

## Porting Scientific codes to GPUs

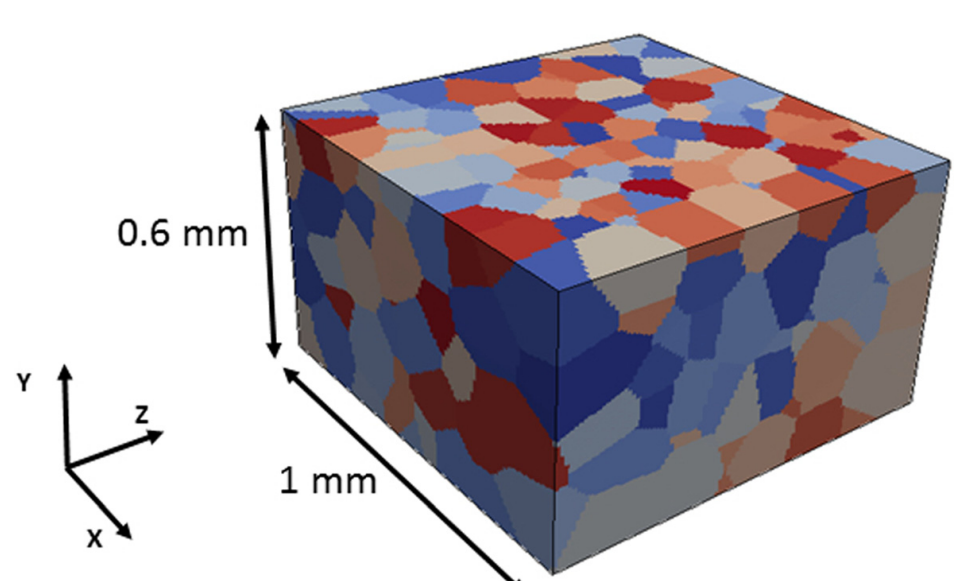
- Common characteristics of scientific codes:
- Usually in Fortran
  - FFT-based simulations involve all-to-all communication
  - High memory requirement

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

- Solutions for porting code:
- Domain decomposition (regular or irregular)
  - Exploit properties of data and convolution kernel
  - Sampling/pruning used so that domain results fit on GPU memory

- Case study: MASSIF
- Hooke's law simulation
  - 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

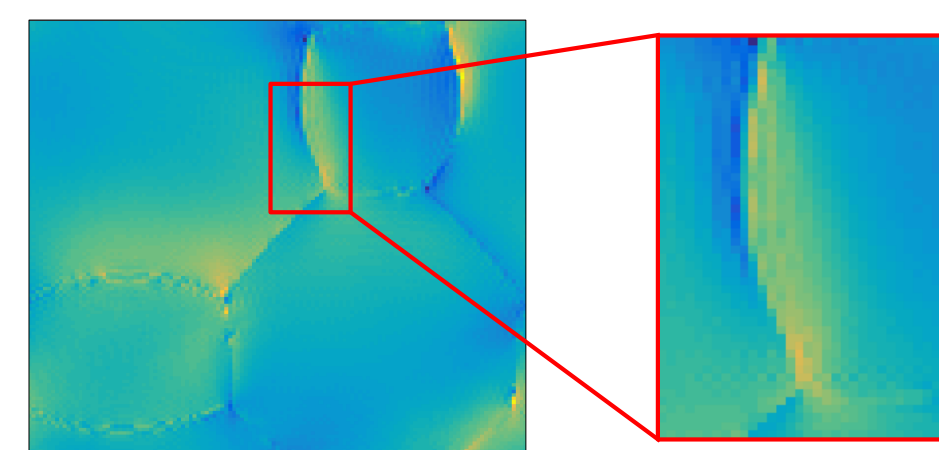
MASSIF simulates Composite microstructure made up of grains



```

Algorithm 1 MASSIF Inner loop
1: Initialize:
    $\epsilon^0 \leftarrow E; \sigma_{mn}^0(x) \leftarrow C_{mnkl}(x) : \epsilon_{kl}^0(x)$ 
2: while  $\epsilon_s > \epsilon_{tol}$  do
3:    $\hat{\sigma}_{mn}^{(i)} \leftarrow \text{FFT}(\sigma_{mn}^{(i-1)}(x))$ 
4:   Check convergence
5:    $\Delta \hat{\epsilon}_{kl}^{(i+1)} \leftarrow \hat{\Gamma}_{klmn}(\hat{\sigma}_{mn}^{(i)}(\xi))$ 
6:   Update strain:  $\hat{\epsilon}_{kl}^{(i+1)}(\xi) \leftarrow \hat{\epsilon}_{kl}^{(i)}(\xi) - \Delta \hat{\epsilon}_{kl}^{(i+1)}(\xi)$ 
7:    $\hat{\epsilon}_{kl}^{(i+1)}(x) \leftarrow \text{iFFT}(\hat{\epsilon}_{kl}^{(i+1)}(\xi))$ 
8:   Update stress:  $\sigma_{mn}^{(i+1)}(x) \leftarrow C_{mnkl}(x) : \hat{\epsilon}_{kl}^{(i+1)}(x)$ 
  
