed to perform high-speed graphics rendering for computer gaming via their massively parallel, multi-core architectures, combined with optimised caches and memory. However, their usage has been subsequently expanded and adapted to general computational problems with the release of development APIs such as NVidia's CUDA (http://www.nvidia.com/object/cuda_home_new.html) and OpenCL from Khronos Group (http://www.khronos.org/). The availability of large numbers of processing cores on a single consumer-level machine equipped with a GPU has enabled particular classes of algorithms including CT reconstruction to be mass-parallelized. This has resulted in speed-ups in the range of an order of magnitude or more, previously only achievable on customised hardware or HPC systems. Many groups have taken advantage of these relatively inexpensive devices, leveraging general purpose development platforms and tools to develop their own GPU-based CT reconstruction implementations, eg. Xu & Mueller (2007), Sharp, et al. (2007) and Hintermüller, et al. (2010).

Even with the performance gains achieved in CT reconstruction with the use of GPUs, datasets generated by synchrotrons and laboratory-based CT scanners are generally beyond the memory and IO capabilities of most single machine systems to process in a reasonable time. Currently, such datasets can be of the order of hundreds of gigabytes or terabytes and are rapidly increasing in size with detector advances and upgrades. In order to rapidly perform CT reconstructions on such data volumes we have developed our XLI software to be deployable on GPU-enabled HPC clusters. In the HPC cluster environment we have at our disposal many nodes, each providing multiple cores, GPUs and many gigabytes of RAM, coupled with high-speed storage and interconnect. This permits us to once again use the embarrassingly parallel property to introduce a second level of parallelisation such that the CT reconstruction task is first split and distributed across the available cores on the cluster nodes prior to the second level of high-speed GPU parallelisation. Such a scheme is scalable for increasing data volumes, given RAM, IO and interconnect specifications are balanced to meet demand.

In this paper we present the results and analysis of XLI's GPU-based FB[CT reconstruction algorithm on two large HPC clusters, namely the 128-node CSIRO GPU cluster (http://www.csiro.au/resources/GPU-cluster.html) located in Canberra, Australia and at the MASSIVE cluster (http://www.massive.org.au), a 48-node GPU-enabled HPC cluster located at the Australian Synchrotron (http://www.synchrotron.org.au) in Melbourne, Australia.

2. PARALLEL-BEAM COMPUTED TOMOGRAPHY

In order to understand the possible ways in which CT reconstruction can be implemented and parallelised, it is necessary to give at least a brief overview of the typical setups used for the acquisition of CT data. Schematic representation of the imaging geometry for the so-called parallel-beam tomography is shown in Figure 1. Here *S* designates an X-ray source which, in the case of the parallel-beam tomography, is located at a large distance from the object. Alternatively, the incident beam is collimated by some additional device. O is an object and D is a position-sensitive detector (for example, a CCD camera).

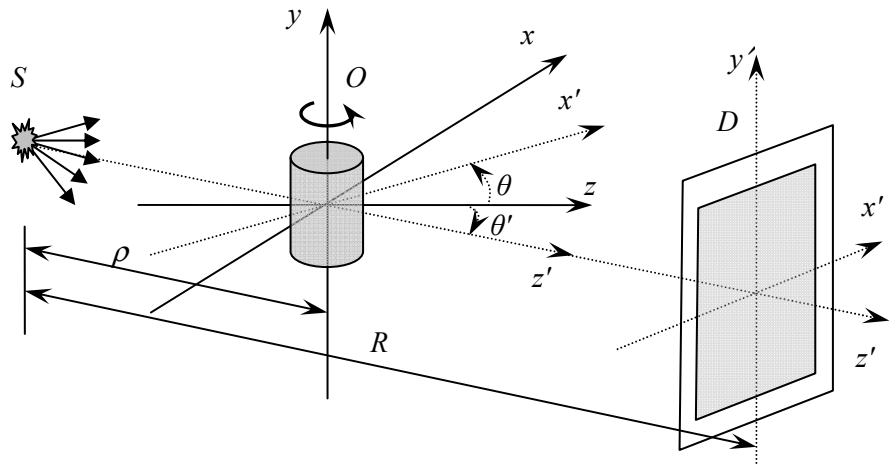

Figure 1. Schematic diagram of the imaging system used for computed tomography.

