Data distribution on a multi-GPU node for TomoBayes CT reconstruction

Mohammed Chghaf, Nicolas Gac

Data distribution on a multi-GPU node for TomoBayes CT reconstruction

Mohammed Chghaf
Université Paris-Saclay, CNRS, CentraleSupélec, L2S
91192 Gif-sur-Yvette cedex, France
mohammed.chghaf@gmail.com

Nicolas Gac
Université Paris-Saclay, CNRS, CentraleSupélec, L2S
91192 Gif-sur-Yvette cedex, France
nicolas.gac@universite-paris-saclay.fr

***Abstract*—Computed tomography (CT) is an imaging technique that uses iterative algorithms to reconstruct the interior of large volumes. Graphics Processing Units (GPUs) are currently the preferred technology for computation acceleration. However, the collected data storage can exceed the internal memory of current GPUs and therefore requires costly CPU GPU data transfers. In this paper, we present a strategy of data distribution over several GPUs avoiding this bottleneck. We provide experimental results showing that our memory-saving method accelerates the iterative reconstruction. We achieve a better parallelisation efficiency using 8 GPUs to reconstruct a volume of size 4 GB than reconstructions based on centralised data storage on CPU.**

***Index Terms*—Computed Tomography, Multi GPU, Data Distribution, Iterative reconstruction**

I. Introduction

Reconstruction time is critical for Computed Tomography (CT) applied to Non-Destructive Testing (NDT) or medical imaging. For instance, in interventional radiology, filtered backprojection is prefered to compute-intensive iterative algorithms, although last ones offer better image quality.

When applied on huge volume (256^3 to 2048^3 voxels), GPU is well adapted to massively parallelise the projection and backprojection operators used in iterative algorithms. In that way, GPU has become the mainstream accelerator [1], and several tomography toolboxes such as Astra [2], Tigre [3] or TomoBayes [4] succeed to reduce computation time by one or two orders of magnitude. However, as the GPUs internal memory is limited, all the input projection data and the reconstructed volumes have to be stored on the host CPU and transferred to the GPU at each operator call. These additional communication costs notably reduce the parallelisation efficiency on multi-GPU nodes. The limited bandwidth of the PCI bus which connects the CPU host to the several GPU boards then becomes a bottleneck.

In order to leverage the potential of multi-GPU parallelisation, CPU-GPU communication times can be partially hidden thanks to streams like in [5], Tigre [6] or TomoBayes [7]. Managing small chunks of data allows overlapping of computation kernels and upload/download data transfers. Although communication costs are reduced in these works, data remains centralised on the CPU. The Message-Passing Interface version of the Astra toolbox [8] refined in [9] manages a data distribution on the several GPUs of a multi-node server. During iteration, data exchanges are made between GPUs and nodes through MPI.

In this paper, we propose an efficient data distribution strategy between multiple GPUs on the same node. This solution is based on high-speed peer-to-peer GPU communication. We provide experimental results for an iterative algorithm.

II. Iterative Computed Tomography algorithms

Iterative CT algorithms aim is to create a three-dimensional (3D) representation of an object using only external measurements. An X-ray source irradiates the object while a flat detector is placed on the opposite side. The intensity of the X-rays decreases due to the object variable density before reaching the sensor. The measurements of the decreased intensities by the detector cells form an image which is called a projection. In order to acquire several views, the object is rotated around Z-axis. A sinogram is created by stacking all of the measured attenuations from the detector cells acquired at different angles.

Mathematically, the projection operation can be expressed by an algebraic equation with the matrix-vector product: $g = Hf$; f is the volume to be reconstructed, g is the vector of measurements and $\mathbf{H}$ is the projection matrix. The backprojection operation is represented by $\mathbf{H}^\top$.

In iterative reconstruction methods, the projection and the backprojection operators correspond to the main computational burden. Therefore, reducing the calculation time implies speeding up these two operations through GPU parallelisation.

III. Proposed method: data partitioning

For cone-beam geometry, an intrinsic data dependency represents a lock for simple data distribution between GPUs. In the parallel-beam case, the problem can be easily solved by splitting the volume to equal slices perpendicular to the axis of rotation and equivalent to the number of GPUs used. The slices of the volume corresponding to the projections on the detector will be separate and will allow the calculations to be carried out independently. However, in the case of cone-beam CT, the divergent beams between the source and the detector can intersect more than one area of the volume. Therefore, to perform a projection where each GPU is responsible for a slice of the volume, some data from the two adjacent slices are

needed to perform the necessary calculation of the projection in these overlapping regions.

The proposed strategy decomposes the volume and the projections into slices of the same number of the used GPUs. This allows copying from the CPU to the GPU only the bare minimum of data in order to speed up the calculations. The remaining data needed for computing is then transferred using only the high-speed peer-to-peer GPU communication.

For the projection operator, the volume data are stored in the CPU; then we distribute them equally over all the GPUs used. Before starting the operation, we compute the limits of the sub-volume necessary to perform the calculation of each zone of the detector. The missing parts are then copied with direct peer-to-peer communication from the adjacent GPUs to the concerned GPU. Once the projection operation is completed, the measurements are then retrieved to the CPU.

Similarly, we can find the limits of each subset of projection measurements that will be used for backprojection for each GPU. The remaining data needed are then copied to the adjacent GPUs.

As shown in Fig. 1, generally, to calculate the projection P_n of a given sub-volume, we will need the sub-volume $V_{n_{middle}}$ stored in GPU_n, the sub-volume $V_{n_{up}}$ stored in GPU_{n+1} and the sub-volume $V_{n_{bottom}}$ stored in GPU_{n-1}. In this case, $V_{n_{up}}$ and $V_{n_{bottom}}$ are copied using peer-to-peer communication from GPU_{n+1} and GPU_{n-1}, respectively to GPU_n just before the projection operation.