```

Stress field at grain boundary



- Proposed solution:
- Domain decomposition with grains are domains
  - Domain-local FFT followed by convolution and tensor contraction
  - Green's function computed on-the-fly to avoid storage
  - Adaptive sampling of dense convolution result to fit problem on GPU memory

Complex data mappings! How to get maximum performance on various platforms?

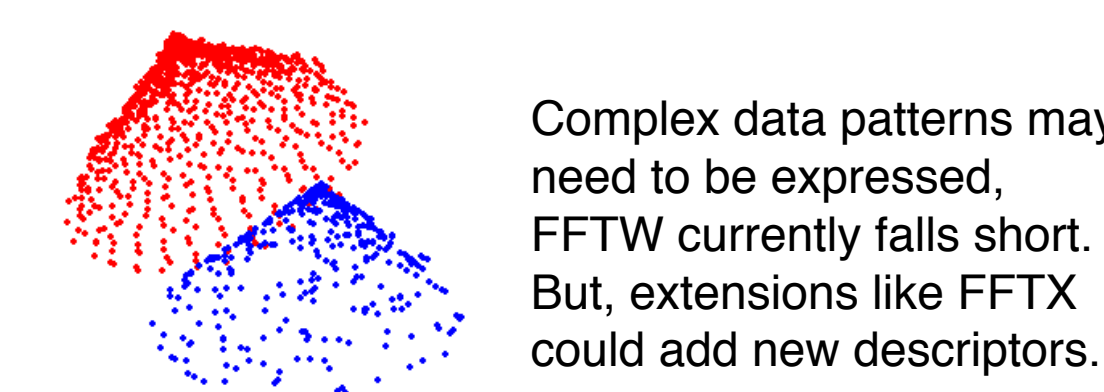
DGX-2 Source: NVIDIA



Summit Source: ORNL

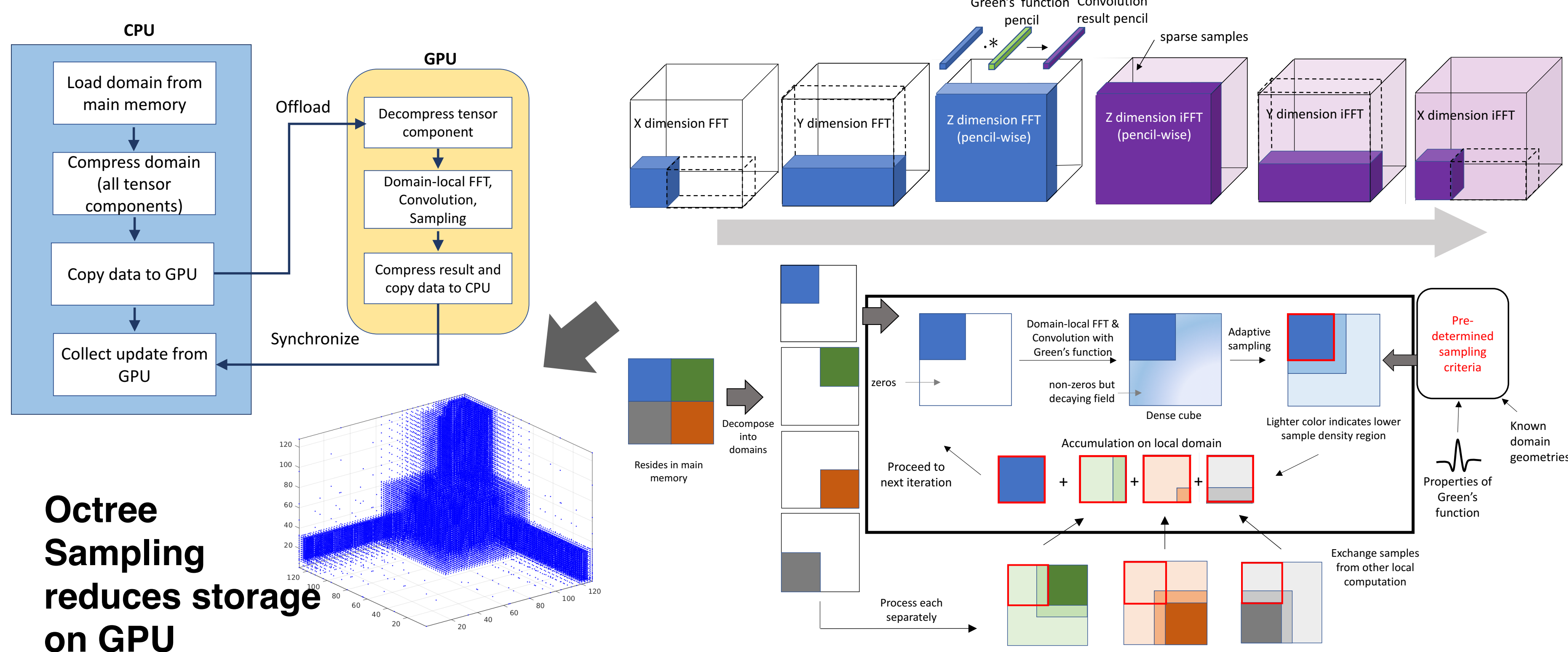
## Challenges FFTW

- 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.

## Front end: Algorithm Specification



## FFT, tensor contraction and sampling

```

//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
tmp = create_zero_temp(cube_size, tensor_size);

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

// DFT on the input
tmp2 = create_complex_temp(size,tmp);
subplans[1] = dft_plan(tmp);

// 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;
  