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 (X-ray transform),

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

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 backprojection reconstruction algorithm (FBP), Herman (1980), Natterer (1986),

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

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

From the algorithmic point of view, the 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 (2) by the direct Fourier transform of the projections, subsequent multiplication by $|\xi'|$ (the ramp filter) and the inverse 1D Fourier transform;
- backprojection, represented by the integral over the rotation angle θ in (2).

The axial slices of the reconstructed object are defined by the 2D sections of the 3D function $f(x, y, z)$ from (2) with fixed y equal to $y_m = m\Delta y$, where $m = 0,1,\ldots,M$, are integer slice indices and Δy is the vertical resolution of the CT system. These slices are contained in planes $y = const$ that are orthogonal to the axis y around which the object is rotated during the CT scan (see Figure 1). The reconstruction of each such slice can be performed independently from that of all other slices which implies a very simple parallelisation scheme for the complete 3D reconstruction. In other words, as mentioned in the Introduction, the conventional parallel-beam CT reconstruction problem, as defined by (2), is "embarrassingly parallel", i.e. it naturally splits into independent reconstruction tasks for each axial slice of the sample with no interaction required between these parallel tasks.

It is the backprojection step of the CT reconstruction algorithm (defined by the integration over the rotation angle in (2) after other operations have been completed) that takes most of the reconstruction time. Fortunately, the backprojection can be efficiently parallelised due to the fact that this operation is essentially independent for different voxels in the reconstructed object and can be carried out in parallel. **Error! Reference source not found.** shows a schematic representation of the "voxel-driven" backprojection. For each voxel of the reconstructed object (gray dot represents its centre), the contribution of each projection to this voxel is calculated by finding the corresponding projection of the voxel onto the

detector plane (gray dot) and estimating the projection value at this point using known projection values at the neighbouring grid points (e.g. using bi-linear interpolation).

