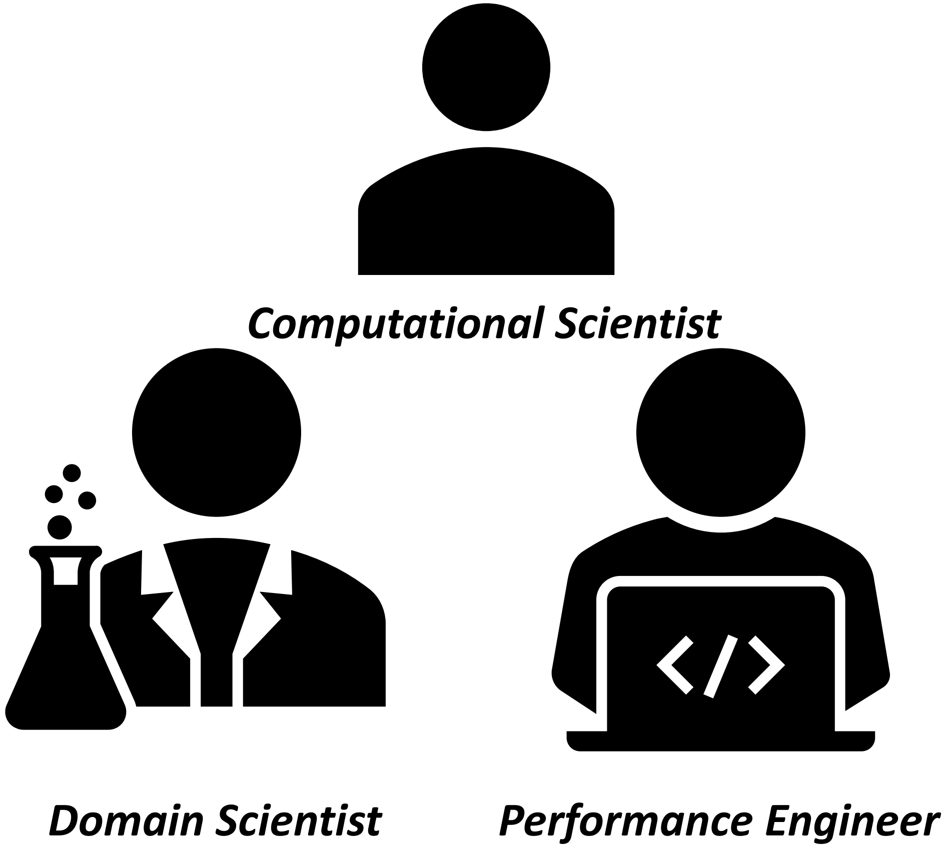
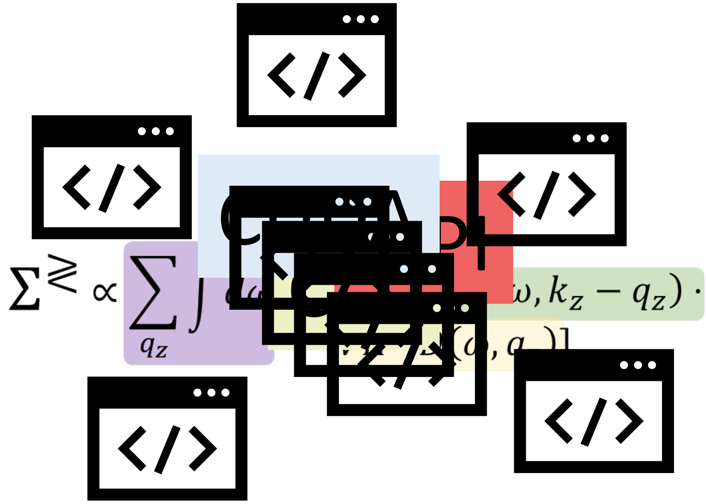
A. N. Ziogas, Tal Ben-Nun, G. Fernandéz, T. Schneider, M. Luisier, T. Hoefler

Optimizing the Data Movement in Quantum Transport Simulations via Data-Centric Parallel Programming

Motivation for a Data-Centric Framework





Data-Centric Framework

Domain Scientist

Problem Formulation

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

Python / numpy

DSLs

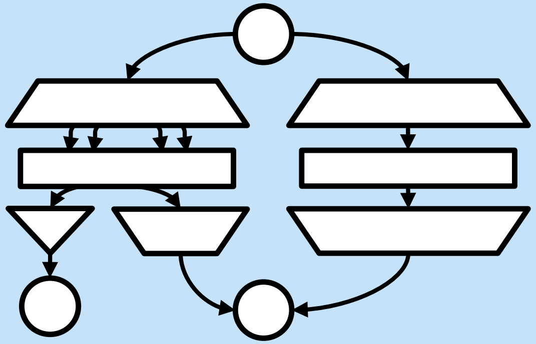
TensorFlow

MATLAB

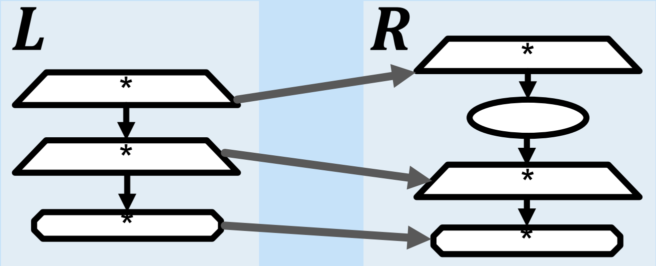
SDFG Builder API

High-Level Program

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Graph Transformations (API, Interactive)

