Massive Scaling of MASSIF: Algorithm Development for Hooke's Law Simulations on Distributed GPU systems

Anuva Kulkarni[1], Jelena Kovačević[2], Franz Franchetti[1]
[1]Carnegie Mellon University
[2]NYU Tandon School of Engineering

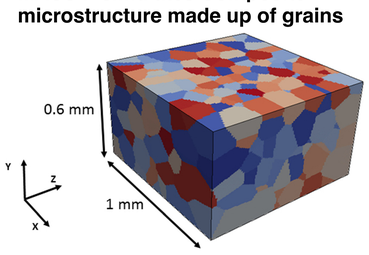
GPUs and the need for algorithm re-design

Motivation
Common characteristics of scientific codes:
- FFT-based simulations involve all-to-all communication
- High memory requirement

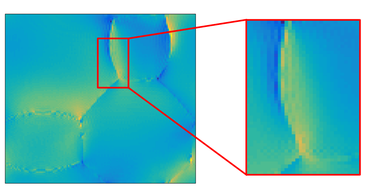
Incompatibility with GPUs:
- GPUs haves small on-chip memory (~16GB)
- Various communication latencies

Case study: MASSIF
- Partial Differential Equation solved by Green's function method
- FFT-based convolution and tensor contraction between rank-2 tensors and rank-4 Green's
- function

MASSIF simulates Composite microstructure made up of grains



Stress field at grain boundary

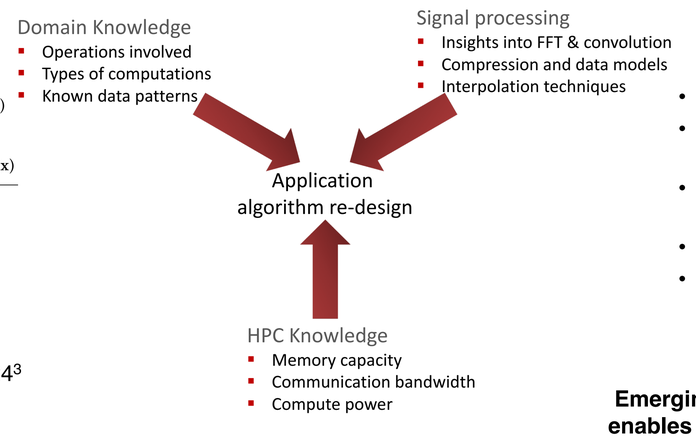


Algorithm 1 MASSIF Inner loop

1: **Initialize:**
$\epsilon^0 \leftarrow E, \quad \sigma^0_{mn}(\mathbf{x}) \leftarrow C_{mnk\ell}(\mathbf{x}) : \epsilon^0_{k\ell}(\mathbf{x})$
2: **while** $e_s > e_{\text{tol}}$ **do**
3: $\hat{\sigma}^{(i)}_{mn}(\boldsymbol{\xi}) \leftarrow \text{FFT}(\sigma^{(i)}_{mn}(\mathbf{x}))$
4: Check convergence
5: $\Delta\hat{\epsilon}^{(i+1)}_{k\ell}(\boldsymbol{\xi}) \leftarrow \hat{\Gamma}_{k\ell mn}(\boldsymbol{\xi}) : \hat{\sigma}^{(i)}_{mn}(\boldsymbol{\xi})$
6: Update strain: $\hat{\epsilon}^{(i+1)}_{k\ell}(\boldsymbol{\xi}) \leftarrow \hat{\epsilon}^{(i)}_{k\ell}(\boldsymbol{\xi}) - \Delta\hat{\epsilon}^{(i+1)}_{k\ell}(\boldsymbol{\xi})$
7: $\epsilon^{(i+1)}_{k\ell}(\mathbf{x}) \leftarrow \text{iFFT}(\hat{\epsilon}^{(i+1)}_{k\ell}(\boldsymbol{\xi}))$
8: Update stress: $\sigma^{(i+1)}_{mn}(\mathbf{x}) \leftarrow C_{mnk\ell}(\mathbf{x}) : \epsilon^{(i+1)}_{k\ell}(\mathbf{x})$

Main issues faced:
As problem size increases for high resolution simulations, storage and communication requirements increase. Problems larger than 1024^3 grids not currently simulated.

Proposed Solution:
Re-design the application algorithm using domain expertise and knowledge about high performance computing platforms.

