Design and Specification of Large-scale Simulations for GPUs using FFTX

Extended Abstract

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

Daniele Spampinato
Electrical and Computer Engineering,
Carnegie Mellon University
Pittsburgh, Pennsylvania
spampinato@cmu.edu

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

ABSTRACT

Large-scale scientific simulations can be ported to heterogeneous environments with GPUs using domain decomposition. However, Fast Fourier Transform (FFT) based simulations require all-to-all communication and large memory, which is beyond the capacity of on-chip GPU memory. To overcome this, domain decomposition solutions are combined with adaptive sampling or pruning around the domain to reduce storage. Expression of such operations is a challenge in existing FFT libraries like FFTW, and thus it is difficult to get a high performance implementation of such methods. We demonstrate algorithm specification for one such simulation (Hooke's law) using FFTX, an emerging API with a SPIRAL-based code generation back-end, and suggest future extensions useful for GPU-based scientific computing.

CCS CONCEPTS

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

KEYWORDS

GPU, Fast Fourier Transforms, Domain Decomposition, Algorithms

1 INTRODUCTION

Supercomputers with thousands of cores are used to run scientific simulations to study various phenomena in science and engineering. However, legacy Fortran simulation codes cannot fully take advantage of modern heterogeneous computing environments due to various reasons associated with the mathematical operations involved in the algorithm. For example, large-scale simulations involving parallel FFTs require all-to-all communication and extreme memory requirements well beyond the small on-chip memory of a GPU.

Thus, algorithm redesign and restructuring is necessary to port legacy Fortran codes to heterogeneous systems with GPUs, and involves factoring in various communication latencies and GPU memory constraints. Algorithm innovations such as data-dependent sampling patterns and pruning can be used, but are very difficult to optimize since they are usually too complex to be expressed with an API like FFTW. The user then cannot leverage the benefit of highly optimized parallel FFT libraries like FFTW [4], MKL [5], etc. In this work, we describe a case study of such a problem and suggest a solution using emerging frameworks for building high-performance FFT-based applications on exascale machines.

For our case study, we use algorithm design strategies from our on-going work on porting a large-scale FFT-based simulation called MASSIF to heterogeneous environments. Micromechanical Analysis of Stress-Strain Inhomogeneities with Fourier transforms (MASSIF) is an FFT-based stress-strain simulation method for composites [10], [9], [7], [11]. The method solves a partial differential equation (PDE) by convolving 3D tensor fields with rank-4 Green's function. The largest size currently simulated with MPI is $1024 \times 1024 \times 1024$ with a memory requirement of more than 2TB. Scaling and accelerating the MASSIF simulation is part of the DoD HPC PETTT project and has a wide range of applications where micromechanical properties of polycrystals are studied.

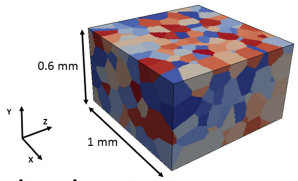

Figure 1: Composite microstructure, composed of individual grains. MASSIF simulates this 3D volume to compute local stress and strain.

In the next section, we briefly describe domain-local strategies to port MASSIF to GPUs[6] under the constraints of GPU memory. However, achieving a high performance implementation of our method is difficult, since some of the operations are not expressible in FFTW. We discuss the use of a new API, FFTX [2], still in early stages of development, to demonstrate its potential in expressing complex data mappings to well-optimized library kernels.

2 BACKGROUND

Our method to port MASSIF to GPUs aims to lower storage and communication requirements by using individual grains in the composite microstructure as domains. However, the local update requires an FFT-based convolution to be computed on each domain, with a non-zero result in the full 3D volume. This exceeds GPU memory capacity for large problem sizes (e.g., $1024 \times 1024 \times 1024$).

