International Journal on Magnetic Particle Imaging
Vol 9, No 1, Suppl 1, Article ID 2303043, 4 Pages

Infinite Science Publishing

Proceedings Article

Parallel MPI image reconstructions in GPU using CUDA

Klaus N. Quelhas [a,b,*] · Mark-Alexander Henn [a,c] · Ricardo C. Farias [b] · Weston L. Tew [a] · Solomon I. Woods [a]

[a] National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, USA
[b] Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil
[c] University of Maryland, College Park, MD 20742, USA
[*] Corresponding author, email: klaus.quelhas@nist.gov

Abstract

This work shows that it is possible to obtain faster MPI image reconstructions by implementing the algorithms in parallel in *Graphics Processing Units* (GPUs) using NVIDIA's CUDA (*Compute Unified Device Architecture*). While the parallel Kaczmarz's algorithm was slower than its serial version running in the *Central Processing Unit* (CPU), the parallel version of the Conjugate Gradient Normal Residual (CGNR) algorithm was about 58 times faster than its serial implementation, and about 10 times faster than the serial implementation of Kaczmarz's.

I. Introduction

Image reconstruction is a fundamental step in Magnetic Particle Imaging (MPI). It allows translating the remotely detected signals from magnetic nanoparticle (MNP) tracers into intelligible 1, 2 or 3D concentration maps [1]. Usually, the reconstruction process consists of solving large systems of equations [2] and, depending on the size of the problem — the number of measurements and image resolution — this computation can be extremely intensive and time-consuming.

Recently, a number of publications suggested the possibility of accelerating the reconstructions by implementing the algorithms to run in parallel in *Graphical Processing Units* (GPUs) [3–5]. Even though GPU-accelerated image reconstructions are already a reality for other biomedical imaging methods such as MRI or CT, no significant progress in this direction has been reported for MPI so far in the literature. One of the reasons may be the fact that Kaczmarz's is not easily run in parallel, in contrast to other algorithms. A comprehensive study compared the performance of several iterative methods, including the Kaczmarz's algorithm and the *Conjugate Gradient Normal Residual* (CGNR), reporting that the Kaczmarz's algorithm may not be the best one to be run in GPU, while the parallel CGNR has shown a speedup of more than 30 times in GPU when compared to its serial version run in CPU [6]. The study, however, does not compare the implemented parallel algorithms against the serial implementation of Kaczmarz's.

This work describes a first attempt to speedup the execution of MPI image reconstruction algorithms by running them in parallel in GPU using NVIDIA's CUDA (*Compute Unified Device Architecture*) [7], comparing the performance of them against their serial implementations in CPU.

II. Methods and materials

The following two sections briefly describe the architecture adopted, as well as the algorithms studied in this work.

10.18416/ijmpi.2023.2303043

II.I. Brief description of CUDA

CUDA is an architecture adopted in NVIDIA's GPUs consisting of a hardware specialized in performing massive numerical computations in parallel (such as image processing), and a software package that allows developing applications that take advantage of the *Single-Instruction Multiple-Thread* (SIMT) architecture model. The computation over large sets of data is performed simultaneously by thousands of *cores* built within *Streaming Multiprocessors* (SMs) which can access different memory levels and run sets of instructions called *kernels* in parallel.

II.II. Algorithms

All the algorithms studied in this work are designed to numerically find the solution of the system of equations

$$K \cdot c = s \tag{1}$$

typically adopted for MPI image reconstructions. The convolution matrix $K \in R^{n \times m}$ is analogous to the system matrix, and contains the measurement system response in such a way that each row contains the *point spread function* (PSF) evaluated over the whole grid for a given *field-free point* (FFP) position, and each of the n columns contains one full scan signal for a delta sample placed in each grid *voxel*. The vector $c \in R^m$ is the unknown concentration grid and the vector $s \in R^n$ contains the measured signal. Both the convolution matrix and the signal vector may contain data either in time or frequency domain (such as in the system matrix method). For simplicity, the signal s is real-valued, contains no noise, and the following algorithms do not apply regularization or any constraints, such as non-negativity.

II.II.1. Kaczmarz's algorithm

The Kaczmarz's algorithm is an iterative row projection method that works by, at each iteration i, projecting the current estimate onto the (i)th row of the convolution matrix and then correcting the estimate. The $(i+1)$th estimate is given by:

$$c^{(i+1)} = c^{(i)} + \frac{s_i - K_i \cdot c^{(i)}}{||K_i||^2} K_i^T \tag{2}$$

where K_i is the (i)th row of the convolution matrix and s_i is the (i)th measurement. This equation has limited matrix-vector operations, and the only operations that could be run in parallel are the dot product and the norm calculation. Since the algorithm works in a way such that each iteration depends on the previous one, it has to be executed serially over all the rows of K, compromising the potential for speedup.