```

## Accumulation

```

//CPU side, accumulate over all domains
#define NUMSUBPLANS 3
plan subplans[NUMSUBPLANS];
plan accum; // top-level accumulate plan

// n x n x n array with 3 x 3 tensor at each point
tmp = create_zero_temp(cube_size, tensor_size);

// smaller temp arrays
output_cube = create_zero_temp(domain_d_size, tensor_size);
net_output_cube = create_zero_temp(domain_d_size, tensor_size);

for j in [1,...,D] except d:
subplans[0] = plan_decode_octree(S[j], data_array, tmp); //decode octree.
copy into tmp

subplans[1] = plan_multires_interpolate(S[j], tmp, domain_d, output_cube,
output_size); //descriptor, input cube (samples missing), filter (only interpolate that
region), outputcube, outputsize

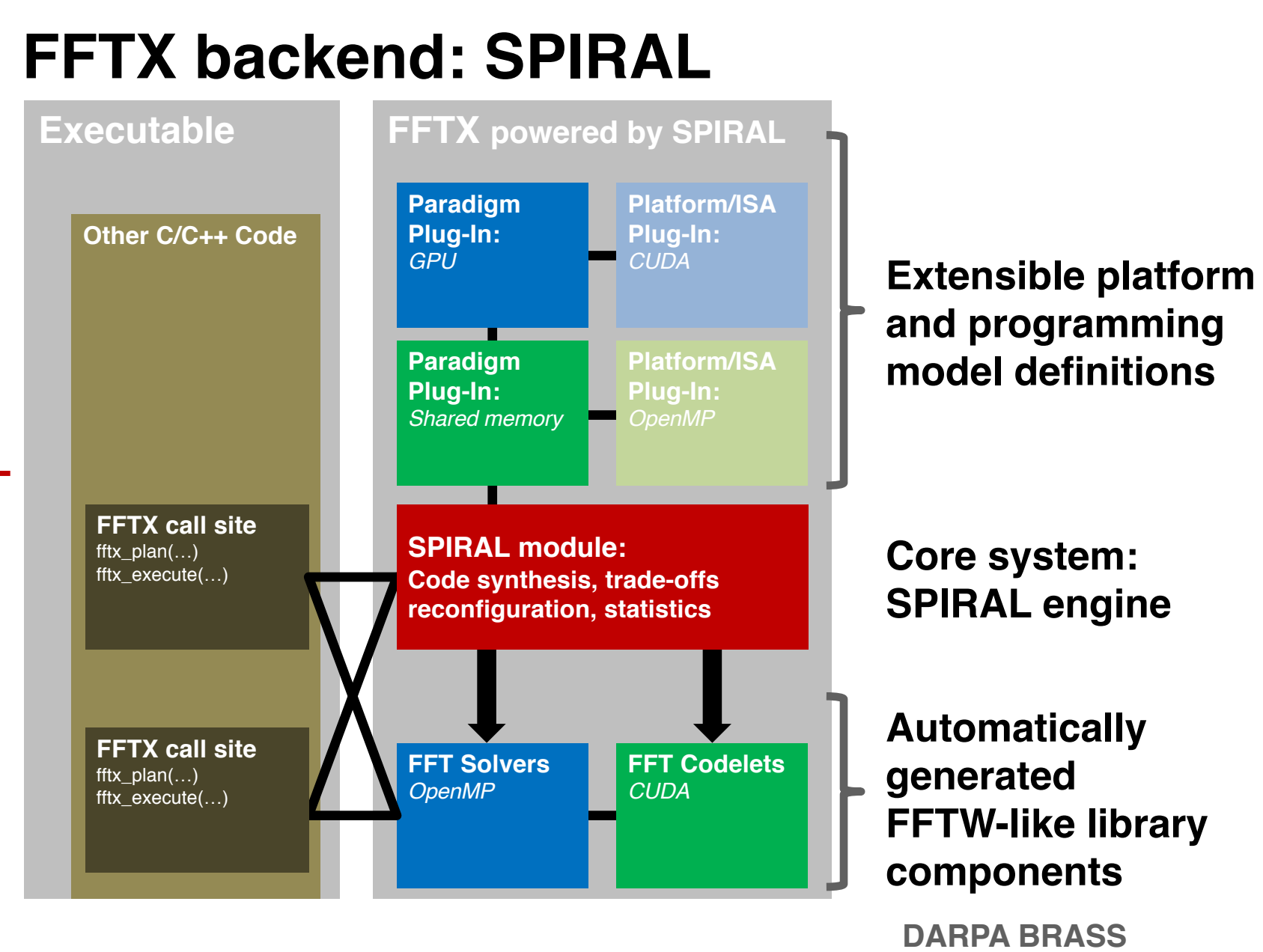
subplan[2] = plan_sum(output_cube, net_output_cube);

// create the top level plan
accum = plan_compose(NUMSUBPLANS, subplans);

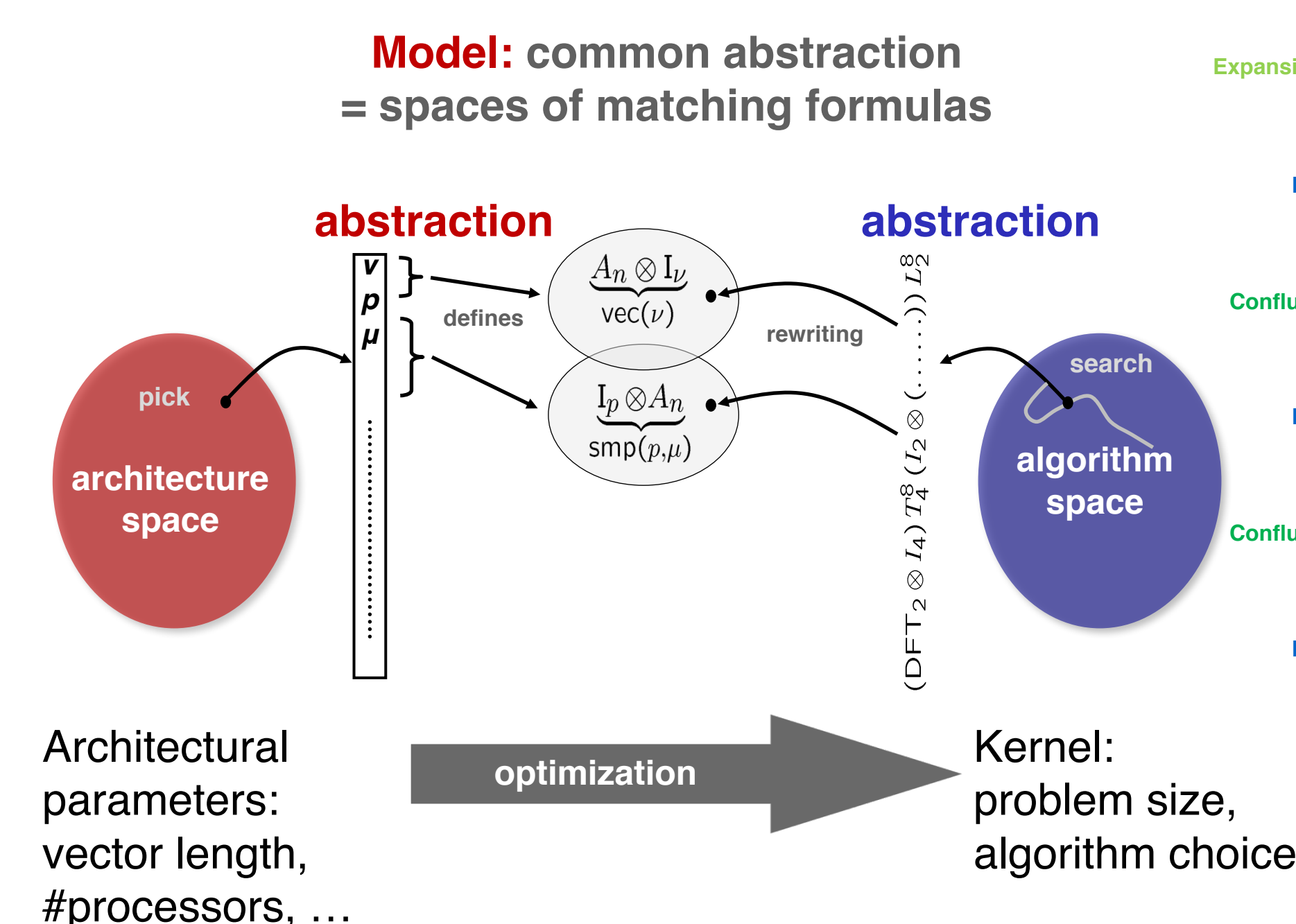
// plan to be used with execute()
return accum;
  
