2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization

Scalable and Modular Online Data Processing for Ultrafast Computed Tomography Using CUDA Pipelines

Tobias Frust, Guido Juckeland and André Bieberle
Helmholtz-Zentrum Dresden-Rossendorf
Bautzner Landstr. 400, 01328 Dresden, Germany
Email: {t.frust, g.juckeland, a.bieberle}@hzdr.de

***Abstract*—For investigations of rapidly moving structures in opaque technical devices ultrafast electron beam X-ray computed tomography (CT) scanners are available at the Helmholtz-Zentrum Dresden-Rossendorf (HZDR). Currently, measurement data must be initially downloaded after each CT scan from the scanner to a data processing machine. Afterwards, cross-sectional images are reconstructed. This limits the application fields of the scanners. For online observations and even automated process control of scanned objects a new modular data processing tool is presented consisting of user-definable pipeline stages that work independently together in a so called data processing pipeline that can keep up with the CT scanner's frame rate of up to 8 kHz. The data processing stages are arbitrarily programmable and combinable and are connected by a fast custom memory pool to optimize data transfer processes. As a result, this processing structure is not limited to CT application only. In order to achieve highest processing performances for the electron beam X-ray CT scanners all relevant data processing steps are individually implemented in separate stages using graphic processing units (GPUs) and NVIDIA's CUDA programming language. Data processing performance tests on two different high-end GPUs (Tesla K20c, GeForce GTX 1080) offer a slice image reconstruction performance that is well-suited for online application.**

***Index Terms*—Computed tomography, Image reconstruction, Multithreading, Parallel algorithms, Pipeline processing, Real-time systems**

I. Introduction

For contact-free investigations of rapidly moving structures in technical devices, like gas bubbles in liquids, ultrafast electron beam X-ray computed tomography (CT) measurement systems were developed at the Helmholtz-Zentrum Dresden-Rossendorf, firstly presented by Fischer et al. [1]. Usually, for CT scans a radiation source is directed to an object of investigation and the correspondingly attenuated radiation is measured behind the object by a detector line or (flat) panel. By collecting projection data from various angular positions a data set, called sinogram, is acquired that is used as input for CT reconstruction algorithms, like filtered back projection [2] or algebraic reconstruction technique [3], to obtain non-superimposed cross-sectional or volumetric material distributions of the scanned objects. Nevertheless, classical CT scanners are not suitable for the investigation of fast moving structures because of their limited frame rate which is due to mechanical rotation of the projection acquisition ensemble or the objects of investigation.

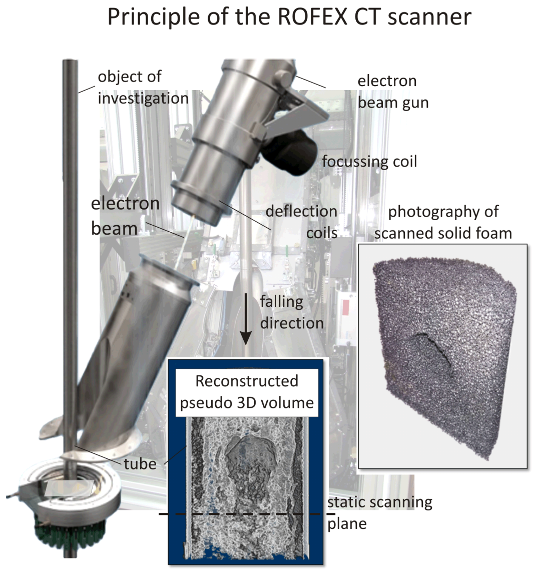

Fig. 1. Principle sketch of the ROFEX CT scanners, here used for three dimensional investigations in solid foams.

In order to overcome these difficulties, the **RO**ssendorf ultra**F**ast **E**lectron beam X-ray CT scanners (ROFEX) are operated with a focused electron beam that is circularly deflected on a tungsten target providing a moving X-ray source at the point of impact (see Figure 1). The electron beam can be deflected up to 8 kHz [1] which leads to a corresponding frame rate. One of these CT scanners is, e.g., equipped with a slightly axially displaced double layer ring detector that provides projection acquisition with a constant sampling frequency of 1 MHz. Each ring detector consists of 432 single CdTe detectors with an active area of $1.5 \times 1.5\ \mathrm{mm}^2$. As a result, a spatial resolution of 1-2 mm can be achieved in terms of gas-liquid contrast.