II.II.2. Cimmino's algorithm

The Cimmino's algorithm works almost the same way as Kaczmarz's, but instead of updating the unknown estimate one row at a time, it averages the correction estimates of all the rows of K. The next estimate is given by:

$$c^{(i+1)} = c^{(i)} + \frac{1}{m} \sum_{l=1}^{m} \frac{s_l - K_l \cdot c^{(i)}}{||K_l||^2} K_l^T \tag{3}$$

This small change turns the algorithm into a great candidate to be run in parallel, since each thread could operate over a single row of the convolution matrix simultaneously. The averaging also can be run in parallel, with each thread operating over a single column of K, computing the new *voxel* estimate. This algorithm is proven to converge to the least-squares solution, but as will be seen, the convergence of this algorithm is very slow.

II.II.3. Hybrid Cimmino's/Kaczmarz's

This algorithm is meant to benefit from the fast-converging properties of the Kaczmarz's algorithm and the highly parallel structure of Cimmino's. The idea is to break the m rows of the system of equations into *runs* that would execute Cimmino's algorithm over this smaller subset. The *runs* are executed serially, which compromises parallelization but increases the convergence rate. For optimum performance, the number of rows processed in each *run* is the same as the number of GPU cores, and in the limit where the number of rows per *run* is equal to one the algorithm reduces to Kaczmarz's.

II.II.4. Conjugate Gradient Normal Residual - CGNR

The *conjugate gradient* (CG) method is an iterative method used for solving systems with a symmetric positive definite matrix. For MPI image reconstruction, the CGNR method is applied to solve the normal equation $(K^T K \cdot c = K^T \cdot s)$ by iterating over a sequence of basis vectors p^i that are mutually conjugate with respect to $(K^T K)$. Each iteration computes the new estimate

$$c^{(i+1)} = c^{(i)} + \frac{||K^T \cdot r^{(i)}||^2}{||K \cdot p^{(i)}||^2} p^{(i)} \tag{4}$$

$$p^{(i)} = K^T \cdot r^{(i)} + \frac{||K^T \cdot r^{(i)}||^2}{||K^T \cdot r^{(i-1)}||^2} p^{(i-1)} \tag{5}$$

and the residual $r^i = s - K \cdot c^i$. The dominance of matrix-vector multiplications over other operations suggests a good potential for parallelization.

III. Experiments

The experiments were performed on a laptop equipped with an Intel(R) i7-7700HQ processor, 32 GB of RAM

10.18416/ijmpi.2023.2303043

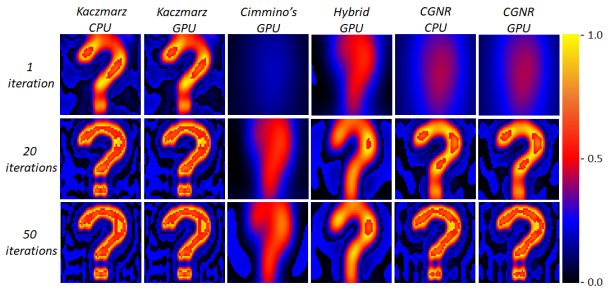

Figure 1: Reconstruction results after 1 iteration (top line), 20 iterations (middle line) and 50 iterations (bottom line). The color scale on the right side describes the relative reconstructed concentrations.

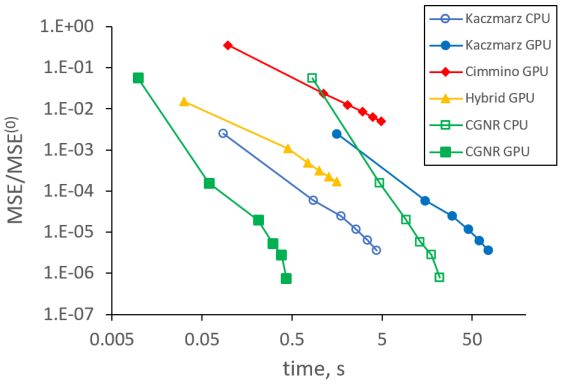

Figure 2: Convergence of the different algorithms, denoted by the MSE as function of time. The points represent the relative MSE values for 1, 10, 20, 30, 40 and 50 iterations.

memory, and a NVIDIA GeForce GTX 1060 GPU containing 10 *SMs* with 128 cores each, and 6 GB of dedicated RAM memory. Simulated signal data was generated using an analytical model [8] for a hypothetical 51x51 element phantom grid with a "?" shape, totaling 20,000 measurements for a full scan following a Lissajous FFP trajectory. With this configuration, the size of the convolution matrix is about 416 MB.