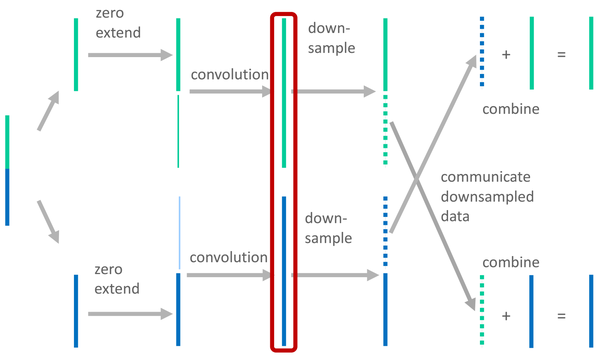


Challenges with FFTW

- **FFTW is de-facto standard interface for FFT**
- **Vendor libraries support the FFTW 3.X interface:**
  Intel MKL, IBM ESSL, AMD ACML (end-of-life), Nvidia cuFFT, Cray LibSci/CRAFFT
- **Some Issues:**
- No native support for accelerators (GPUs, Xeon PHI, FPGAs) and SIMT
- Parallel/MPI version does not scale beyond 32 nodes
- No analogue to LAPACK for spectral method
- Complex data patterns may need to be expressed, FFTW currently falls short. But, extensions like FFTX could add new descriptors.

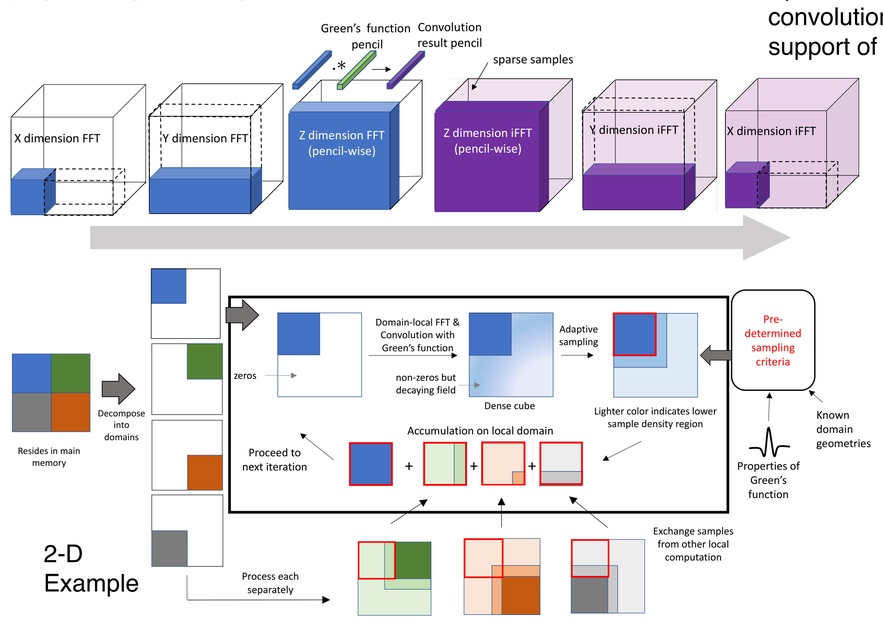
Emerging interfaces like FFTX, extension of FFTW, enables algorithm specification as composition of sub-plans

Front end: Algorithm Specification

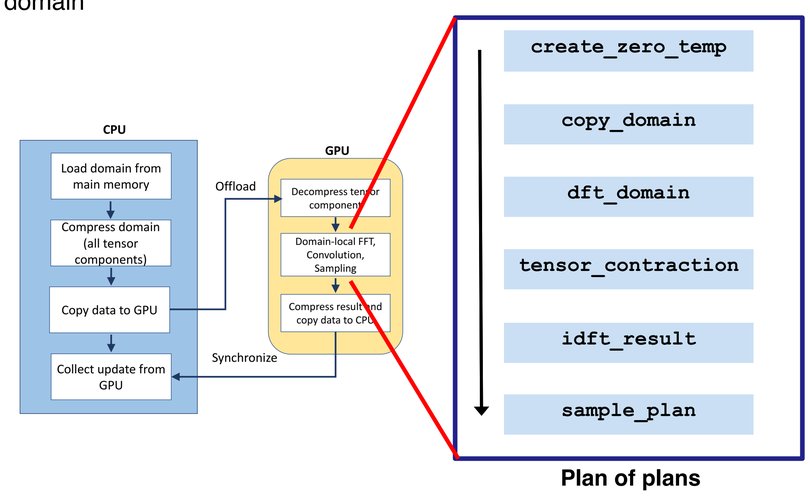


Key Idea: Approximate Convolution.
Exploit locality arising due to domain decomposition + Properties of Green's function.