Currently, the ROFEX scanners are operated in off-line mode which means that a CT scan is initially blindly performed, the detector data is downloaded afterwards, and finally reconstructed by a recently developed data processing tool-kit that uses graphics processing units (GPUs) [4]. All single detectors are simultaneously sampled with an accuracy of 12 bit. Thus, a data rate of approximately 1.2 GByte/s is generated that must be efficiently pre-processed and reconstructed. In the near future, the CT scanners need to be operated in online mode to optimize the scanning procedure (visual inspection) and to automatically react to the process being observed (active process feedback control). Hence, the online data processing has to be able to reconstruct sinograms at the typical operating rate of 1 kHz with the potential to scale up to the full 8 kHz. Therefore, a new modular data processing tool is constructed using NVIDIAs' CUDA and multiple data stages performed in different simultaneous working threads. In this contribution related works, the architecture of the modular data processing program as well as first performance results are presented.

II. Related Work

Nowadays, there are numerous computed tomography systems requiring high computational power to provide data processing in real-time for a data stream bandwidth of several Gigabits per second [5], [6]. Thus, many research groups spent effort in accelerating image reconstruction algorithms using recent parallel hardware architectures.

Significant speedup over a parallel implementation on a single CPU can be achieved, for instance, by using GPUs. Vázquez, Garzón and Fernández [7], [8] presented a matrix approach to tomographic reconstruction for the filtered back projection (FPB) algorithm as well as simultaneous iterative reconstruction method (SIRT) using sparse matrix-vector products for the back and forward projection processes. With CUDA-capable devices the run-time could be improved by a factor of up to 42 compared to the CPU implementation. Mueller et al. [9], [10] discussed the suitability of different hardware accelerators, like GPUs, FPGAs or the IBM Cell B.E. architecture, for image reconstruction concluding that GPU-based approaches are more cost-effective than the other solutions.

The filtered back projection algorithm and iterative reconstruction methods have been accelerated by Díez et al. [11] on graphics processors resulting in significant speedups as well. A GPU-based image processing framework for online monitoring, called UFO, has been published by Vogelgesang et al. [12] providing an opportunity for real-time monitoring at synchrotron facilities. There, data processing is modeled by a graph of nodes with the objective of a simple extensibility and usability. The implemented algorithms are accelerated using the OpenCL framework. The results showed speedups between approximately 7 to 37 for their test cases.

Previously, the data processing process for ROFEX data could be successfully accelerated using GPUs [4]. Nevertheless, not all data processing steps are optimized for an execution on CUDA-capable devices. There are, e.g., multiple data transfers required that reduce the entire potential of optimal data processing. Furthermore, the data processing requirements for the ROFEX are not yet met by any of the existing image reconstruction frameworks. Due to the special geometry of the measuring system standard reconstruction algorithms, like the filtered back projection, could not be applied directly: an additionally massive amount of data preprocessing is required that is currently not implemented in existing libraries, such like UFO [12] or TOMO3DGPU [7]. Furthermore, the usage of a non-standard GPGPU streaming framework like StreamIt [13] or GStream [14] is not productive.

For these reasons, a new pipelined architecture is introduced that implements all the data processing steps on a GPU using CUDA. Its structure is slightly comparable to the UFO framework [12]. As shown in [8] and [11] iterative reconstruction can also be accelerated using graphics processors, but for real-time applications the filtered back projection algorithm is well-suited. Unlike [7], there is no pixel-ray matrix storage approach implemented but a pixel-driven back projection with direct computation in CUDA kernels.