```

## Back end: Code Optimization

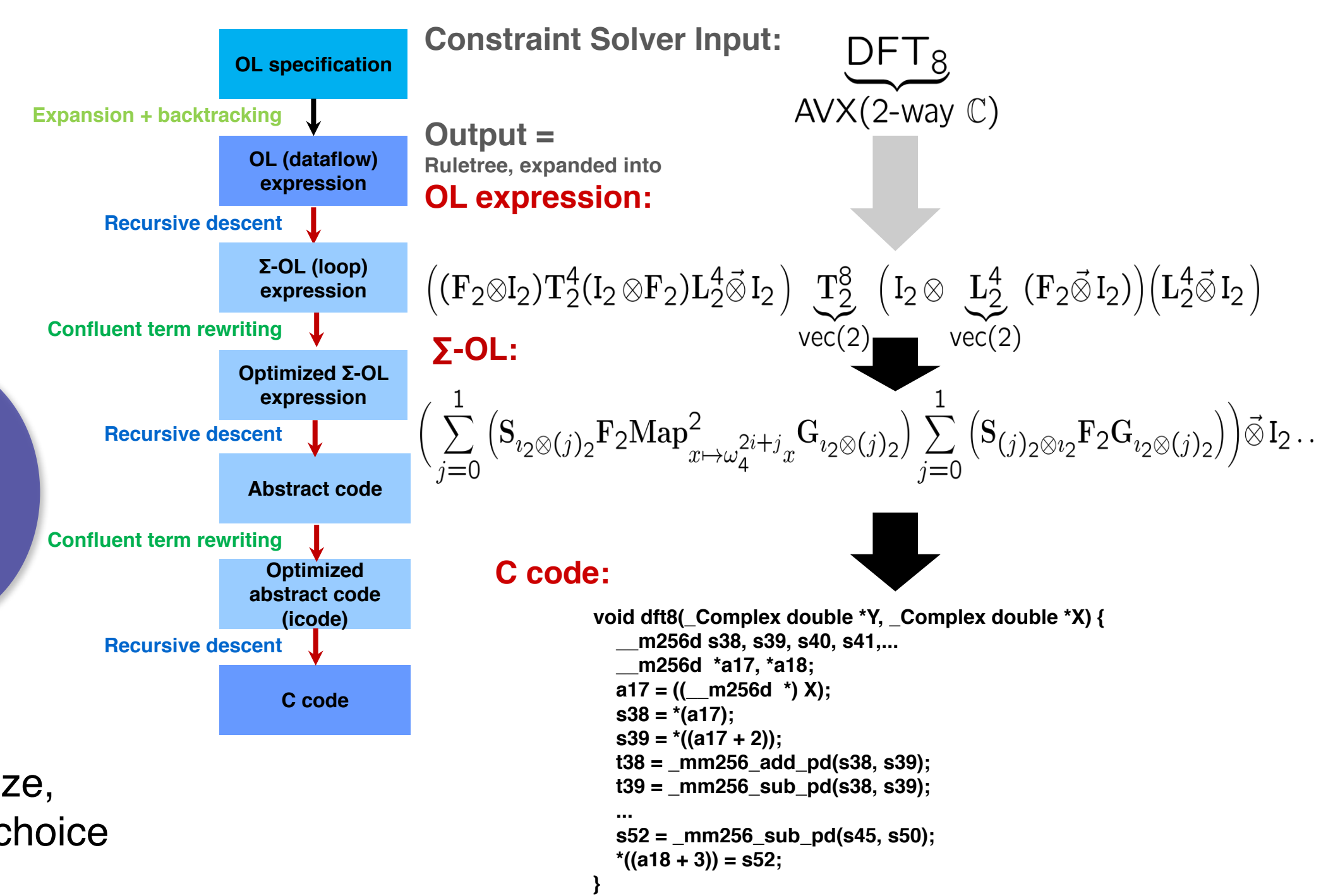
- FFTX is..
- Modernized FFTW-style interface
  - Backwards compatible to FFTW 2.X and 3.X
  - Small number of new features, familiar interface
- Code generation backend using SPIRAL
- Library/application kernels are interpreted as specifications in DSL extract semantics from source code and known library semantics
  - Compilation and advanced performance optimization cross-call and cross library optimization, accelerator off-loading,...
  - Reference library implementation and bindings to vendor libraries