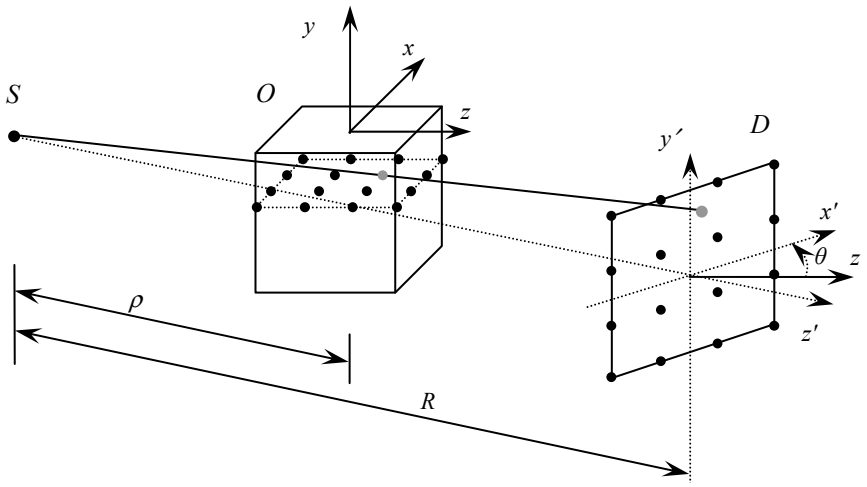

Figure 2. Schematic representation of the voxel-driven backprojection

Our GPU-based implementations of the FBP reconstruction algorithm further parallelise the backprojection task for each axial slice which represents the most computationally intensive part of the CT reconstruction. For this purpose we use NVidia's CUDA computing architecture (http://www.nvidia.com/object/cuda_home_new.html) to create GPU-oriented execution code (kernels) for the backprojection operations. Each computation thread of the backprojection kernel in the GPU calculates the value of the object function $f(x_l, y_m, z_n)$ in a single pixel (x_l, z_n) in the slice $y = y_m$ by evaluating the corresponding "outer" backprojection integral in (2). A CUDA kernel (a code whose execution is initiated by a CPU thread or process and which runs on a GPU device) implements the SIMD (Single Instruction Multiple Data) strategy: namely, the same code runs on all the processor cores of the GPU but is applied to different input data and produces different output data. Both the input and output data usually have the form of a 1D, 2D or 3D-array. Each individual run of the code on a single GPU core is called a device thread. For the backprojection operation, the total number of device threads involved in the reconstruction of a single axial slice is equal to (or sometimes larger than) the number of pixels in the slice. The device threads have a two-level hierarchy (a block of threads and a grid of blocks, respectively) which specifies the order of the threads execution and possibilities for data interchange between different threads. This thread hierarchy is directly related to the hardware design of the NVidia GPUs. For details on this, the reader is referred to the official CUDA Programming Guide, see NVidia (2011). We only mention that an NVidia GPU contains one or more multiprocessors (MPs) and a global memory accessible by all the MPs. Each MP, in its turn, contain multiple processor cores with associated registers, shared memory and two types of cached memories including constant memory and texture memory. Threads of a block (its size is currently limited to 512 threads) are executed on the same MP. That is, different MPs execute different blocks of threads. This implies certain segmentation of the input and output data corresponding to different blocks. For instance, in our implementation of the cone-beam backprojection, each block of GPU device threads reconstructs a square fragment (16×16 pixels) of the axial slice.

Thus, in addition to the higher-level parallelisation described above, we have implemented additional independent parallelisation of the most computationally demanding part of the CT reconstruction, the backprojection operation. This "lower-level" parallelisation allows us to take advantage of the many-core hardware architecture of the GPUs to achieve a much higher degree of parallelisation than is typically available with conventional CPUs. A relatively straightforward implementation of this lower-level parallelisation has been made possible by NVidia's CUDA programming language and tools which can be integrated with the popular C++ development tools and compilers.

It should be noted that our XLI application also contains a GPU-enabled implementation of the well-known Feldkamp-Davis-Kress (FDK) reconstruction algorithm for cone-beam CT, see Feldkamp et al. (1984) and Nesterets et al. (2009). However, we do not present any analysis of this algorithm in this paper.

3. PERFORMANCE AND SCALABITIY

We begin this section with an analysis of the performance of the GPU-based XLI implementation of the FBP CT reconstruction algorithm. CT reconstructions have been performed in a single-thread execution mode, using a high-performance Dell Precision T7400 workstation running the Microsoft Windows XP x64 operating 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). Attached was a GeForce GTX260 GPU (with 192 processor cores and 896 MB memory onboard) connected via PCI-E 2.0 ×16 bus.

The FBP CT reconstruction algorithm as implemented by XLI consists of the following four steps:

- Reading a single sinogram file from the hard drive.
- Ramp-filtering the sinogram in Fourier space using FFT on CPU.
- Backprojection on GPU, resulting in reconstruction of a single axial slice of the object.
- Writing the resultant axial slice to the hard drive as a single file.

Total reconstruction times together with the execution times for each reconstruction step are summarised in Table 1 below. The 512×512×512 volume has been reconstructed from 180 projections, with 512×512 pixels in each projection. Reconstruction of a volume with twice the linear size needed twice the number of projections, with twice the linear size of the projections.

Table 1. GPU-based (NVIDIA GeForce GTX260) FBP CT reconstruction times performed on a local machine (Dell Precision T7400)

Reconstructed volume	Total time, s	Backprojection time, s	Sinograms reading time, s	Slices writing time, s	Total read/write time, s	Other operations time, s
512×512×512	35.1	10.0 (28.5%)	16.2 (46.3%)	4.6 (13.1%)	20.8 (59.4%)	4.3 (12.1%)
1024×1024×1024	207.1	84.6 (40.9%)	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%)

Analysis of Table 1 shows that the backprojection step of our GPU-based CT reconstruction algorithm is roughly equivalent to the combined input and output IO steps where IO is performed on a local hard drive. With increasing reconstruction volumes it can be noted that the backprojection step quickly becomes dominant due the algorithmic complexity bounds of $O(N^4)$, $O(N^3)$ and $O(N^2 \log N)$ for the backprojection, IO and filtering operations respectively.

