Algorithm Design for Large Scale FFT-Based Simulations on CPU-GPU Platforms

Extended Abstract

Anuva Kulkarni
Electrical and Computer Engineering,
Carnegie Mellon University
Pittsburgh, Pennsylvania
anuvak@andrew.cmu.edu

Franz Franchetti
Electrical and Computer Engineering,
Carnegie Mellon University
Pittsburgh, Pennsylvania
franzf@cmu.edu

Jelena Kovačević
Electrical and Computer Engineering,
Carnegie Mellon University
Pittsburgh, Pennsylvania
jelenak@cmu.edu

ABSTRACT

Extreme memory requirements and high communication overhead prevent scaling of large scale iterative simulations involving parallel Fast Fourier Transforms (FFTs) to higher grid sizes, which is necessary for high resolution analysis. To overcome these limitations, we propose an algorithm to run stress-strain simulations on CPU-GPU platforms for larger problem sizes using irregular domain decomposition and local FFTs. Early results show that our method lowers iteration cost without adversely impacting accuracy of the result.

CCS CONCEPTS

• **Theory of computation → Parallel algorithms**; • **Data intensive parallel algorithms**;

KEYWORDS

FFT, Domain Decomposition, Communication-Avoiding Algorithms

1 INTRODUCTION

Large scale simulations running on machines with many cores model various phenomena in physics, biological sciences and engineering. However, high data movement and memory usage are often the chief limiting factors in computation in such methods [4]. Particularly, simulations involving partial differential equations (PDEs) usually make use of large parallel FFTs, which use all-all communication. Scaling the problem size results in prohibitive memory requirements and communication bottlenecks, which makes high resolution analysis with finer and finer grids impossible. One such method is the Moulinec-Suquet Composite (MSC) Basic Scheme, which is a FORTRAN scheme for local stress-strain computation in composites [2] [1].

In MSC Basic Scheme, a PDE with periodic boundary conditions is solved iteratively using convolution with Green's functions [3] to obtain local stress and strain fields. Increasing the resolution is desirable to study interesting behavior at grain boundaries, but larger problems require parallel FFT computations (3-D FFT for each tensor component). For serial code, memory requirement for a 32^3 grid is approximately 70 MB, while that for a 1024^3 grid is 2272 GB. This severely limits scaling to even larger grid sizes. Thus, a new generation of algorithm and software co-design is required to deal with challenges arising in extreme scale computing, such as communication overhead, difficulty in scaling and implementation of existing algorithms on newer machines and hardware. In this work, we describe MSC Alternate Scheme, an algorithm designed to run stress-strain simulations for large datasets on heterogeneous platforms with GPUs by using irregular domain decomposition and local FFTs to reduce communication cost.

In the background section, we describe the MSC Basic Scheme. The MSC Alternate Scheme is described in the next section, followed by some proof-of-concept results.

ICPP'18, August 2018, Eugene, Oregon USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

In this section, we describe the MSC Basic Scheme in more detail. Note that Einstein notation is used to represent tensor components and operations. Thus, A_{ij} refers to component (i, j) of the rank-2 tensor A. Repetition of indices implies a summation over those particular indices. An important tensor operation is the *contraction of indices* (denoted by ':'). Eg., $C_{ijkl} : D_{ij} = \sum_i \sum_j C_{ijkl} D_{ij} = E_{kl}$ and yields a rank-2 tensor.

The MSC Basic Scheme is a fixed-point iterative numerical method used as an alternative to Finite Element Methods (FEM) to compute local stress and strain fields using Hooke's law. The pseudocode for MSC Basic Scheme is as given below. $\epsilon(\mathbf{x})$ and $\sigma(\mathbf{x})$ are strain and stress tensor fields at point $\mathbf{x}$ respectively. $C_{ijkl}(\mathbf{x})$ is the rank-4 stiffness tensor. E is initial average strain. $\hat{\Gamma}_{mnkl}(\boldsymbol{\xi})$ is the Green's operator in Fourier space at frequency point $\boldsymbol{\xi}$. The convergence error is e_s and tolerance error is e_{tol}. $\Delta\epsilon_{kl}$ is the computed perturbation in component (k, l) of the strain tensor. Superscripts indicate iteration number. The iterative scheme continues till convergence is reached. For more details, refer to [2].

The convolution with Green's function requires computation of 3D FFTs of each of the 9 components of the stress field, hence the need for extensive resources for large grids. MSC Basic Scheme is implemented in serial FORTRAN and MPI parallel (using FFTW) versions.

3 PROPOSED METHOD

This section briefly describes the proposed algorithm, designed to be implemented on a CPU-GPU hardware setup. For the purpose

Algorithm 1 MSC Basic Scheme

Initialize:
  $\epsilon^0 \leftarrow E$,
  $\sigma^0_{mn}(\mathbf{x}) \leftarrow C_{mnkl}(\mathbf{x}) : \epsilon^0_{kl}(\mathbf{x})$
while $e_s > e_{tol}$ **do**
  $\hat{\sigma}^i_{mn}(\boldsymbol{\xi}) \leftarrow \text{FFT}(\sigma^i_{mn}(\mathbf{x}))$
  Check convergence
  $\Delta\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi}) \leftarrow \hat{\Gamma}_{klmn}(\boldsymbol{\xi}) : \hat{\sigma}^i_{mn}(\boldsymbol{\xi})$
  Update strain: $\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi}) \leftarrow \hat{\epsilon}^i_{kl}(\boldsymbol{\xi}) - \Delta\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi})$
  $\epsilon^{i+1}_{kl}(\mathbf{x}) \leftarrow \text{IFFT}(\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi}))$
  Update stress: $\sigma^{i+1}_{mn}(\mathbf{x}) \leftarrow C_{mnkl}(\mathbf{x}) : \epsilon^{i+1}_{kl}(\mathbf{x})$

of algorithm development, a MATLAB-FORTRAN workflow has been used to build a prototype and obtain preliminary results.