III. The ROFEX image pipeline

In order to meet and exceed the required frame rate, the implementation of the data processing pipeline needs to fulfill the following requirements: (1) all data processing steps need to be executed on the (GPU) device to prevent multiple data transfers; (2) memory transfer operations and kernel execution must overlap by use of CUDA streams; (3) support for multiple and various GPUs; (4) the support of an adaptable memory concept to support various hardware configurations and (5) a simple extensibility and usability. Therefore, a generic and reusable structure, as shown in Figure 2, is introduced with a pipeline stage being its core element. Each stage contains a thread-safe input and output queue of user-defined size performing the communication and synchronization with neighboring stages. Each stage runs its own host thread and can perform user-defined tasks either on a host or a device. The computational power of multiple GPUs is exploited by splitting the stage for each available device. For the code implementation C++11 and its multi-threading support is used.

The GPU can only achieve highest compute utilization for a given problem, if memory transfer operations and kernel executions can overlap without locks or device synchronization. Such synchronizations occur when memory transfers are executed on the default stream or when allocating or releasing device memory. Thus, each stage uses its own stream which will also overlap kernel executions from different stages. Furthermore, a memory pool is introduced to manage the

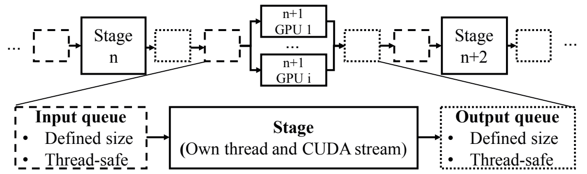

Fig. 2. A principal sketch of the pipeline's structure. Stages communicate through an input and an output queue.

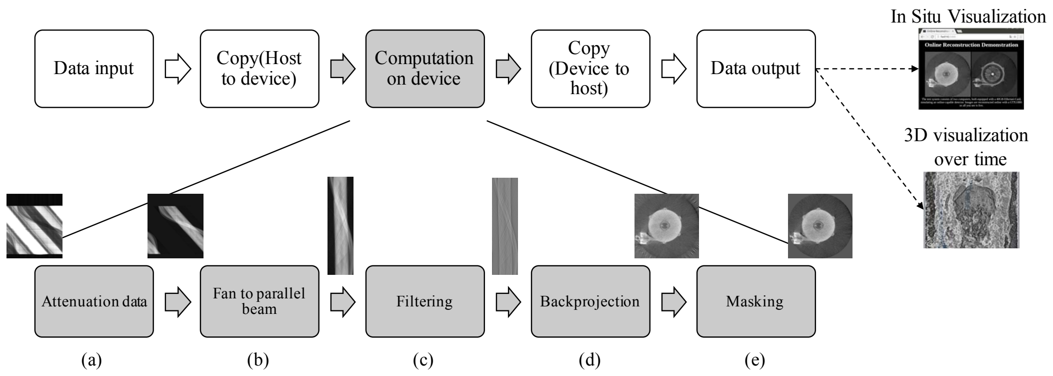

Fig. 3. The implemented ROFEX image processing pipeline with data preprocessing and reconstruction (gray represents stages accelerated using CUDA).

allocation of memory in the constructor of all used device data structures. In this way, the call of `cudaMalloc()` and `cudaFree()` is not required during data processing and maximum device task concurrency is ensured.

The image reconstruction pipeline of ROFEX is separated into the computation of (a) attenuation data, (b) the fan to parallel beam re-binning, (c) the filtering, (d) the back projection and (e) masking of the relevant area as shown in Figure 3. All boxes marked in gray are accelerated using CUDA as shown below:

To compute the attenuation data for ROFEX a dark I_{dark} and reference measurement I_{ref} need to be captured beside the data recording I_{data}. All static elements which appear in I_{ref} as well as in I_{data} are dissolved in the reconstructed image. The attenuation coefficient E for one pixel in the fan beam sinogramm is given in equation 1. This processing step is mapped into CUDA by creating a single CUDA thread for each pixel solving equation 1 independently.

$$E = -\log\left(\frac{I_{\mathrm{data}} - I_{\mathrm{dark}}}{I_{\mathrm{ref}} - I_{\mathrm{dark}}}\right) \tag{1}$$