## Platform-aware formal program synthesis



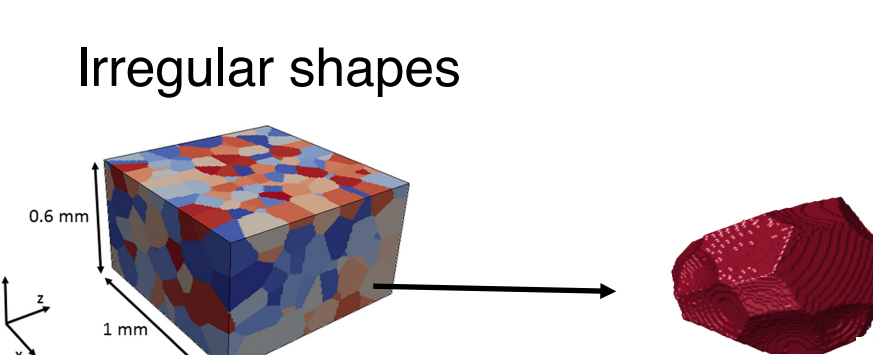
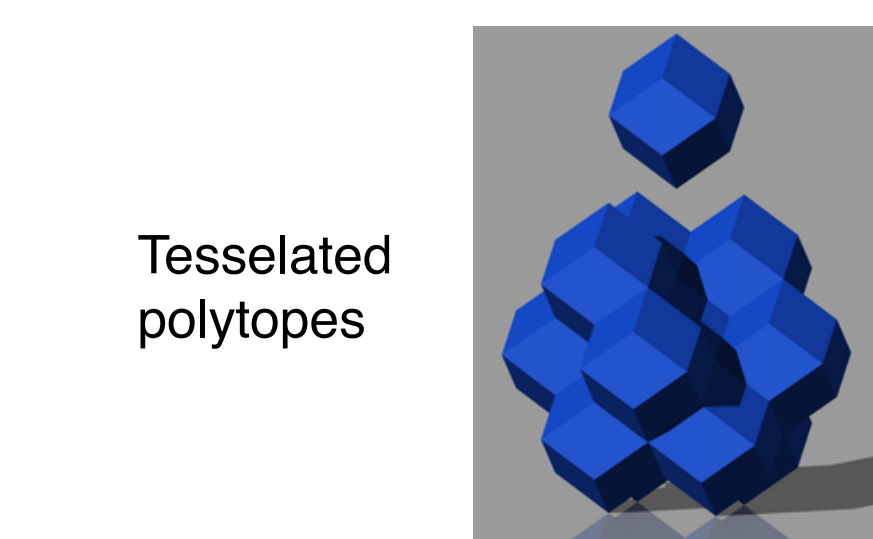
## Translating an OL expression into code