After initialization on the CPU side, an irregular domain decomposition method decomposes the volume into smaller regions with smooth stress fields (grain interiors). Data models are used to communicate initial stress fields in the smaller volumes to GPUs, with one small volume per GPU. Grain boundaries are communicated at full resolution. In each GPU-based iteration, local FFTs are performed on these fields and convolution with the Green's function is computed. Then, communication between GPUs serves to transfer parts of the result to respective grains on different GPUs so that stress update for a grain is a self-contained problem. The effect of convolution is summarized by data communicated from different GPUs. This makes the GPU part of the code intrinsically parallel. In this way, stress and strain fields are updated till convergence. A small number of iterations are performed on the CPU side without approximations. However, a majority of the iterations are performed in a distributed manner on the GPUs at a lowered communication cost since communication consists of only a few coefficients to summarize the effect of convolution.

4 RESULTS

For proof-of-concept results, a simple microstructure test dataset with two types of grain orientations was created using MATLAB for various grid sizes. Grains in each grid of size N^3 are $N/2 \times N/2 \times N/2$ cubes arranged in a periodic lattice. The cubical shape makes it easier to test the prototype with simple windowing techniques. More complicated pre-processing will be used for irregularly shaped grains.

In the MSC Alternate Scheme, n lower cost iterations of the fixed-point method are to be performed on GPUs using data models and local FFTs. A few (4 to 5) high cost iterations are performed using the MSC Basic Scheme to reduce approximation errors in the final answer. The plot in Fig. 1 shows the convergence of stress fields in MSC-Basic Scheme and MSC Alternate Scheme for different values of n in a simulation of size $128 \times 128 \times 128$. We observe that for equal error thresholds for both methods, number of iterations for convergence changes depending on n, but not drastically. The mismatch in stress field between the original and proposed method, arising due to approximations is about 4% for Case 1 of $n = 10$ and $n = 15$ and about 2% for $n = 10$, Case 2. This is due to convergence to a different local minimum which is at a small deviation from the MSC-Basic Scheme. GPU-side iterations have a lowered cost in

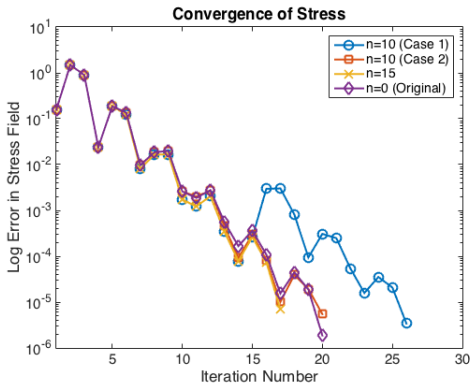

Figure 1: Convergence in stress for problem size 128^3. Case 1 of n=10 refers to performing iterations 5 to 15 on GPUs, and case 2 refers to performing iterations 10 to 20 on GPUs.

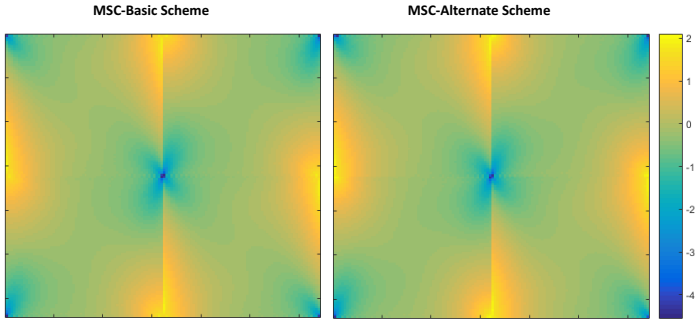

Figure 2: Comparison of stress fields in final iteration for the $\sigma_{11}(\mathbf{x})$ component in MSC Basic and MSC Alternate Schemes ($n = 10$, Case 2).

terms of communication and data movement. Metrics for iteration cost are not provided here.

5 CONCLUSIONS & FUTURE WORK

The proposed MSC Alternate Scheme is a co-design of algorithm and software for heterogeneous platforms. It enables scaling of stress-strain simulations to large grids by overcoming high memory requirements and communication bottlenecks. The algorithm uses small local FFTs and data modeling to perform iterations with a lowered cost, which converge to the same solution as the MSC-Basic Scheme with a small accuracy tradeoff, as is seen in proof-of-concept results presented here. Future work includes implementation and testing on various platforms.

REFERENCES

[1] R. A. Lebensohn. 2001. N-site modeling of a 3D viscoplastic polycrystal using fast Fourier transform. *Acta Materialia* 49, 14 (2001), 2723–2737.

[2] H. Moulinec and P. Suquet. 1998. A numerical method for computing the overall response of nonlinear composites with complex microstructure. *Computer methods in applied mechanics and engineering* 157, 1-2 (1998), 69–94.

[3] Toshio Mura. 1983. Micromechanics of Defects in Solids. *The Journal of the Acoustical Society of America* 73, 6 (1983), 2237–2237. https://doi.org/10.1121/1.389536 arXiv:http://dx.doi.org/10.1121/1.389536

[4] Y. Sabharwal, S. Garg, R. Garg, J. Gunnels, and R. K. Sahoo. 2008. Optimization of Fast Fourier Transforms on the Blue Gene/L Supercomputer. *High Performance Computing - HiPC 2008: 15th International Conference, Bangalore, India, December 17-20, 2008. Proceedings* (2008), 309–322. https://doi.org/10.1007/978-3-540-89894-8_29