For the fan to parallel beam re-binning stage a hash table is created at the program initialization. The CUDA kernel spans $N_{\mathrm{parDet}} \times 2 \cdot N_{\mathrm{parProj}}$ CUDA threads, with N_{parDet} being the number of detectors and N_{parProj} being the number of projections in the parallel ray sinogram over 180 degrees, performing the interpolation from fan to parallel beam geometry. Thus, N_{parDet} and N_{parProj} can be user-defined to balance between the pictorial quality and required reconstruction rate. For parallel projections $p(t, \Phi)$, where t represents the orthogonal distance of the ray from the iso center and Φ the angle between the ray and the abscissa, a symmetry property is valid [15] given in equation 2.

$$p(t, \Phi + \pi) = p(-t, \Phi) \tag{2}$$

For this reason, the CUDA kernel transforms the projections distributed over 360 degrees into a parallel ray sinogram from 0 to 180 degrees without loss of information. Hence, the back projection process requires only the half of the operations.

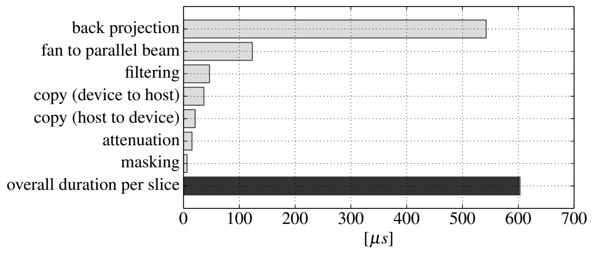

Fig. 4. The average runtime of each kernel on a GeForce GTX 1080 and the overall runtime per slice for $N_{\mathrm{parDet}} = 256$, $N_{\mathrm{parProj}} = 1024$ and 256×256 pixels. The overall runtime is shorter than the sum of the single durations.

The reconstruction is implemented using the filtered back projection algorithm [2]. Via the Fast Fourier Transform (FFT) the projections are transformed into the Fourier space, weighted with a filter function and inverse transformed via the inverse FFT. The FFT is implemented using the cuFFT-library. The filter function is stored in the architecture-specific constant memory of the GPU to ensure good memory access pattern. For the implementation of the back projection process there exist generally two approaches: a ray-driven and a pixel-driven approach. Due to the computational savings and its suitability for parallelization, the pixel-driven back projection is chosen. This approach starts at an image pixel center and computes the intensity contribution from the projection, following the ray path and locating its intersection with the filtered projection. The intersection does not line up with the discrete values and, thus, a 1D-interpolation is performed. [15] Our approach implements a simple linear interpolation. The required $\sin$- and $\cos$-values are precomputed and stored in constant memory, again. Finally, the reconstructed image is multiplied with a mask to hide non-relevant areas, e.g. the exterior of a pipe.

IV. Performance Evaluation

The performance of the system is evaluated on a Ubuntu 16.04. workstation, equipped with two Intel Xeon E5-2637 v3 CPUs (each with four physical cores at 3.5 GHz), 16 GB RAM, the founders edition of an NVIDIA GeForce GTX 1080

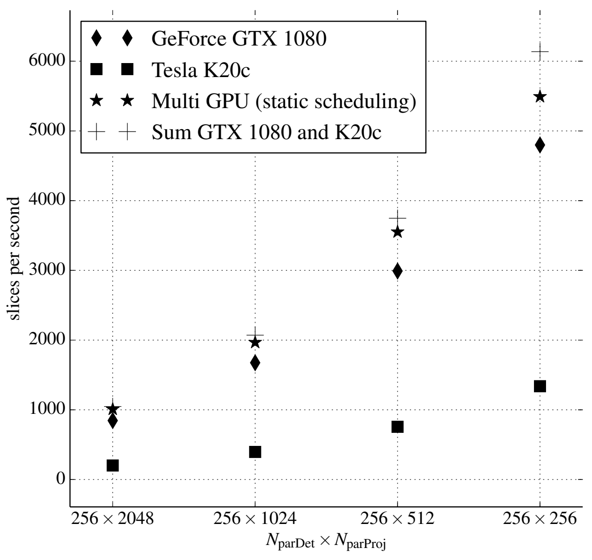

Fig. 5. Reconstruction rates for GeForce GTX 1080, Tesla K20c and both simultaneously with static scheduling for different sizes of the parallel ray sinogramm at 256×256 pixels in the reconstruction grid.