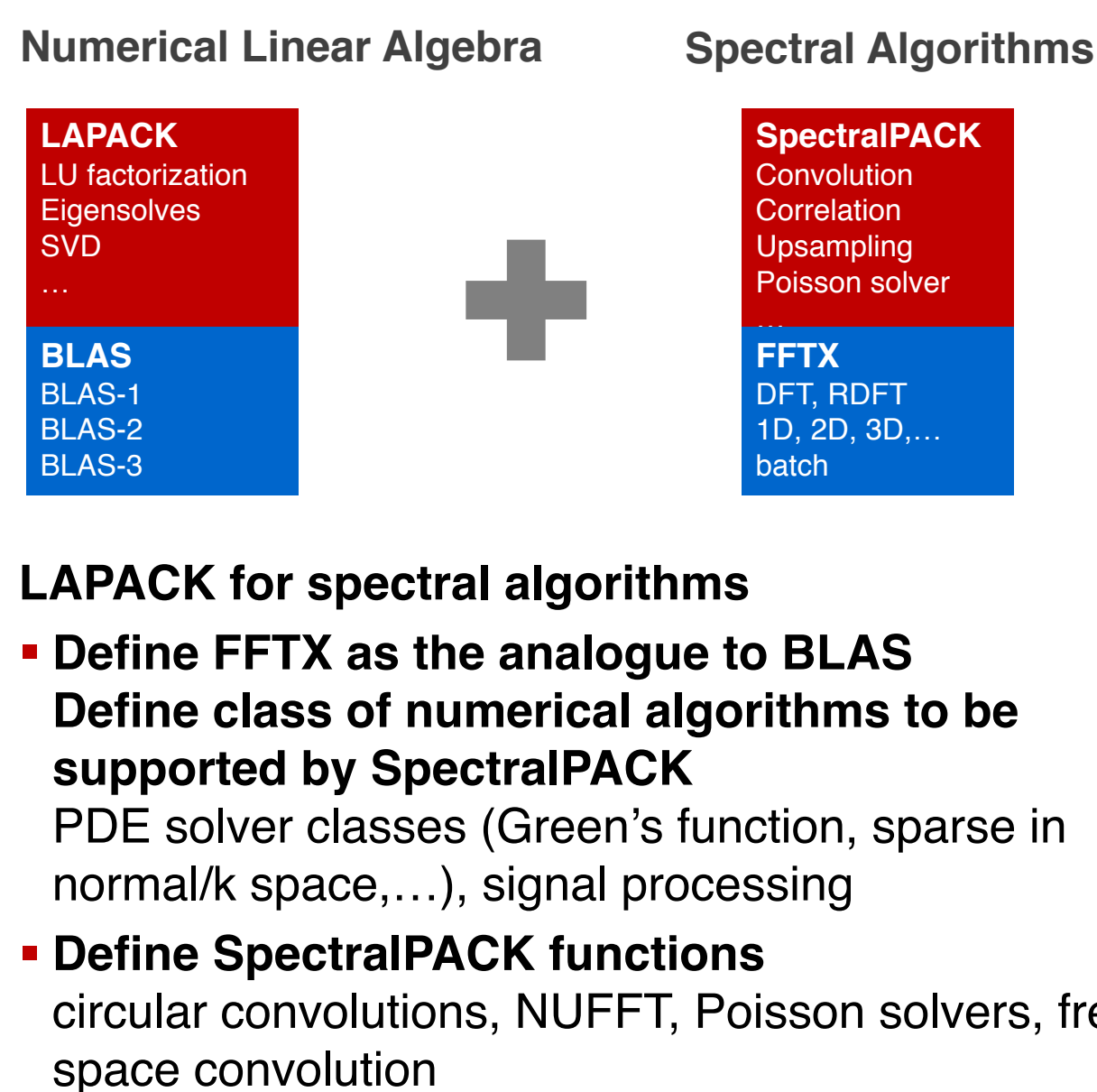
## Future Plans + Other Applications

### Future work: MASSIF

- Irregular domain decomposition
- Extension of adaptive sampling for irregular domains



### Future work: FFTX and SpectralPACK



### Poisson's equation in free space

Partial differential equation (PDE) Solution

$$\Delta(\Phi) = \rho \quad \Phi: \mathbb{R}^3 \rightarrow \mathbb{R}$$

$$\rho: \mathbb{R}^3 \rightarrow \mathbb{R}$$

$$D = \text{supp}(\rho) \subset \mathbb{R}^3$$

Poisson's equation.  $\Delta$  is the Laplace operator

$$\Phi(\vec{x}) = \frac{Q}{4\pi|\vec{x}|} + o\left(\frac{1}{|\vec{x}|}\right) \text{ as } |\vec{x}| \rightarrow \infty$$