Transformed Dataflow

Performance Results

System

Hardware Information

SDFG Compiler

Runtime

CPU Binary

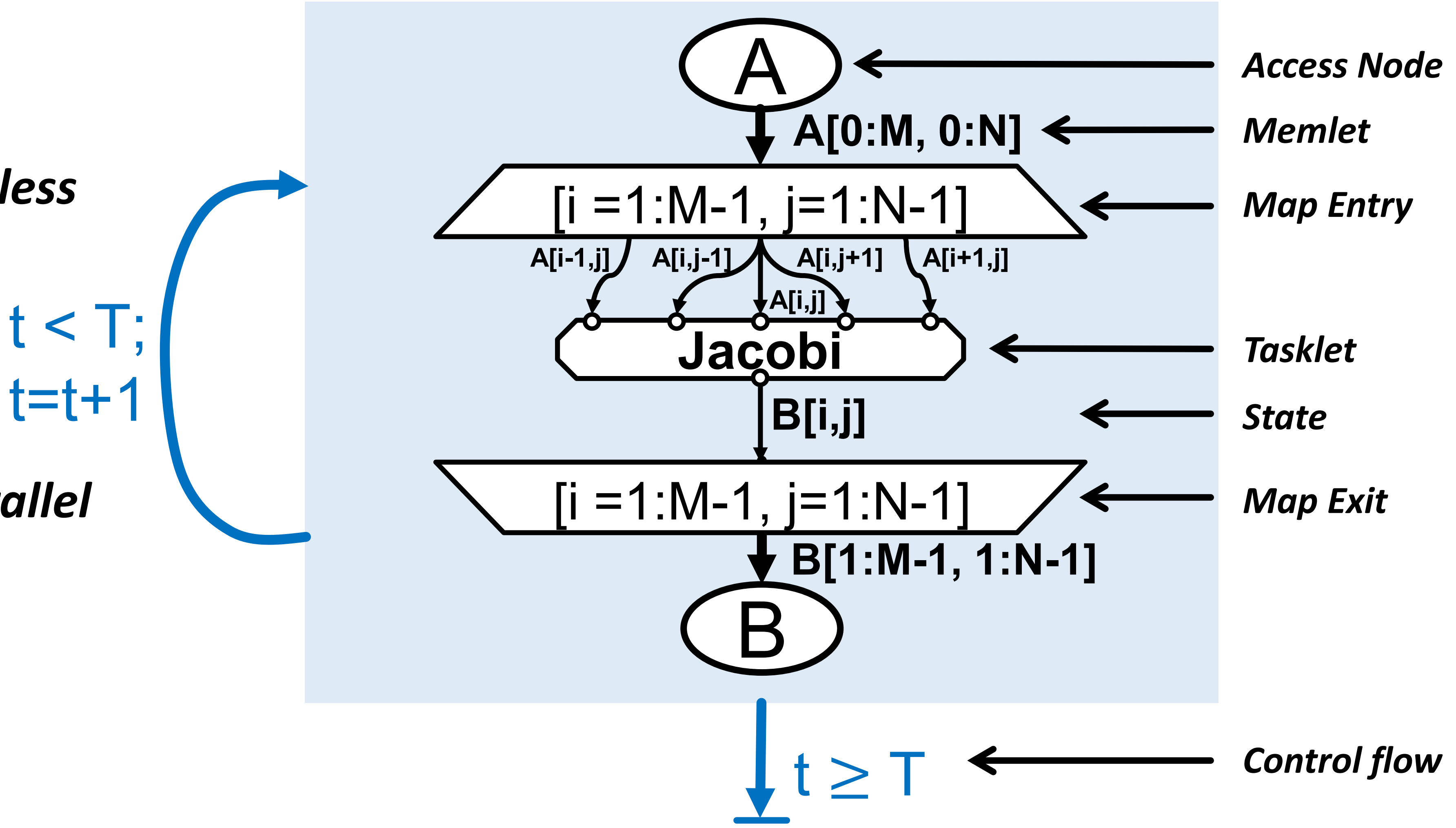
GPU Binary

FPGA Modules

Thin Runtime Infrastructure

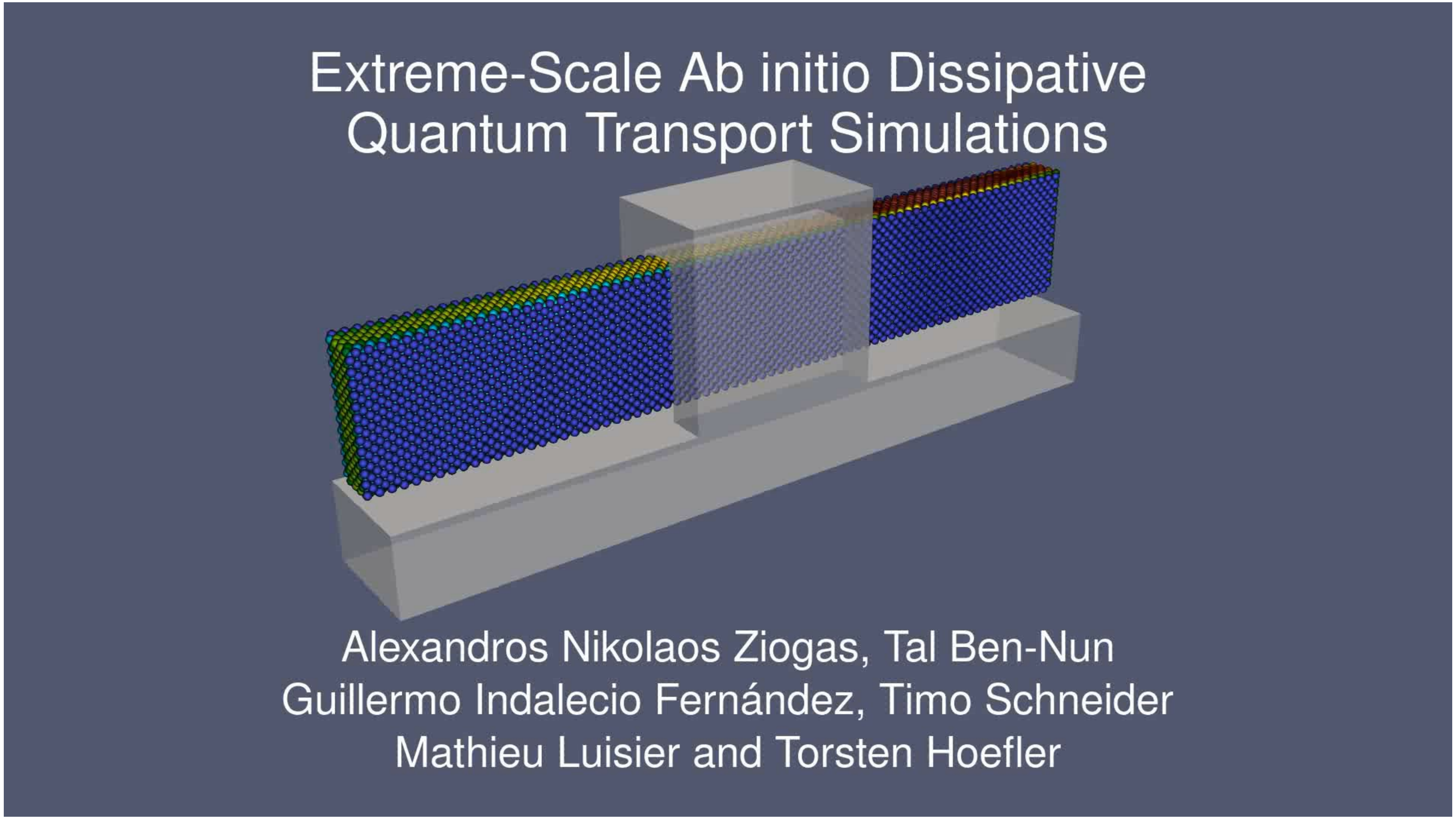
Basic Elements of the Data-Centric IR

Data
Fine-grained stateless computations
Dataflow
Control-flow
Abstraction of independently parallel computations

