Embarrassingly Parallel GPU Based Matrix Inversion Algorithm for Big Climate Data Assimilation

M. Varalakshmi, VIT University, Vellore, India

Amit Parashuram Kesarkar, National Atmospheric Research Laboratory, Chittoor, India

Daphne Lopez, VIT University, Vellore, India

ABSTRACT

Attempts to harness the big climate data that come from high-resolution model output and advanced sensors to provide more accurate and rapidly-updated weather prediction, call for innovations in the existing data assimilation systems. Matrix inversion is a key operation in a majority of data assimilation techniques. Hence, this article presents out-of-core CUDA implementation of an iterative method of matrix inversion. The results show significant speed up for even square matrices of size 1024 X 1024 and more, without sacrificing the accuracy of the results. In a similar test environment, the comparison of this approach with a direct method such as the Gauss-Jordan approach, modified to process large matrices that cannot be processed directly within a single kernel call shows that the former is twice as efficient as the latter. This acceleration is attributed to the division-free design and the embarrassingly parallel nature of every sub-task of the algorithm. The parallel algorithm has been designed to be highly scalable when implemented with multiple GPUs for handling large matrices.

KEYWORDS

Big Climate Data, Convergence Rate, GPU, Iterative Method, Matrix Type Identification, Numerical Weather Prediction, Parallel Matrix Inverse, Parallel Reduction

1. INTRODUCTION

The advent of Big data technology has brought a great revolution in the science of Numerical Weather Prediction. Big data in NWP actually refers to 'climate big data' that come from rapid and dense observations from advanced sensors and very high-resolution model output. A ten-fold increase in the model resolution would require 10^4 more computations for the four dimensions in space and time. To achieve this massively challenging throughput and to fully utilize this big data so as to provide more accurate and rapidly updated weather prediction, innovations have to be brought to the existing Data Assimilation and NWP systems (Big Data Assimilation) (Miyoshi et al., 2016a; Miyoshi et al., 2016b). This can help strengthen our early warning system against regional, sudden and severe calamities such as hurricanes, heavy rain, flooding, landslides and the alike. Innovative research has already started towards speeding up the various phases of NWP such as observation data processing, model run and data transfer between model and DA. Even in the Data assimilation phase, ways to improve storage and processing of large matrices and vectors can be explored. With the three spatial dimensions and one temporal dimension considered in Variational data assimilation algorithms and Kalman Filter based assimilation algorithms, the atmospheric state variables such as Wind, Pressure,

DOI: 10.4018/IJGHPC.2018010105

Humidity etc at all grid points for various vertical layers and time instants are represented in a vector with around 10^8 entries. Likewise, the measurement vector contains 10^6 observation entries. Due to large size of these vectors, the resulting model error covariance and observation error covariance matrices too will be large, of the order of $O(10^8X10^8)$. Hence the performance of these assimilation methods depends on the design and implementation of better algorithms for processing of large matrices in general and inversion in particular, and this was the impetus behind our proposed work.

The massive number crunching capacity needed to work with large matrices can be made possible by employing Graphics Processing Units (GPUs). CUDA is well suited for data-parallel algorithms (Garland et al., 2008) such as shallow water model (Playne & Hawick, 2015), delivering high computational throughput if few design principles are followed to fully utilize the GPU's processor cores and their shared memory that is critical to the performance of many efficient algorithms. Various improvements made to the storage format for efficient execution of SpMV operations on GPUs (Gao, Qi & He, 2016; Koza, Matyka, Szkoda & Mirosław, 2014; Dziekonski, Lamecki & Mrozowski, 2011) have shown this. Wu, Ke, Lin and Jhan (2014) claim that adjusting the number of threads dynamically helps to completely utilize the compute power of GPUs. Modeling tools (Zouaneb, Belarbi & Chouarfia, 2016) also lend a helping hand in validating task scheduling on GPUs and analyzing the performance. Earlier studies show that GPU implementations are several times faster than its CPU counterpart (Helfenstein & Koko, 2012) and can be efficient if the matrix is represented and processed using the two-dimensional textures that GPUs are optimized for (Galoppo, Govindaraju, Henson & Manocha, 2005). Further studies have revealed that parallel implementation of algorithms on hybrid platform consisting of CPU and GPUs (Ezzatti, Quintana & Remón Gómez, 2011a; Benner, Ezzatti, Quintana-Ortí & Remón, 2009; Ezzatti, Quintana-Orti, & Remon, 2011b) has proved to be more efficient for both small and large size matrices than the pure GPU implementation.

To support the efficient execution of linear algebra applications, there are several linear Algebra libraries optimized for GPU architecture such as CUBLAS and MAGMA for finding matrix inverse. MAGMA linear algebra C/C++ library (A. Chrzeszczyk & J. Chrzeszczyk, 2013) provides code for calculating matrix inverse for a regular matrix and positive definite matrix both in single precision and double precision. However, these libraries are not efficient for certain applications and there are other findings that show that further enhancements can be made to these implementations.

According to Haidar, Abdelfatah, Tomov and Dongarra (2017), high performance GPU-only algorithm developed for dense Cholesky factorization to run on latest GPUs and the hybrid panel-based LU decomposition algorithm outperform the existing libraries. The tile data layout followed in Cholesky-based matrix inversion (Ibeid, Kaushik, Keyes & Ltaief, 2011) results in up to 5 and 6-fold improvement compared to the equivalent routines from MAGMA V1.0 by completely removing the synchronization points and unlike Magma it is not memory-limited and can scale beyond the available device memory. Efficient batched solvers have been developed for a set of small dense matrices as the pre-existing solvers were either just memory-bound, or even if highly optimized, did not exceed in performance the corresponding CPU versions (Haidar, Dong, Luszczek, Tomov & Dongarra, 2015). If matrix inversion algorithms are tailored to handle specific application requirements, they outperform the methods employed in the standard libraries to calculate direct inverse (Prabhu, Rodrigues, Edfors & Rusek, 2013; Ylinen, Burian & Takala, 2003; Xingbo, 2011).

Moreover, these libraries employ direct methods that either use LU-decomposition with partial pivoting or Cholesky decomposition, for factorization of matrix. While the former suffers from lack of optimal stability, high convergence time for sparse matrices as compared to dense matrices and inability to find approximate solution (Agarwal & Mehr, 2014), the latter is not very robust and works only for symmetric positive definite matrices. On the other hand iterative methods are more stable, simpler, less prone to numerical errors, best suited for large matrices due to smaller storage requirements and more specifically, efficient for sparse matrices. They compensate for individual and accumulation of round-off errors as they are a process of successive refinement (Jamil, 2012).