One quickly notices that the $O(N^4)$ bound for the backprojection step makes CT reconstruction relatively infeasible even on GPU-enabled machines in the single-threaded execution model as described above. Reconstruction volumes such as those generated by synchrotrons can be of the order of 4096^3 voxels and larger and would require many hours or days of computation time. Aside from the time constraint, the only other factors affecting scalability of the algorithm is that of RAM availability on the CPU and GPU. The FBP algorithm requires only enough RAM to store at any time only two 2D arrays of data in CPU and GPU memory, that of the input sinogram and reconstructed slice. For example, an axial slice of size 4096×4096 containing single-precision floating-point numbers, occupies 64MB of RAM. As such the FBP algorithm has a relatively small memory footprint which makes it theoretically possible to reconstruct slices of up to 16k×16k or fewer pixels with as little as 2GB of CPU/GPU RAM.

To be able to rapidly reconstruct large data volumes it is necessary to exploit the "embarrassingly parallel" property of the CT reconstruction algorithm as mentioned in Section 2. As reconstructions are performed slice-by-slice and are independent from one another, it allows us to efficiently distribute slices across a cluster for reconstruction in parallel amongst the total pool of available CPU cores and GPUs.

Moreover, cluster-based reconstruction imposes some additional scalability constraints to those mentioned above. In the cluster implementation, an XLI "worker" process is generally created for every CPU core across the desired number of nodes. As such a compute node must have enough RAM for N input sinograms and reconstructed slices, where N is the number of worker processes to execute on that node. The GPU memory constraint remains the same as before – our implementation requires worker processes to "share" attached GPUs whereby processes attain an exclusive lock on an available GPU for the purpose of the backprojection step of a slice, then relinquishing it upon completion for other processes. This scheme does lead to some latency in the algorithm with processes competing for GPU resources. However, for the tested datasets this latency is relatively insignificant compared to the IO overhead as described below.

A significant constraint to scalability in the cluster implementation is currently due to IO. Unlike the single machine model which performs all IO on a local hard disk, the cluster model requires the use of a centralised data store in which all nodes have access for reading and writing. Such a data store is usually provided by a Network Attached Storage (NAS) system. This leads to IO performance being constrained by interconnect performance and network load. Unlike the local model, the cluster model's IO component of the CT reconstruction algorithm is generally dominant for the reconstruction volumes we have used, resulting in performance of the algorithm being IO bound. Of course, due to the more significant performance bound on the backprojection step, increases in reconstruction volumes will quickly result in the algorithm becoming compute-bound once again. This is an important consideration in choosing cluster specifications such that compute and IO performance are ideally balanced for optimal results where practicable.

We carried out comparative performance tests of GPU-based FBP CT reconstruction on the CSIRO GPU cluster and the MASSIVE cluster. The CSIRO GPU cluster has 128 dual 4-core Xeon E5462 nodes, each with 32 GB of RAM and two NVidia Tesla S2050 GPUs. Data storage is provided by an 80 TB Hitatchi NAS file system and interconnect between the nodes and storage is DDR Inifiniband (4 Gbit/s). The MASSIVE cluster has 42 dual 6-core Xeon nodes, each with 48 GB of RAM and two NVidia Tesla M2070 GPUs. Data storage is a 58 TB IBM GPFS parallel file system and interconnect between nodes and storage is 4x QDR Gigabyte/s Infiniband (32 Gbits/s).

Both clusters are configured as dynamically provisioned dual-boot systems for simultaneous use of the Windows HPC 2008 R2 cluster operating system and Linux. XLI is currently designed for use with WinHPC only.