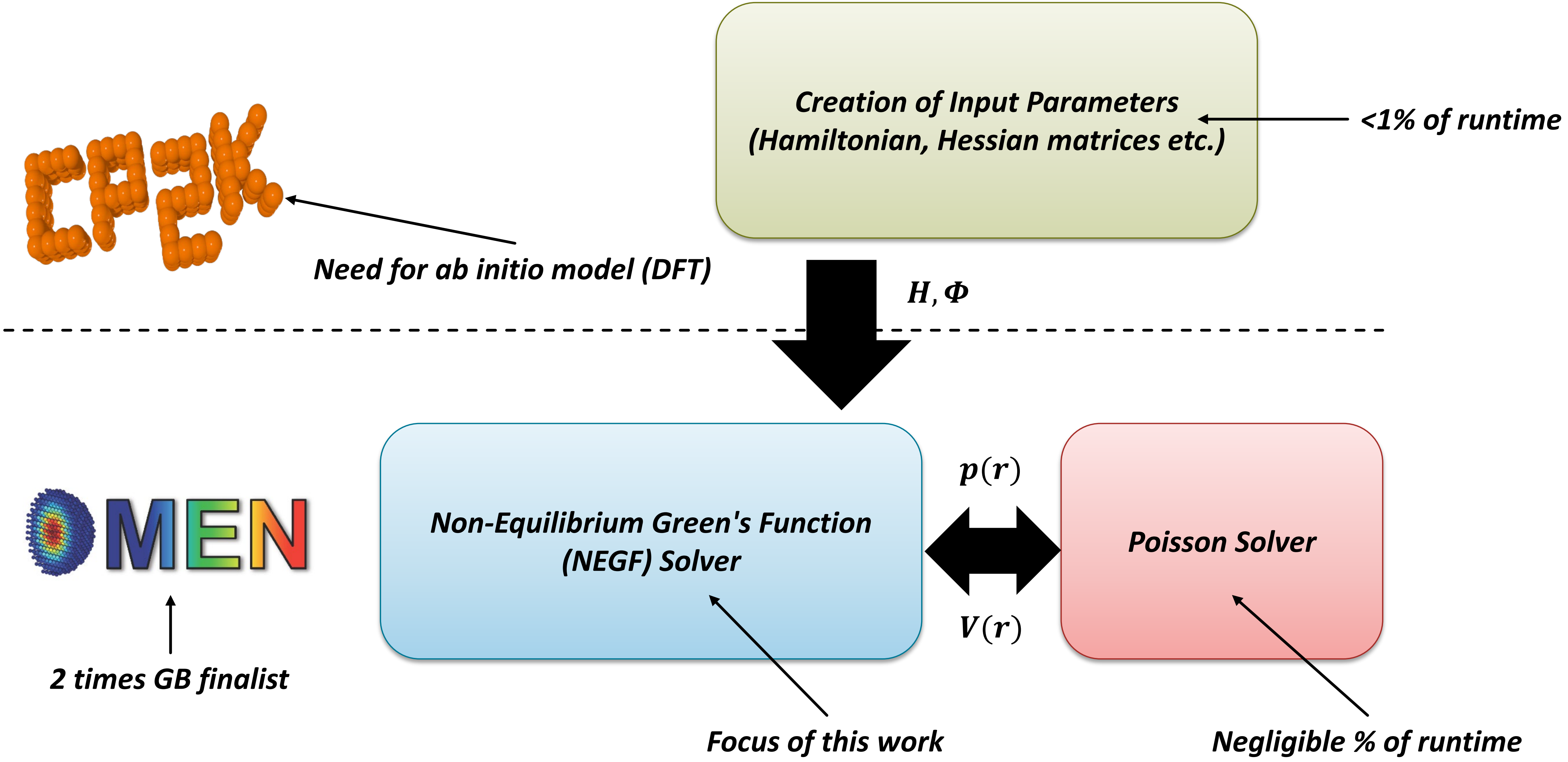


Ben-Nun et al. "Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures", SC19.

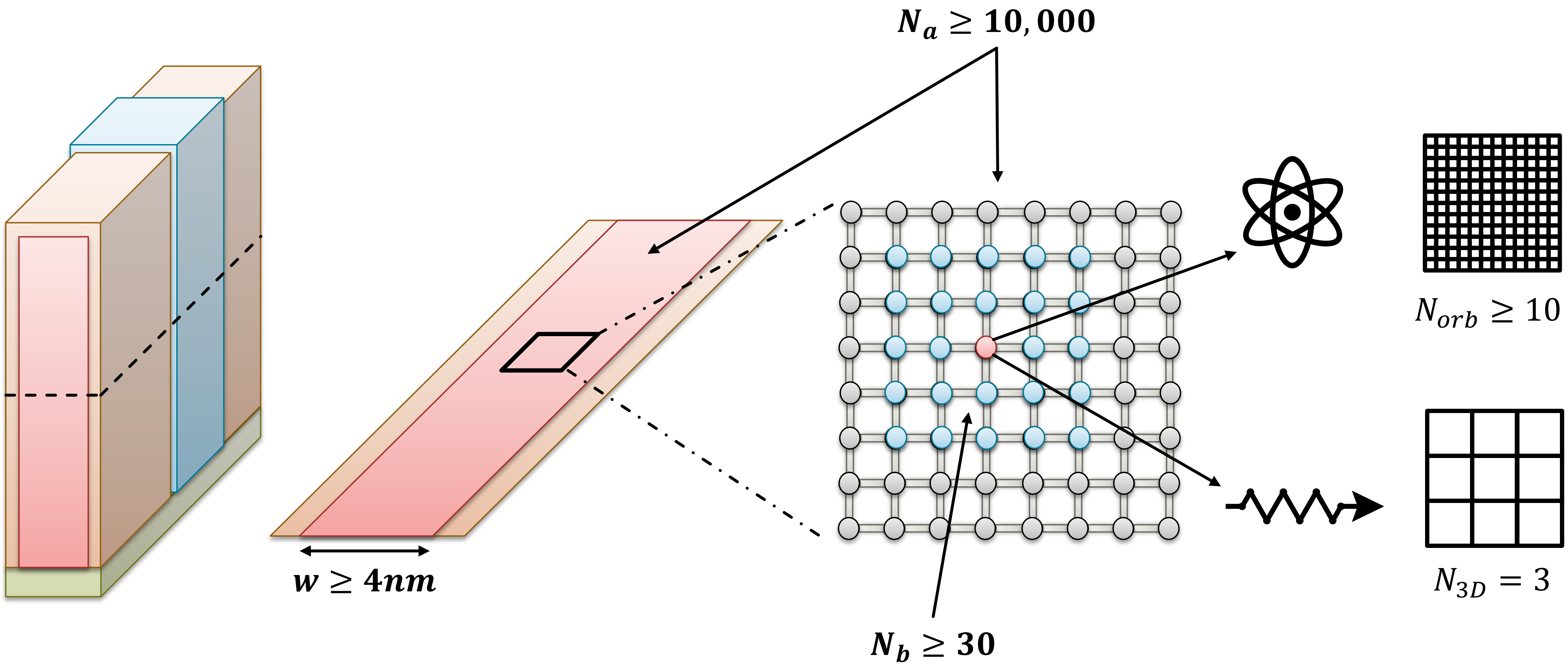
Quantum Transport Simulation



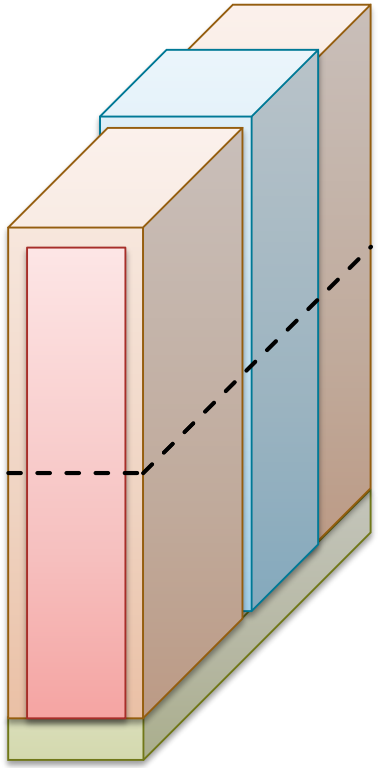
Workflow of Quantum Transport Simulation