$$Q = \int_D \rho d\vec{x}$$

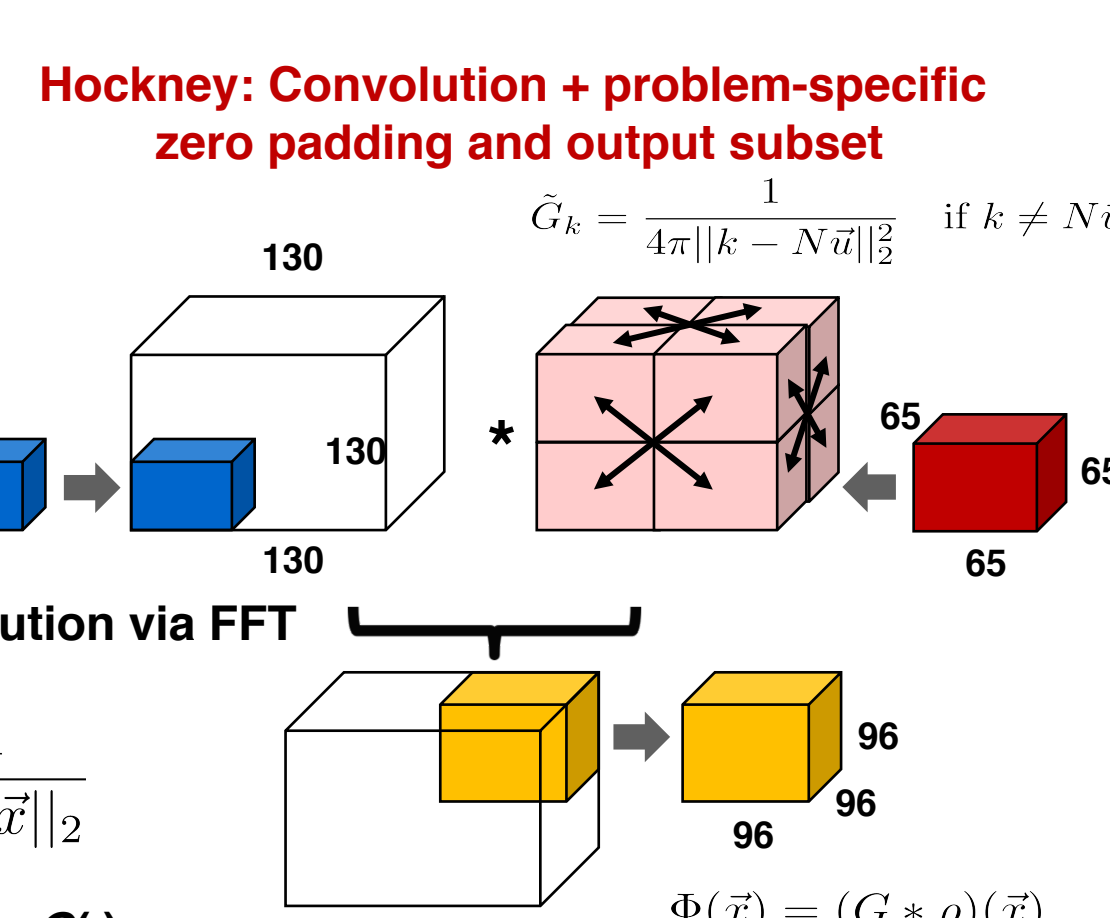
Approach: Green's function

$$\Phi(\vec{x}) = \int_D G(\vec{x} - \vec{y}) \rho(\vec{y}) d\vec{y} \equiv (G * \rho)(\vec{x}), \quad G(\vec{x}) = \frac{1}{4\pi|\vec{x}|_2}$$

Solution:  $\Phi(\cdot) = \text{convolution of RHS } \rho(\cdot) \text{ with Green's function } G(\cdot)$ . Efficient through FFTs (frequency domain)

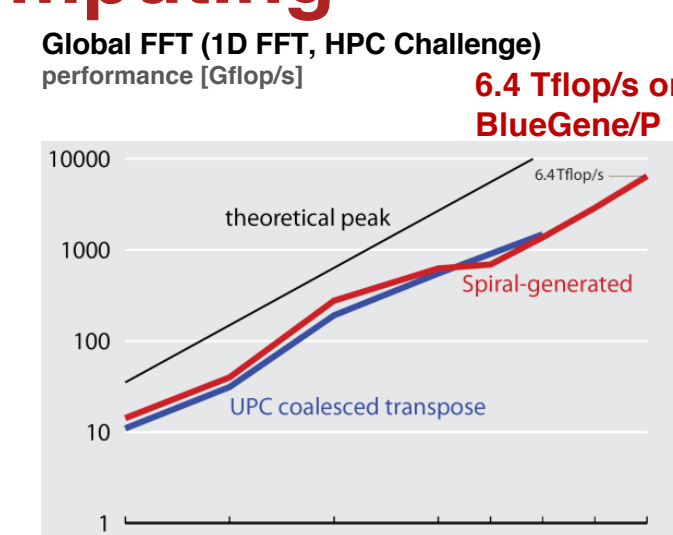
$$\hat{G}_k = \frac{1}{4\pi|k - N\vec{u}|_2^2} \text{ if } k \neq N\vec{u} \quad \text{Green's function kernel in frequency domain}$$

### Hockney free-space convolution



### SPIRAL: success in HPC/supercomputing

- NCSA Blue Waters PAID Program, FFTs for Blue Waters
  - RIKEN K computer FFTs for the HPC-ACE ISA
  - LANL RoadRunner FFTs for the Cell processor
  - PSC/XSEDE Bridges Large size FFTs
  - LLNL BlueGene/L and P FFTW for BlueGene/L's Double FPU
  - ANL BlueGene/Q Mira Early Science Program, FFTW for BGG QPX
- 2006 Gordon Bell Prize (Peak Performance Award) with LLNL and IBM
- 2010 HPC Challenge Class II Award (Most Productive System) with ANL and IBM



BlueGene/P at Argonne National Laboratory 128k cores (quad-core CPUs) at 850 MHz

## 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-FAL. *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. Kovacevic. 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 (HPCDA).

[5] J.P. McCorquodale, F. Colella, G. T. Bates, and S.B. Baden. 2006. A Local Corrections Algorithm for Solving Poisson's Equation in Three Dimensions. 2 (10 2006).