The realisation of the iterative algorithm requires intermediate operations such as the calculation of the differences between the estimated measurements and the real measurements, image filtering steps or reduction operations. The proposed solution allows all these operations to be carried out directly on the GPUs without the need for intermediate exchanges between CPU and GPU.

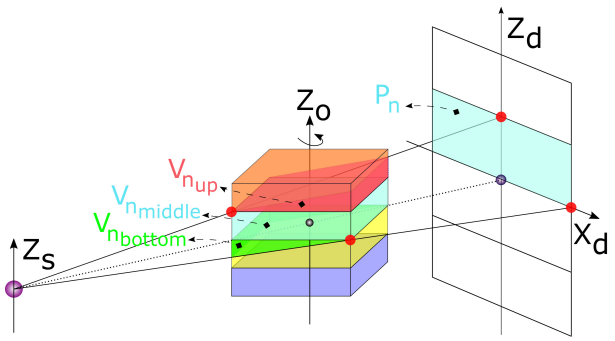


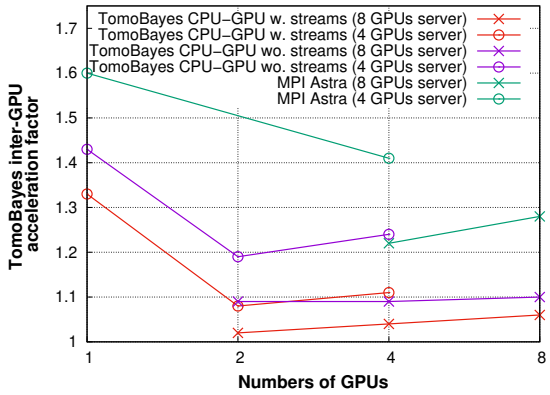
Fig. 1. Data partitioning for a cone-beam forward projection using 4 GPUs.

IV. Results

For the experiments, two servers made of intel Xeon CPU hosts with several GPUs were used. On the first one, 8 Maxwell Titan X GPUs are connected through a PCI gen3. On the second one, 4 V100 GPUs are connected through a NVLink bus. The reconstructions have been done for 1024^3 (4 GB) sized Shepp-Logan ghost.

In order to evaluate the impact of data communication, a simple loop with only $\mathbf{H}$ and $\mathbf{H}^\top$ has been evaluated. At each iteration, the volume f is updated: $f_{i+1} = \mathbf{H}^\top \mathbf{H} f_i$. The graph on Fig. IV presents the acceleration factor reached by our inter-GPU strategy for this $\mathbf{H}^\top \mathbf{H}$ loop in comparison with the CPU-GPU strategy, with and without streams, and the MPI Astra. On multiGPU servers, the acceleration factor increases with the number of GPUs. Our inter-GPU strategy is indeed less sensitive to the data communication needed at each iteration loop by the multiGPU compute distribution.

The table on Fig. IV exposes the reconstruction time for the $\mathbf{H}^\top \mathbf{H}$ loop and a complete iterative algorithm minimising the Mean Square (MS) criterium $||g - Hf||^2$. At each iteration, the intermediate operations are thus also taken into account: $f_{i+1} = f_i - 2\mathbf{H}^\top(g - \mathbf{H}f_i)$. Because of the data storage on GPU, the CUDA implementation of the intermediate computations has been eased for inter-GPU TomoBayes. It has therefore allowed a 3.5 acceleration factor compared with CPU-GPU TomoBayes. It also reaches a 1.23 acceleration compared to the all GPU Astra implementation.



	inter-GPU			
HtH (s)	5.8	6.1 ×1.05	6.4 ×1.09	7.4 ×1.28
MS (mn)	6.3	21.9 ×3.46	22.2 ×3.52	7.8 ×1.23

Fig. 2. Reconstruction for a 1024^3 volume with 100 iterations on servers with 8 Maxwell Titan X† and 4 V100*; (top) Acceleration factor obtained with TomoBayes inter-GPU for the HtH loop; (bottom) Processing time for a HtH loop and MS algorithm with 8 Titan X GPUs.

V. Conclusion

In this work, we clarified the existing parallelisation bottleneck on multiGPU server for cone-beam iterative CT reconstruction. We presented a data distribution and synchronization strategy avoiding the costly CPU-GPU communications. We showed a clear improvement in performance on iterative algorithms with a better multiGPU parallelisation potential benefit compared to some state of the art implementations.

[1] F. Xu *et al.*, "CT on GPU," *Phys. Med. Biol.*, 2007, open access.
[2] "Astra toolbox," 2012, www.astra-toolbox.com.
[3] A. Biguri *et al.*, "MATLAB-GPU TIGRE," *BPEX*, 2016, open access.
[4] C. Chapdelaine *et al.*, "TomoBayes," 2019, hal-01771489.
[5] H. Muthukrishnan *et al.*, "GPU Scaling," in *CT meet.*, 2018, open access.
[6] A. Biguri *et al.*, "multigpus TIGRE," 2019, arXiv:1905.03748.
[7] N. Georgin *et al.*, "MultiGPU BP/P," in *Fully 3D*, 2019, hal-02070223.
[8] W. Palenstijn *et al.*, "Astra toolbox," *ASCI*, 2017, open access.
[9] J.-W. Buurlage *et al.*, "A projection-based distributed tomographic reconstruction," in *SIAM PPSC*, 2020, open access.

† L2S/GEOPS server ; * Saclay-IA HPC plateform of Université Paris-Saclay supported by CNRS and Région Île-de-France.