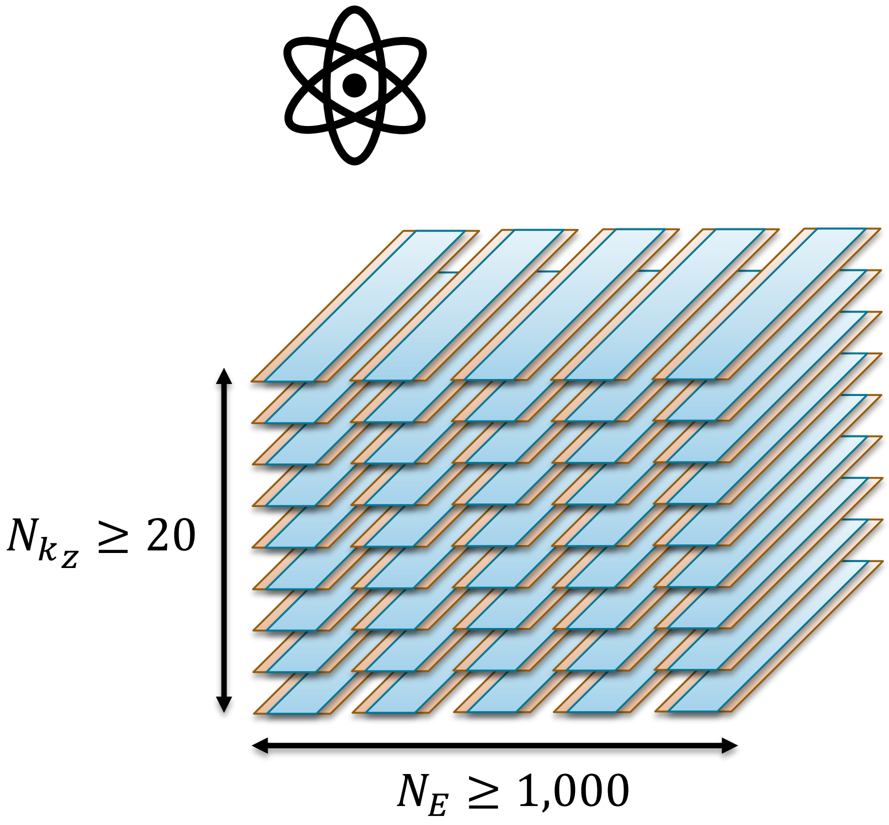


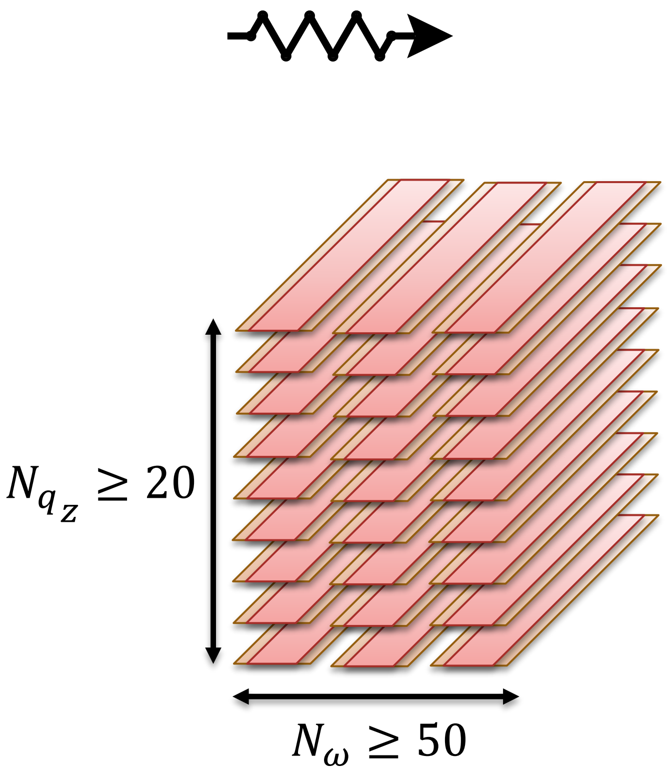
OMEN Model



OMEN Model







NEGF Mathematical Formulation

OMEN

NEGF

SSE $\Sigma[G(E \pm \hbar\omega, k_z - q_z)\, D(\omega, q_z)](E, k_z)$

Electrons $\boldsymbol{G(E, k_z)}$

$$\left(E \cdot S - H - \Sigma^R\right) \cdot G^R = I$$
$$G^< = G^R \cdot \Sigma^< \cdot G^A$$

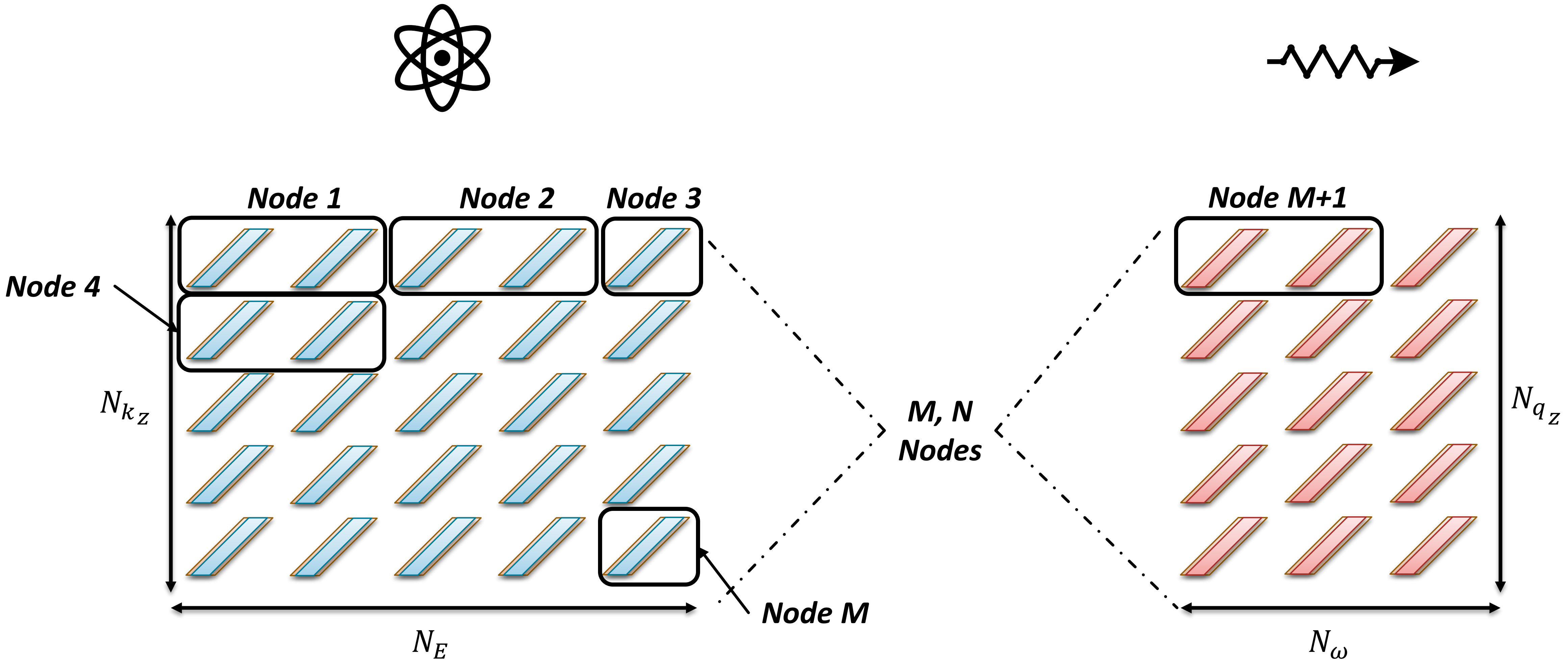
Phonons $\boldsymbol{D(\omega, q_z)}$

$$\left(\omega^2 - \Phi - \Pi^R\right) \cdot D^R = I$$
$$D^< = D^R \cdot \Pi^< \cdot D^A$$