Sparse sampling of convolution result outside the support of the domain



Making algorithm specification easier for the user

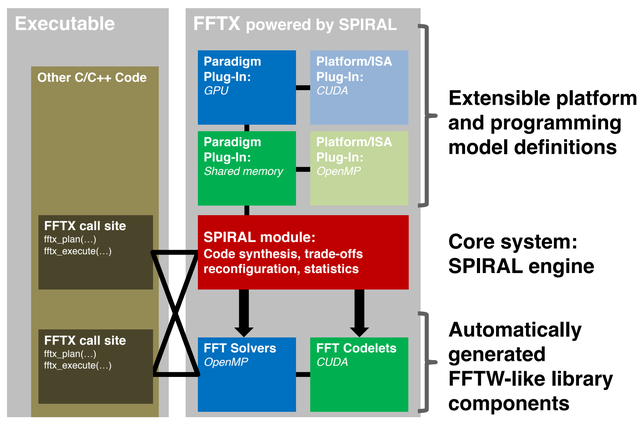


Goal: Domain decomposition and sampling expressed easily using an FFTW-like interface

Back end: Code Optimization

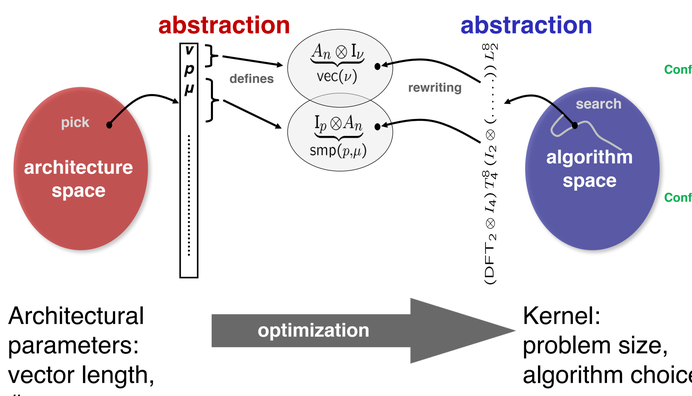
FFTX is..
- **Modernized FFTW-style interface**
- **Backwards compatible to FFTW 2.X and 3.X**
- **Small number of new features, familiar interface**
- **Code generation backend using SPIRAL**
- **Compilation and advanced performance optimization** cross-call and cross library optimization, accelerator off-loading,…

FFTX backend: SPIRAL

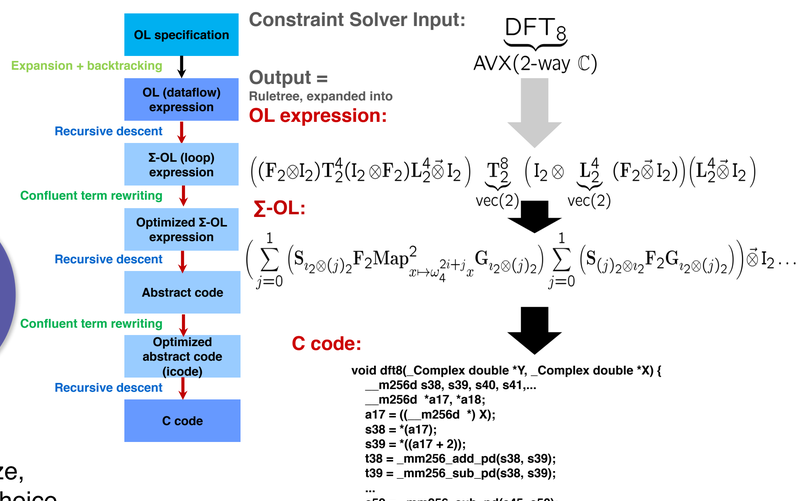


DARPA BRASS

Platform-aware formal program synthesis



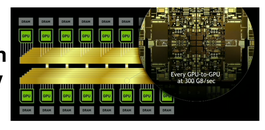
Translating an OL expression into code



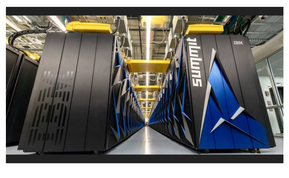
First-order Performance Analysis

Single node, multi-GPU (HPE Apollo 6500 node)