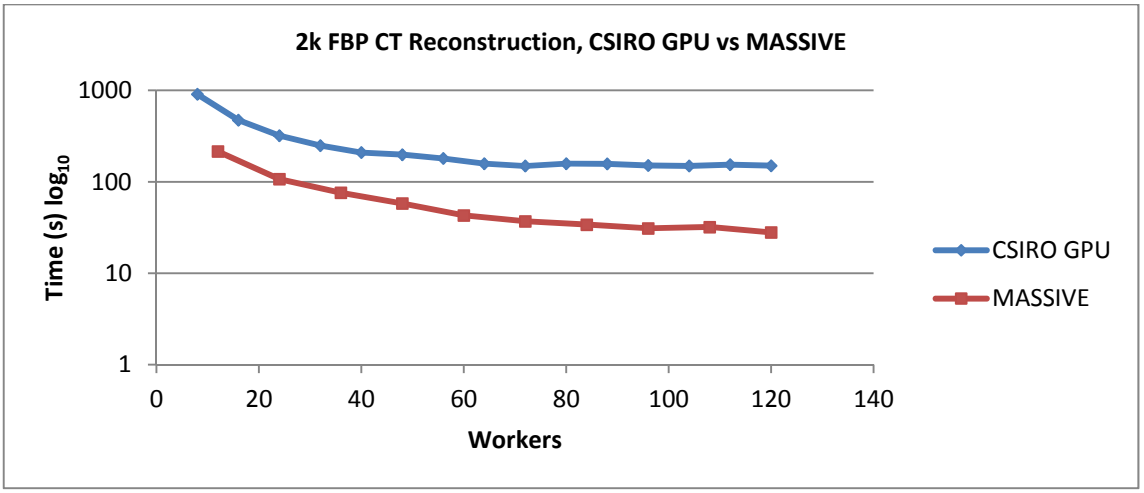

Figure 3. Total GPU-based, FBP CT reconstruction times between CSIRO GPU and MASSIVE cluster of a 2K reconstruction volume (720 input projections).

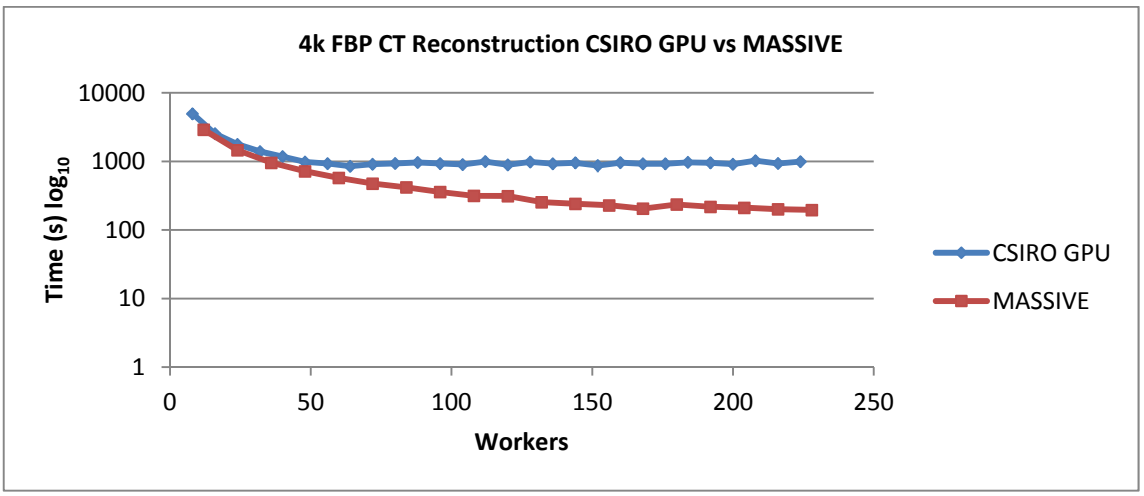

Figure 4. Total GPU-based, FBP CT reconstruction times between CSIRO GPU and MASSIVE cluster of a 4K reconstruction volume (1441 input projections).