GF

SSE $\Pi[G(E, k_z)\, G(E + \hbar\omega, k_z + q_z)](\omega, q_z)$

Assignment to Compute Resources



Green’s Functions Phase

OMEN

Electrons $G(E, k_z)$

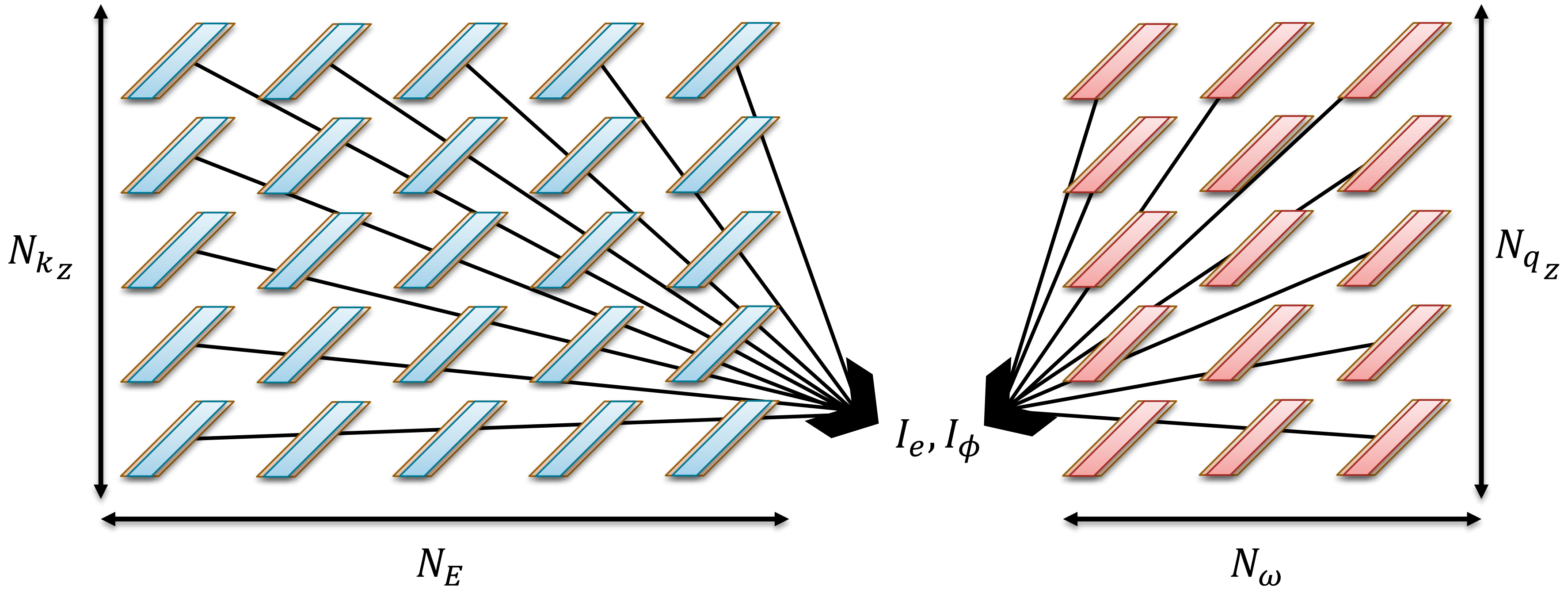
$$\left(E \cdot S - H - \Sigma^R\right) \cdot G^R = I$$
$$G^< = G^R \cdot \Sigma^< \cdot G^A$$

Phonons $D(\omega, q_z)$

$$\left(\omega^2 - \Phi - \Pi^R\right) \cdot D^R = I$$
$$D^< = D^R \cdot \Pi^< \cdot D^A$$

GF

Embarrassingly parallel computation of G/D + Reduction for I



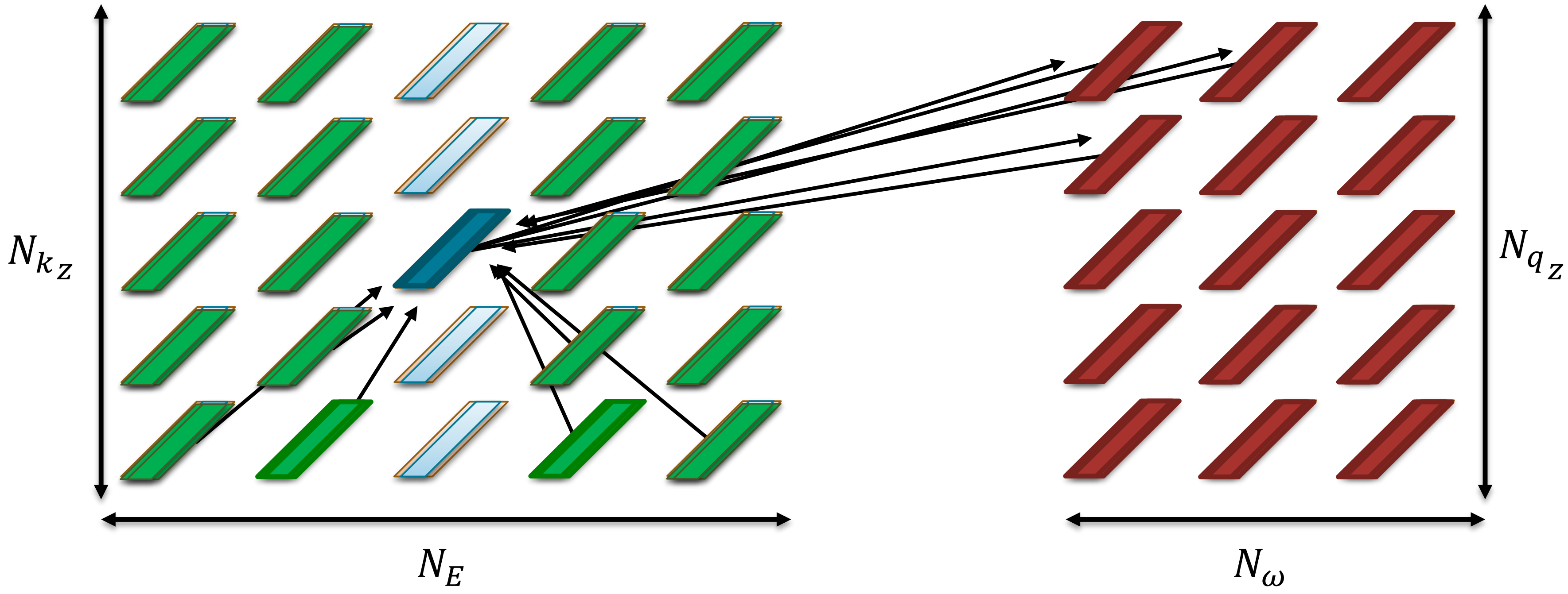
Scattering Self-Energies Phase

SSE $\Sigma[G(E \pm \hbar\omega, k_z - q_z)\, D(\omega, q_z)](E, k_z)$

Stencil-like computation for Σ/Π

$$2N_{q_z}N_\omega$$

SSE $\Pi[G(E, k_z)\, G(E + \hbar\omega, k_z + q_z)](\omega, q_z)$



Data-Centric Framework

Data-Centric Intermediate Representation (SDFG)