DGX2 Workstation with 16 GPUs connected by NVLINK

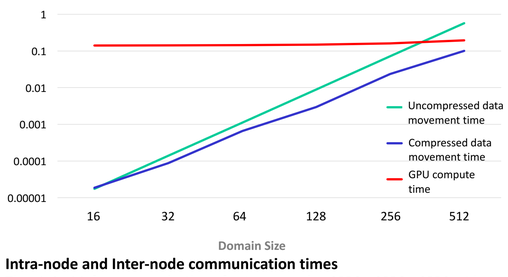


Summit supercomputer: many-node, with multiple GPUs/node.

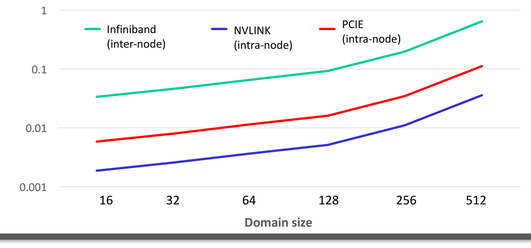


Algorithm design ⟶ ⟵ Performance modeling

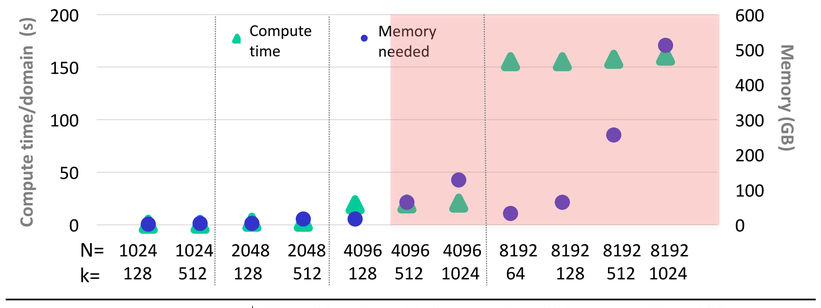
Comparison of communication time and compute time
Time in log(s) for problem size of 1024 x 1024 x 1024 Nvidia Tesla V100



Intra-node and Inter-node communication times
Time in log(s) for communication of octree samples, 1024 x 1024 x 1024 grid



GPU compute time & Memory requirement



Machine	k →	N = 1024		N = 2048		N = 4096			N = 8192			
		128	512	128	512	128	512	1024	64	128	512	1024
HPE Apollo 6500 node		✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
DGX2		✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Summit (ORNL)		✓	✓	✓	✓	✓	(✓)	(✓)	✓	(✓)	✗	✗

Memory requirement exceeds available GPU memory

Future work: FFTX and SpectralPACK

Numerical Linear Algebra
LAPACK
LU factorization
Eigensolves
SVD
…

BLAS
BLAS-1
BLAS-2
BLAS-3

Spectral Algorithms
SpectralPACK
Convolution
Correlation
Upsampling
Poisson solver
…

FFTX
DFT, RDFT
1D, 2D, 3D,…
batch

- LAPACK for spectral algorithms
- Define FFTX as the analogue to BLAS
- Define class of numerical algorithms to be supported by SpectralPACK
- Define SpectralPACK functions

FFTX project:

Carnegie Mellon

SpiralGen Accelerating Innovation in Computing

ECP EXASCALE COMPUTING PROJECT

References

[1]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 Di raction Microscopy on Ti-7Al. Acta Materialia 154 (8 2018). https://doi.org/10.1016/j.actamat.2018.05.036
[2] 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
[3] 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
[4]F. Franchetti, et al., 2018. FFTX and SpectralPack: A First Look. In IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC).
[5]P. McCorquodale,P. Colella,G. T.Balls,and S.B.Baden.2006.A Local Corrections Algorithm for Solving Poisson's Equation inThree Dimensions.
2 (10 2006).
Images credit: NVIDIA, ORNL website, SpiralGen.