Both serial and parallel versions of the Kaczmarz's algorithm were tested, as well as the parallel Cimmino's algorithm, the parallel hybrid one, and the serial and parallel implementations of CGNR. For performance comparison purposes, one iteration is considered a set of operations that cover all the rows of the system of equations, and there was no row selection, meaning that all of them were covered by all algorithms at each iteration. The execution times reported are averages of five executions of each algorithm, and the mean square errors (MSE) were computed with relation to the input signal to assess the accuracy of the reconstructions. The values were taken for 1, 10, 20, 30, 40 and 50 iterations.

IV. Results and discussion

Figure 1 shows the reconstruction results for the algorithms studied in this work. The Kaczmarz's algorithm shows good reconstructions after only a few iterations, a performance followed by CGNR and the Hybrid Cimmino's/Kaczmarz's algorithm. The Cimmino's algorithm, on the other hand, shows a very poor convergence rate.

A comparison of the algorithms' convergence as function of time (Figure 2) shows that the parallel CGNR implementation performs better than the other algorithms. In comparison to its serial implementation, the parallel CGNR has show an average *speedup* of 58.2x, and when compared to the serial Kaczmarz implementation (the second in performance) it ran nearly 10x faster in average. The quality of the reconstruction, however, has to be taken into account when comparing the performance of two algorithms that converge at different rates, so considering the time taken to reach a relative MSE lower than 10^{-5}, we find a *speedup* of 11.2x, against 8.3x when considering the time taken to reach less than $5 \cdot 10^{-4}$. It happens because the Kaczmarz's algorithm converges very quickly in the first iterations but the convergence speed slows down as the estimate gets closer to the optimal solution. The parallel implementation of Kaczmarz algorithm has shown a poor performance, running 17.5x slower than its serial version, while Cimmino's algorithm has shown no *speedup* and the worst reconstruction results. The Hybrid Cimmino's/Kaczmarz's resulted in an execution 2.5x faster than the serial Kaczmarz's algorithm in average, but delivered less accurate reconstructions.

V. Conclusion

This work has demonstrated the possibility of obtaining faster MPI image reconstructions by implementing the algorithms in parallel using CUDA-enabled GPU cards. While the parallel Kaczmarz's algorithm has shown poor performance in GPU, the parallel CGNR performed much better than the other algorithms studied. Future studies will assess the performance of these and other algorithms when operating over noisy data, by employing regularization, weights, row selection and matrix preconditioning to reach better convergence.

Acknowledgments

The authors acknowledge funding from NIST's Innovations in Measurement Science (IMS) grant.

Author's statement

Authors state no conflict of interest.

References

[1] B. Gleich and J. Weizenecker. Tomographic imaging using the nonlinear response of magnetic particles. *Nature*, 435(7046):1214–1217, 2005, doi:10.1038/nature03808.

[2] T. Knopp and T. M. Buzug, Magnetic Particle Imaging: An Introduction to Imaging Principles and Scanner Instrumentation. Berlin/Heidelberg: Springer, 2012, doi:10.1007/978-3-642-04199-0.

[3] M. Storath, C. Brandt, M. Hofmann, T. Knopp, J. Salamon, A. Weber, and A. Weinmann. Edge preserving and noise reducing reconstruction for magnetic particle imaging. *IEEE Transactions on Medical Imaging*, 36(1):74–85, 2017, doi:10.1109/TMI.2016.2593954.

[4] P. Vogel, S. Herz, T. Kampf, M. A. Rückert, T. A. Bley, and V. C. Behr. Low latency real-time reconstruction for MPI systems. *International Journal on Magnetic Particle Imaging*, 3(2):8, 2017, doi:10.18416/ijmpi.2017.1707002.

[5] X. Chen, Z. Jiang, X. Han, X. Wang, and X. Tang. The reconstruction of magnetic particle imaging: Current approaches based on the system matrix. *Diagnostics*, 11(773):18, 2021, doi:10.3390/diagnostics11050773.

[6] J. M. Elble, N. V. Sahinidis, and P. Vouzis. GPU computing with Kaczmarz's and other iterative algorithms for linear systems. *Parallel Computing*, 36:215–231, 2009, doi:10.1016/j.parco.2009.12.003.

[7] Reference is made to commercial products to adequately specify the experimental procedures involved. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that these products are the best for the purpose specified.

[8] M.-A. Henn, K. N. Quelhas, T. Q. Bui, and S. I. Woods. Improving model-based MPI image reconstructions: Baseline recovery, receive coil sensitivity, relaxation and uncertainty estimation. *International Journal on Magnetic Particle Imaging*, 2022, doi:10.18416/IJMPI.2022.2208001.