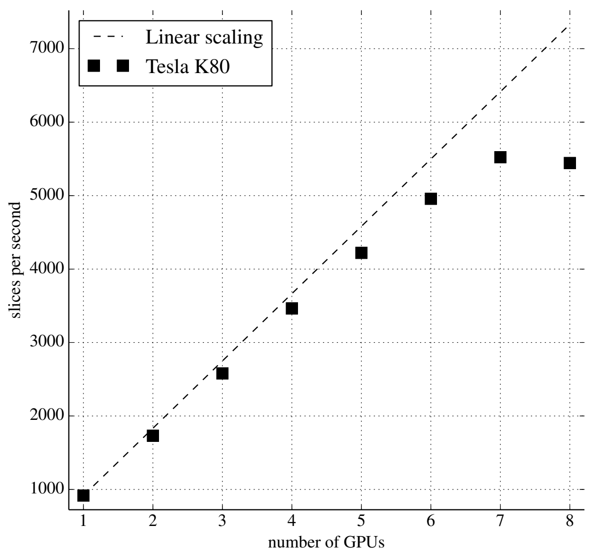

Fig. 6. Reconstruction rates for up to 8 Tesla K80 on the HPC-Cluster hypnos at HZDR ($N_{\mathrm{parDet}} = 256, N_{\mathrm{parProj}} = 512, 256 \times 256$ pixels in the reconstruction grid).

and a NVIDIA Tesla K20c. The code is compiled with GCC 5.4, the release candidate of CUDA 8 and the optimization option `-O3`. The analysis is performed with a test data set at a size of 432 detectors and 500 projections in the fan beam sinogram. The input data type is 16 bit unsigned integer, which is converted to single precision floating point numbers during the attenuation computation.

Figure 4 lists the average duration time of each device operation for a specific reconstruction scenario. The bottleneck of the implementation is currently the back projection algorithm that consumes most of the processing time with the given configuration options. Furthermore, the overall duration per slice is depicted. Its value is smaller than the summation of the single operations. That shows that kernels and memory transfer operations overlap and hide nearly completely behind the back projection kernel. Thus, the overall performance correlates with the most time consuming operation, in this case the back projection kernel.

The maximum achievable reconstruction rate is depicted in Figure 5 for a constant reconstruction grid size of 256×256 pixels for the GeForce GTX 1080 as well as the Tesla K20c as a function of various parallel sinograms' sizes. For both devices the reconstruction rate is directly proportional to the number of parallel projections in the parallel ray sinogram providing a balancing option between pictorial quality and required reconstruction rate, as expected. For these configurations, processing rates between 850 and 4500 slices per second are achievable providing real-time capability for the measuring system ROFEX.

The results also demonstrate the scalability of the multi GPU approach even for heterogeneous GPU devices. Up to now, a static scheduling between the GPUs is implemented achieving a processing rate only slightly below the maximum peak. Thus, the approach proves the ability of exploiting the computational power of systems equipped with homogeneous or heterogeneous GPUs.

Figure 6 depicts the reconstruction rates for up to 8 Tesla K80 on the HPC-Cluster hypnos at HZDR on one node. For up to 4 GPUs the implementation scales nearly linear. For more than 4 GPUs the slope decreases, with 8 GPUs the reconstruction rate saturates. It is under investigation where the scalability limitations arise from.

V. Summary and Outlook

The developed ROFEX image processing pipeline fulfills the initially set performance requirements. A single powerful GPU is already able to keep up with the typical frame rates of ROFEX measurements @ 1 kHz. The scalability between different GPU generations is shown. In order to scale even further the processing pipeline already features a round-robin distribution to multiple GPUs. Thus, processing at higher rates is a question of adding more GPUs.