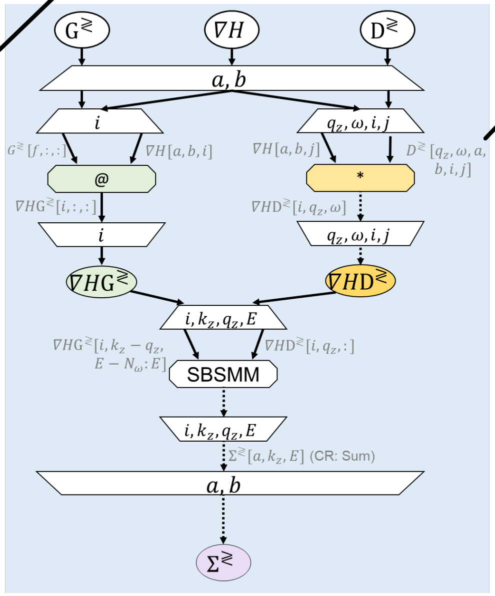
Problem F

Original Application | *OMEN rewritten with the DaCe framework*

OMEN	DaCe OMEN	
15,798 C++ Lines	3,155 Python Lines	2,015 SDFG Nodes

```python
@dace.program
def sse_sigma(neigh_idx: dace.int32[NA, NB],
              dH: dace.complex128[NA, NB, N3D, Norb, Norb],
              G: dace.complex128[Nkz, NE, NA, Norb, Norb],
              D: dace.complex128[Nqz, Nw, NA, NB, N3D, N3D],
              Sigma: dace.complex128[Nkz, NE, NA, Norb, Norb]):

    # Declaration of Map scope
    for k, E, q, w, i, j, a, b in dace.map[0:Nkz, 0:NE,
                                           0:Nqz, 0:Nw,
                                           0:N3D, 0:N3D,
                                           0:NA, 0:NB]:

        f = neigh_idx[a, b]
        dHG = G[k-q, E-w, f] @ dH[a, b, i]
        dHD = dH[a, b, j] * D[q, w, a, b, i, j]
        Sigma[k, E, a] += dHG @ dHD
```

High-Level Program

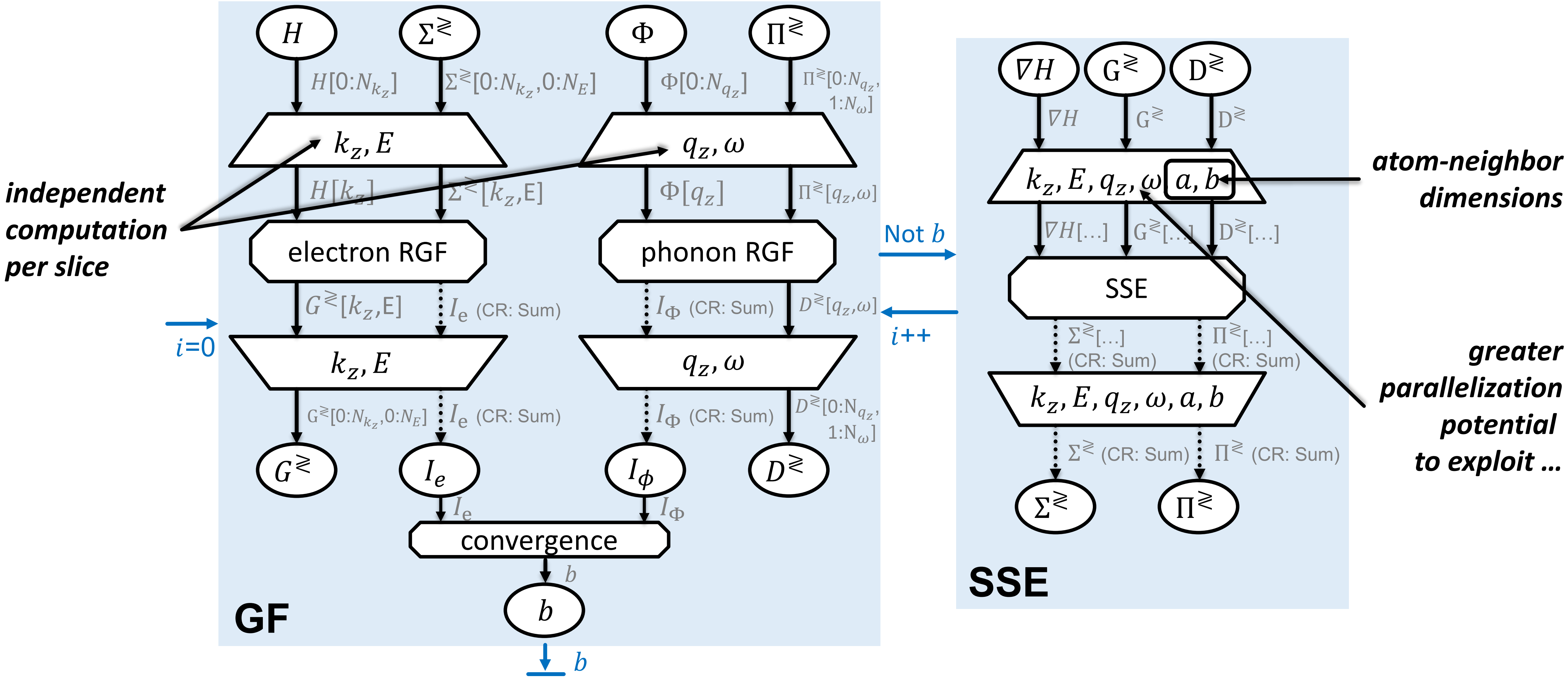


Graph Transformations (API, Interactive)

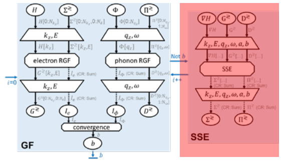
```cpp
...plex<double> * __restrict__ A,
   complex<double> * __restrict__ B,
   x<double> * __restrict__ C,
   Nmat, int Nbatch, int Csize) {

   int t = threadIdx.x*ILP;
   int j = threadIdx.y;
   int klen = (Norb * Nmat / TOTALK);
   int kstart = blockIdx.y;

   A += t * Norb * Norb + i + kstart*klen*Norb;
   B += j + kstart*klen*Norb;

   thrust::complex<double> res[ILP] = {0};

   for(int k = 0; k < klen; ++k) {
       auto b = B[0];
       #pragma unroll
       for(int l = 0; l < ILP; ++l)
           res[l] += A[l] * b;
       A += Norb;
       B += Norb;
   }

   C += kstart * Csize + t * Norb * Norb + j * Norb + i;
   for(int l = 0; l < ILP; ++l)
       C[l] = res[l];
}
```