Figure 3 and Figure 4 above show the total reconstruction times for identical $2k^3$ (720 input projections) and $4k^3$ (1441 input projections) volumes on both clusters for an increasing number of workers. Each point on the graph represents a single reconstruction job using the total number of CPU cores available for the desired number of nodes. In the case of the CSIRO GPU cluster, 8 workers per node are executed while on the MASSIVE cluster, 12 workers per node are created. The 2k volume was reconstructed from 720, 2048×2048 pixel input projections, 16 MB per projection, ~11GB total. Similarly the 4k volume was reconstructed from 1441, 4096×4096 pixel projections, 64MB per projection, ~90GB total.

Notably, the MASSIVE cluster is significantly quicker than the CSIRO GPU cluster across both datasets. We attribute this primarily to the substantially higher IO performance of the GPFS file system and, to a lesser degree, to the higher speed interconnect. From a computational perspective, the MASSIVE cluster would hold a slight advantage over the CSIRO GPU cluster due to more cores per node and more powerful GPUs, but these alone would not be enough to explain the displayed difference. Both clusters achieve near-linear scaling, however both also exhibit a leveling-off of reconstruction times at a finite number of workers. On the CSIRO GPU cluster this appears to be around 9 nodes (72 workers) and around 14 nodes (168 workers) on the MASSIVE cluster. We believe this corresponds to the saturation of network bandwidth to the attached storage which subsequently imposes a finite bound on the scalability of the algorithm.

4. CONCLUSIONS

We have demonstrated that rapid FBP CT reconstruction of large datasets is possible on a GPU-enabled HPC cluster leading to our goal of near-realtime reconstruction. Our algorithms exploit the embarrassingly parallel nature of CT reconstruction, allowing us to split the problem at a higher level into independent tasks for distribution and execution amongst available CPU cores on the cluster. A second, lower level of parallelisation is also attained using the multiple-cores of attached GPUs to significantly speed up the core backprojection step. We have found that near-linear scalability can be achieved subject to sufficient bandwidth existing between compute nodes and data storage.

ACKNOWLEDGMENTS

The authors wish to acknowledge the National eResearch Architecture Taskforce (NeAT) which has provided major funding towards this project.

REFERENCES

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

Forster, I. (1995). *Designing and Building Parallel Programs.* Addison Wesley.

Herman, G. (1980). *Image Reconstruction from Projections. The Fundementals of Computerized Tomography.* New York: Academic Press.

Hintermüller, C., Marone, F., Isenegger, A., & Stampanoni, M. (2010). Image processing pipeline for synchrotron-radiation-based tomographic microscopy. *J. Synchrotron Rad. , 17* (4), 550-559.

Kachelrieß, M., Knaup, M., & Bockenbach, O. (2006). Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware. *Medical Physics , 34* (4), 1474-1486.

Leeser, M., Coric, S., Miller, E., Yu, E., & Trepanier, M. (2005). Parallel-Beam Backprojection: An FPGA Implementation Optimized for Medical Imaging. *J. VLSI Signal Process. , 39* (3), 295-311.

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

Nesterets, Y. I., & Gureyev, T. E. (2009). High-performance tomographic reconstruction using graphics processing units. In R. S. Anderssen, R. D. Braddock, & L. Newham (Ed.), *18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation*, (pp. 1045-1051).

NVIDIA. (2011). *NVIDIA Programming Guide.* Retrieved from http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/docs/CUDA_C_Programming_Guide.pdf.

Scherl, H., Koerner, M., Hofmann, H., Eckert, W., Kowarschik, M., & Hornegger, J. (2007). Implementation of the FDK algorithm for cone-beam CT on the cell broadband engine architecture. *Proc. SPIE , 6510* (651058).

Sharp, G., Kandasamy, N., Singh, H., & Folkert, M. (2007). GPU-based streaming architectures for fast cone-beam CT image reconstruction and demons deformable registration. *Phys. Med. Biol. , 52* (19), 5771-5783.

Xu, F., & Mueller, K. (2007). Real-time 3D computed tomographic reconstruction using commodity graphics hardware. *Physics in Medicine and Biology , 52* (12), 3405-3419.