SC'19, November 2019, Denver, Colorado USA

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

Hence, we propose a multi-resolution sampling pattern to reduce storage while maintaining the ability to reconstruct solutions with good accuracy. We make use of octrees (or quadtrees for 2D datasets) to not only capture sampling patterns but also make it easier to accumulate results on a distributed system. The construction of the octree is based on sample density of an octree cell as a function of distance from the domain. We use the decay properties of the Green's function convolution kernel and the size of the domain as parameters to construct the octree. Fig. 2 shows the octree-derived sampling pattern after convolving a domain with the Green's function (shown in 2D for simplicity). The sampling rate decreases with distance from the domain due to the dampening effect of the Green's function operator. The work in [6] discusses the slab-pencil domain-local FFT method for the tensor fields involved in MASSIF in more detail. Once all domains are processed, the results must be accumulated locally before proceeding to the next iteration. This is done by an algorithm for multi-resolution interpolation of the octree cells that lie on the support of a domain.

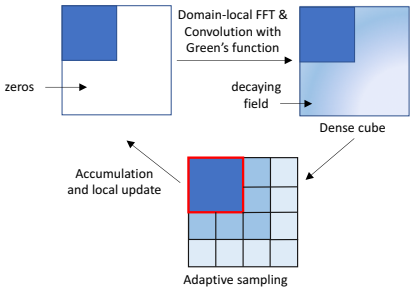

Figure 2: An individual domain convolved with Green's function, resulting in a dense field. Octree-based sampling pattern is used to sample the domain exterior, so that storage on GPU is reduced but reconstruction with good accuracy is possible. The octree cells around the domain have decreased sampling rate, denoted by lighter colors.

Performance models have been used to theoretically predict the performance of our method. However, it is highly difficult to optimize the operations involved in domain-local FFT and tensor contraction coupled with octree-based sampling across various heterogeneous platforms during implementation.

3 ALGORITHM SPECIFICATION WITH FFTX

The FFTX API [3] is designed to provide a familiar interface for expressing complex mappings of multidimensional data to well-optimized FFT-based kernels while a SPIRAL-based code generation back-end [2] handles optimizations across various hardware platforms. The API acts as a domain specific language (DSL) for the code generator, effectively decoupling algorithm specification and code optimization.

The use of a new descriptor derived from octree and quadtree data structures instead of offsets and strides in FFTX data mapping functions can allow the expression of the sampling patterns seen in our algorithm. The quadtree or octree object S will have a stored array of corner locations (x, y, z) of the octree cells, sizes sz and their corresponding sampling rates r. The ith entry in the descriptor is of the form $[(x_i, y_i, z_i), sz_i, r_i]$. S is generated based on known

```
//GPU side, compute on individual domain
#define NUMSUBPLANS 5
plan subplans[NUMSUBPLANS];
plan p; // top-level plan
//... Initialize ....

// create zero-initialized temporary
// n x n x n array with 3 x 3 tensor at each point
tmp1 = create_zero_temp(cube_size, tensor_size);

// copy k x k x k input domain into n x n x n tmp1
subplans[0] = copy_plan(domain, tmp1);// (from, to)

// DFT on the input
tmp2 = create_complex_temp(size_tmp1);
subplans[1] = dft_plan(tmp1);

//Tensor contraction
//In this case we know that output size is the same as tmp2
tmp3 = create_zero_temp(size_tmp2);
subplans[2] = tensor_contraction_plan(tmp2, data, tmp3,
       dimensions_to_contract);//(in,data,out,info)

// iDFT on the contracted output
tmp4 = create_complex_temp(size_tmp3);
subplans[3] = inverse_dft_plan(tmp3, tmp4);

//The next plans apply adaptive sampling
subplans[4] = plan_sample(tmp4, final_output, Octree_S); // (from, to ,
       Octree_descriptor)

// create the top level plan. this copies the sub-plan pointers
p =  plan_compose(NUMSUBPLANS, subplans);

// plan to be used with execute()
return p;
```

Figure 3: FFTX multi-resolution sampling algorithm expressed as a collection of sub-plans.

local domain geometry and Green's function, and used to map sampled points to their respective locations in a dense cube by the code-generation back-end of the library. This representation is better suited for a distributed computing environment, since sub-regions of varying size and resolutions can be located in memory using addresses inferred from the descriptor S. An example FFTX program written using API calls is as shown in Fig. 3. For the accumulation algorithm, an FFTX multi-resolution interpolation plan can copy out required samples and perform accumulation after linear interpolation.