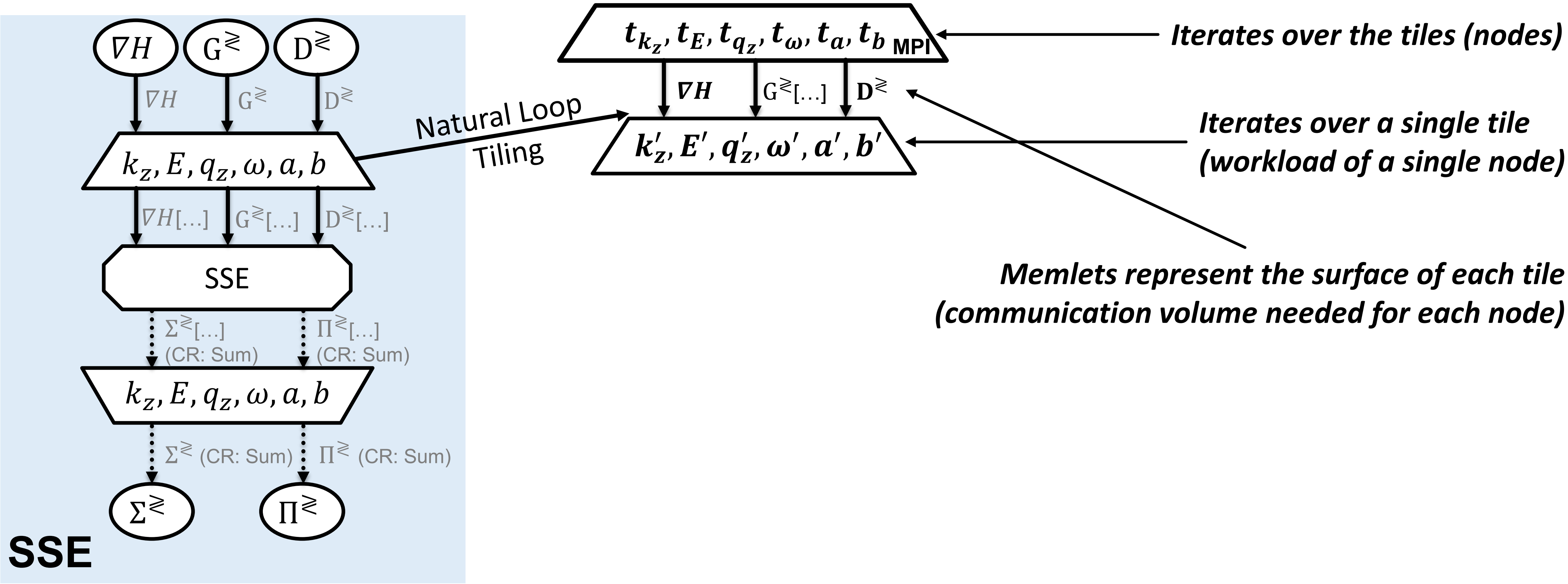
Code Generation

Data-Centric Representation of OMEN

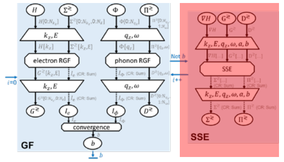


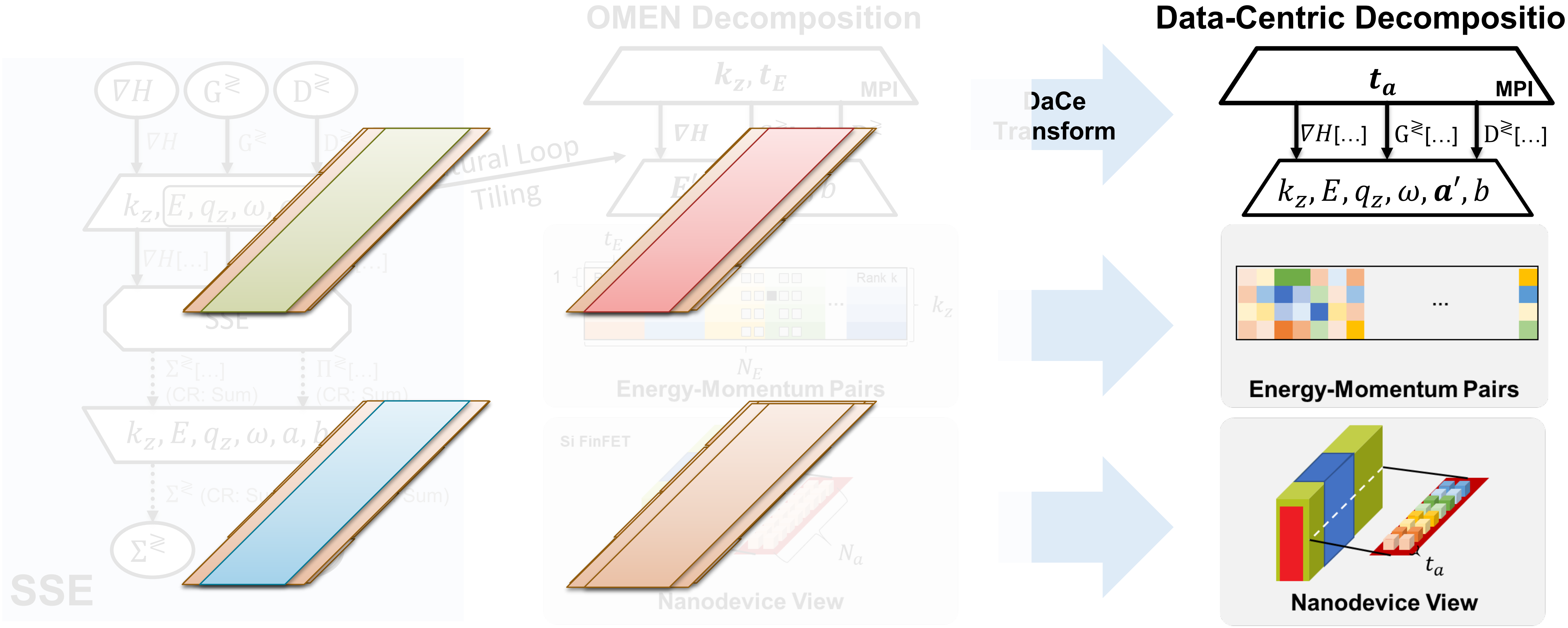
Optimizing Coarse-Grained Data Movement





Optimizing Coarse-Grained Data Movement

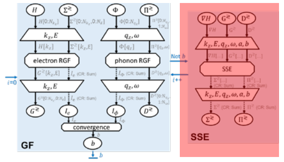


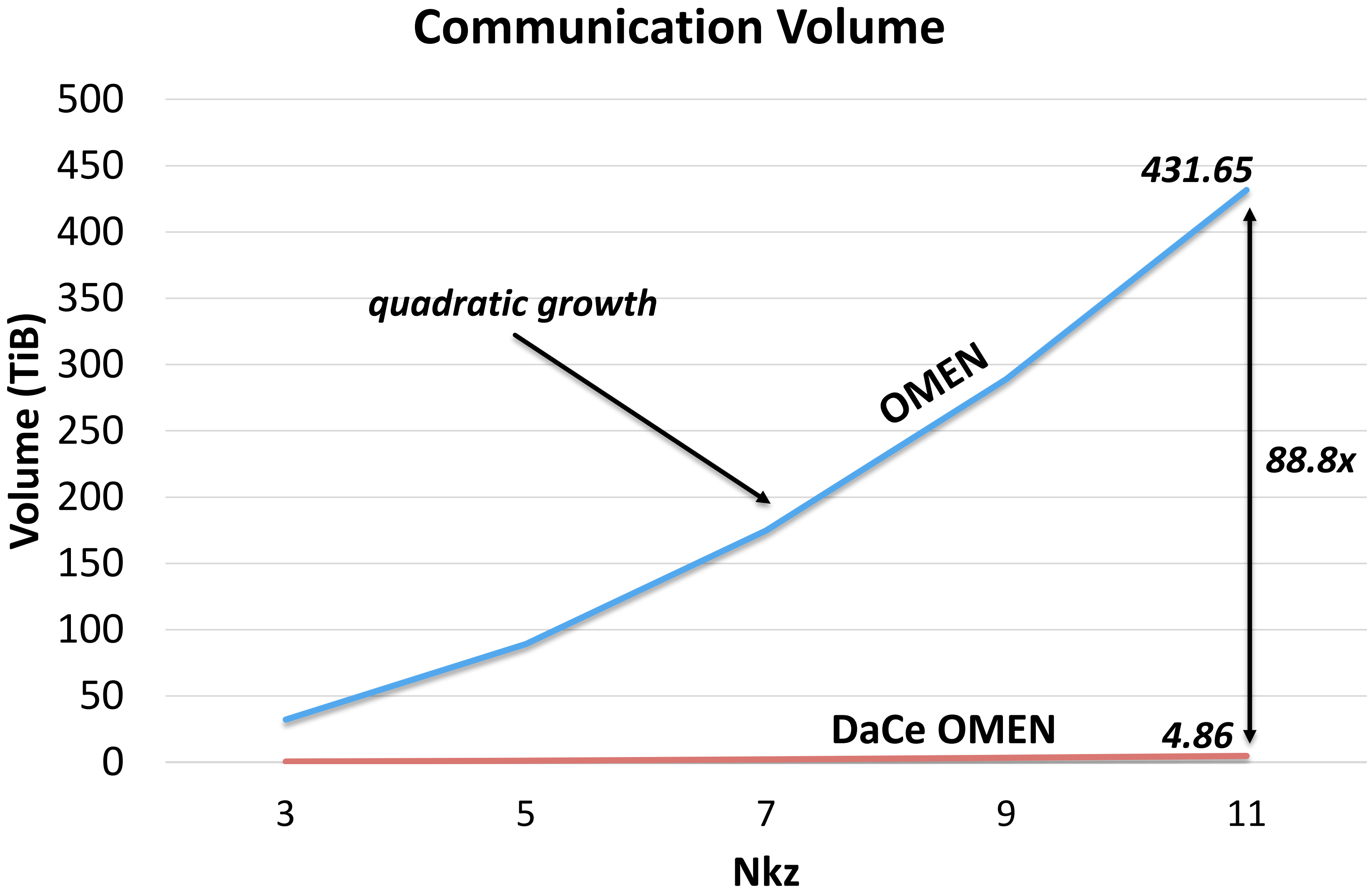


Method: Broadcast, Reduce, P2P
Volume: $\mathcal{O}(N_{k_z} N_E N_{q_z} N_\omega N_a N_{orb}^2)$
MPI Invocations: $9 N_\omega N_{q_z}$

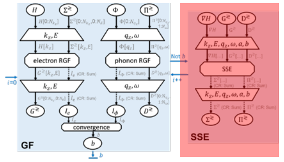
Method: Collective Alltoall