The multi-device distribution currently uses a static distribution requiring GPUs of the same type for a perfect scaling. A dynamic load distribution is planned for an upcoming version of the software. The software is freely available at: https://github.com/HZDR-FWDF/RISA

References

[1] F. Fischer and U. Hampel, "Ultra fast electron beam X-ray computed tomography for two-phase flow measurement," *Nuclear Engineering and Design*, vol. 240, no. 9, pp. 2254 – 2259, 2010, experiments and {CFD} Code Applications to Nuclear Reactor Safety (XCFD4NRS). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029549309005755

[2] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[3] R. Gordon, R. Bender, and G. T. Herman, “Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography,” *Journal of Theoretical Biology*, vol. 29, no. 3, pp. 471 – 481, 1970. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022519370901098

[4] A. Bieberle, T. Frust, M. Wagner, M. Bieberle, and U. Hampel, “Data processing performance analysis for ultrafast electron beam X-ray CT using parallel processing hardware architectures,” *Flow Measurement and Instrumentation*, pp. –, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0955598616300280

[5] A. Rack, T. Weitkamp, S. B. Trabelsi, P. Modregger, A. Cecilia, T. dos Santos Rolo, T. Rack, D. Haas, R. Simon, R. Heldele, M. Schulz, B. Mayzel, A. Danilewsky, T. Waterstradt, W. Diete, H. Riesemeier, B. Müller, and T. Baumbach, “The micro-imaging station of the TopoTomo beamline at the ANKA synchrotron light source,” *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 267, no. 11, pp. 1978 – 1988, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168583X09004789

[6] A. Rack, S. Zabler, B. Müller, H. Riesemeier, G. Weidemann, A. Lange, J. Goebbels, M. Hentschel, and W. Görner, “High resolution synchrotron-based radiography and tomography using hard X-rays at the BAMline (BESSY II),” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 586, no. 2, pp. 327 – 344, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900207023315

[7] F. Vázquez, E. Garzón, and J. Fernández, “A matrix approach to tomographic reconstruction and its implementation on GPUs,” *Journal of Structural Biology*, vol. 170, no. 1, pp. 146 – 151, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S104784771000033X

[8] F. Vázquez, E. M. Garzón, and J. J. Fernández, “Matrix implementation of simultaneous iterative reconstruction technique (SIRT) on GPUs,” *The Computer Journal*, p. bxr033, 2011.

[9] K. Mueller, F. Xu, and N. Neophytou, “Why do commodity graphics hardware boards (GPUs) work so well for acceleration of computed tomography?” pp. 64 980N–64 980N–12, 2007. [Online]. Available: http://dx.doi.org/10.1117/12.716797

[10] F. Xu and K. Mueller, “Real-time 3D computed tomographic reconstruction using commodity graphics hardware,” *Physics in Medicine and Biology*, vol. 52, no. 12, p. 3405, 2007. [Online]. Available: http://stacks.iop.org/0031-9155/52/i=12/a=006

[11] D. C. Díez, H. Mueller, and A. S. Frangakis, “Implementation and performance evaluation of reconstruction algorithms on graphics processors,” *Journal of Structural Biology*, vol. 157, no. 1, pp. 288 – 295, 2007, software tools for macromolecular microscopy. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047847706002760

[12] M. Vogelgesang, S. Chilingaryan, T. d. Santos, and A. Kopmann, “UFO: A scalable GPU-based image processing framework for on-line monitoring,” in *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, June 2012, pp. 824–829.

[13] A. Udupa, R. Govindarajan, and M. J. Thazhuthaveetil, “Software pipelined execution of stream programs on GPUs,” in *Code Generation and Optimization, 2009. CGO 2009. International Symposium on*, March 2009, pp. 200–209.

[14] Y. Zhang and F. Mueller, “GStream: A general-purpose data streaming framework on GPU clusters,” in *2011 International Conference on Parallel Processing*, Sept 2011, pp. 245–254.

[15] J. Hsieh, *Computed tomography: principles, design, artifacts, and recent advances*. SPIE Bellingham, WA, 2009.