Other examples. Examples of other scientific codes requiring pruning for use with GPUs include Poisson's equation solvers [8], Maxwell's equations using pseudo-spectral methods [12],[1]. FFTX can play a crucial role in these and similar PDE-based simulations that extensively use FFTs. Additionally, there are opportunities for innovative simulation algorithm design on clusters like Summit with IBM POWER9 CPUs, GPUs connected by NVLINK and unified memory.

4 CONCLUSIONS & FUTURE WORK

Algorithmic strategies such as domain-local adaptive sampling or pruning enable us to simulate larger problem sizes than previously possible, thus leading to new insights in exploratory scientific computing. We suggest the use of an API like FFTX to express algorithm specifications such as complex data mappings required to manipulate large-memory problems in GPU environments. The FFTX programs in this paper demonstrate effectiveness of the API in abstracting code optimization away from the user, while a code-generation back-end such as SPIRAL can be used to obtain high performance on various hardware platforms.

REFERENCES

[1] A. Canning. 2008. Scalable Parallel 3d FFTs for Electronic Structure Codes. In *High Performance Computing for Computational Science - VECPAR 2008*, José M. Laginha M. Palma, Patrick R. Amestoy, Michel Daydé, Marta Mattoso, and João Correia Lopes (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 280–286.

[2] F. Franchetti, T. M. Low, D. T. Popovici, R. M. Veras, D. G. Spampinato, J. R. Johnson, M. PÃijschel, J. C. Hoe, and J. M. F. Moura. 2018. SPIRAL: Extreme Performance Portability. *Proc. IEEE* 106, 11 (Nov 2018), 1935–1968. https://doi.org/10.1109/JPROC.2018.2873289

[3] F. Franchetti, D. G. Spampinato, A. Kulkarni, T. Popovici, T. M. Low, M. Franusich, A. Canning, P. McCorquodale, B. Van Straalen, and P. Colella. 2018. FFTX and SpectralPack: A First Look. In *IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC)*.

[4] M. Frigo and S. G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (Feb 2005), 216–231. https://doi.org/10.1109/JPROC.2004.840301

[5] Intel. [n. d.]. Math Kernel Library. ([n. d.]). software.intel.com/mkl.

[6] A. Kulkarni, F. Franchetti, and J. Kovačević. 2018. Large-Scale Algorithm Design for Parallel FFT-based Simulations on GPUs. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 301–305. https://doi.org/10.1109/GlobalSIP.2018.8646675

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

[8] Peter McCorquodale, Phillip Colella, Gregory T. Balls, and Scott B. Baden. 2006. A Local Corrections Algorithm for Solving Poisson's Equation inThree Dimensions. 2 (10 2006).

[9] J.C. Michel, H Moulinec, and Pierre Suquet. 2000. A computational method based on augmented Lagrangians and fast Fourier Transforms for composites with high contrast. *CMES - Computer Modeling in Engineering and Sciences* 1 (01 2000), 79–88.

[10] 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.

[11] V. Tari, R. A. Lebensohn, R. Pokharel, T. J. Turner, P. A. Shade, J. V. Bernier, and A. D. Rollett. 2018. Validation of micro-mechanical FFT-based simulations using High Energy Diffraction Microscopy on Ti-7Al. *Acta Materialia* 154 (8 2018). https://doi.org/10.1016/j.actamat.2018.05.036

[12] J.-L. Vay, A. Almgren, J. Bell, L. Ge, D.P. Grote, M. Hogan, O. Kononenko, R. Lehe, A. Myers, C. Ng, J. Park, R. Ryne, O. Shapoval, M. Thevenet, and W. Zhang. 2018. Warp-X: A new exascale computing platform for beam plasma simulations. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* (2018). https://doi.org/10.1016/j.nima.2018.01.035