Volume: $\mathcal{O}(N_{k_z}(N_E + N_\omega)(N_a + N_b)\, N_{orb}^2)$
MPI Invocations: 4

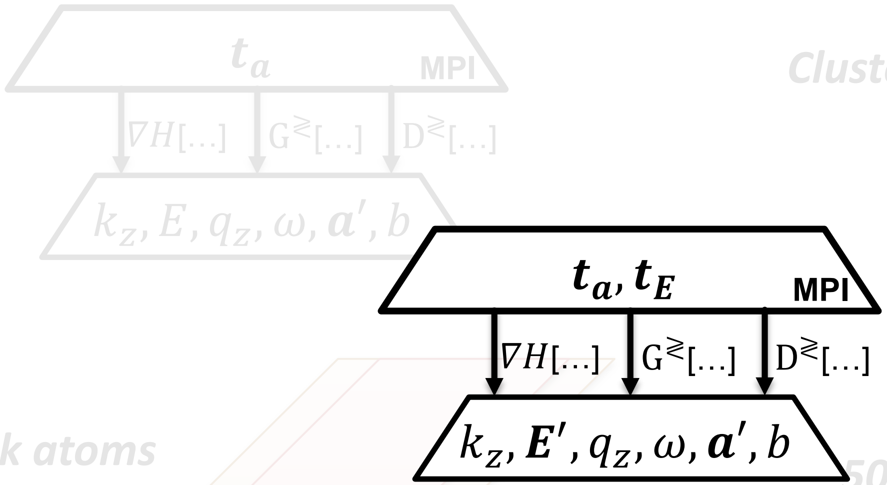
Optimizing Coarse-Grained Data Movement





Optimizing Coarse-Grained Data Movement





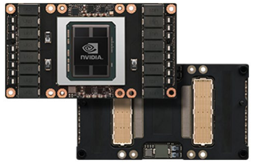
Cluster with 100 nodes

5k atoms

50 atoms per node

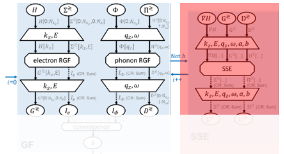
Need for flexibility

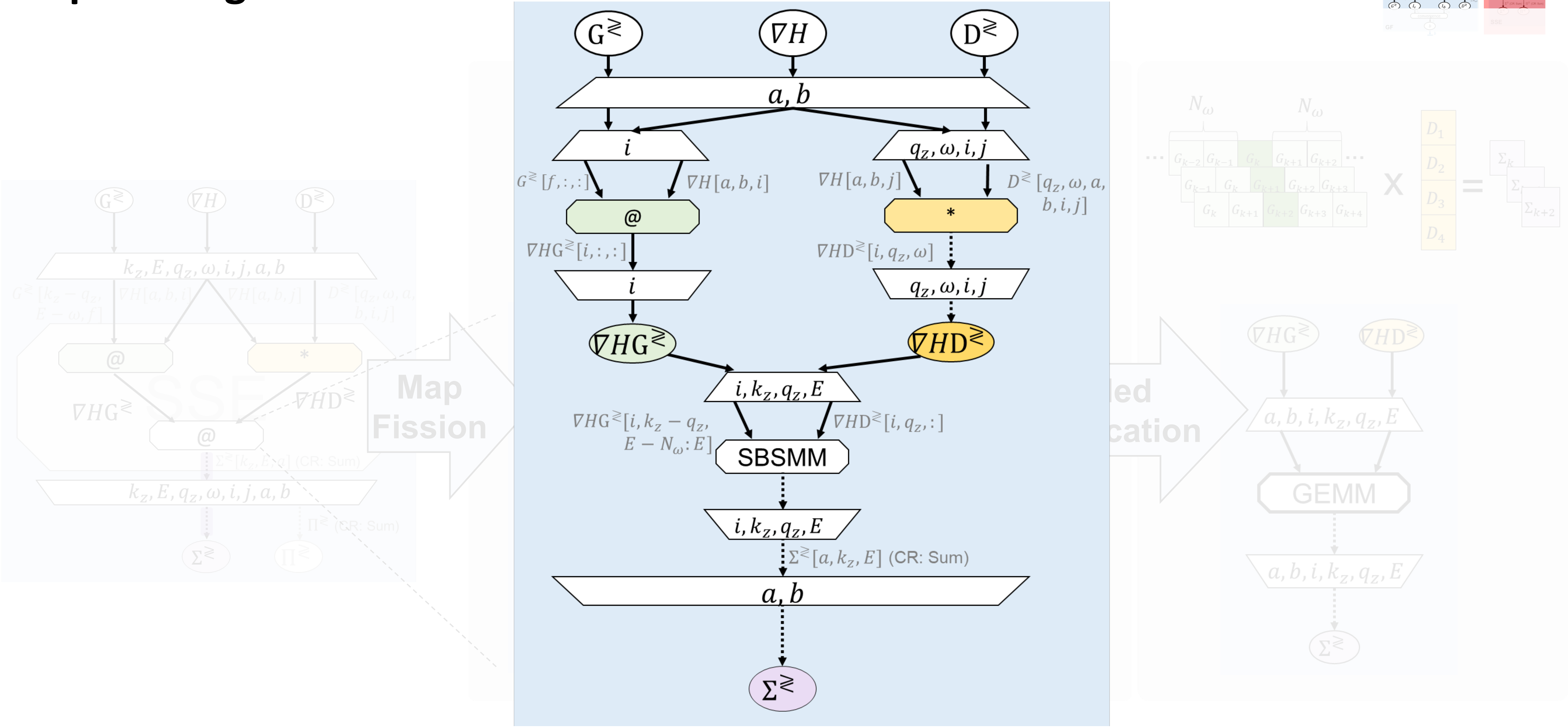
Source: Swiss National Supercomputing Centre

5.7k **X** 

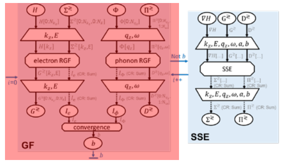
Source: NVIDIA

Optimizing Fine-Grained Data Movement





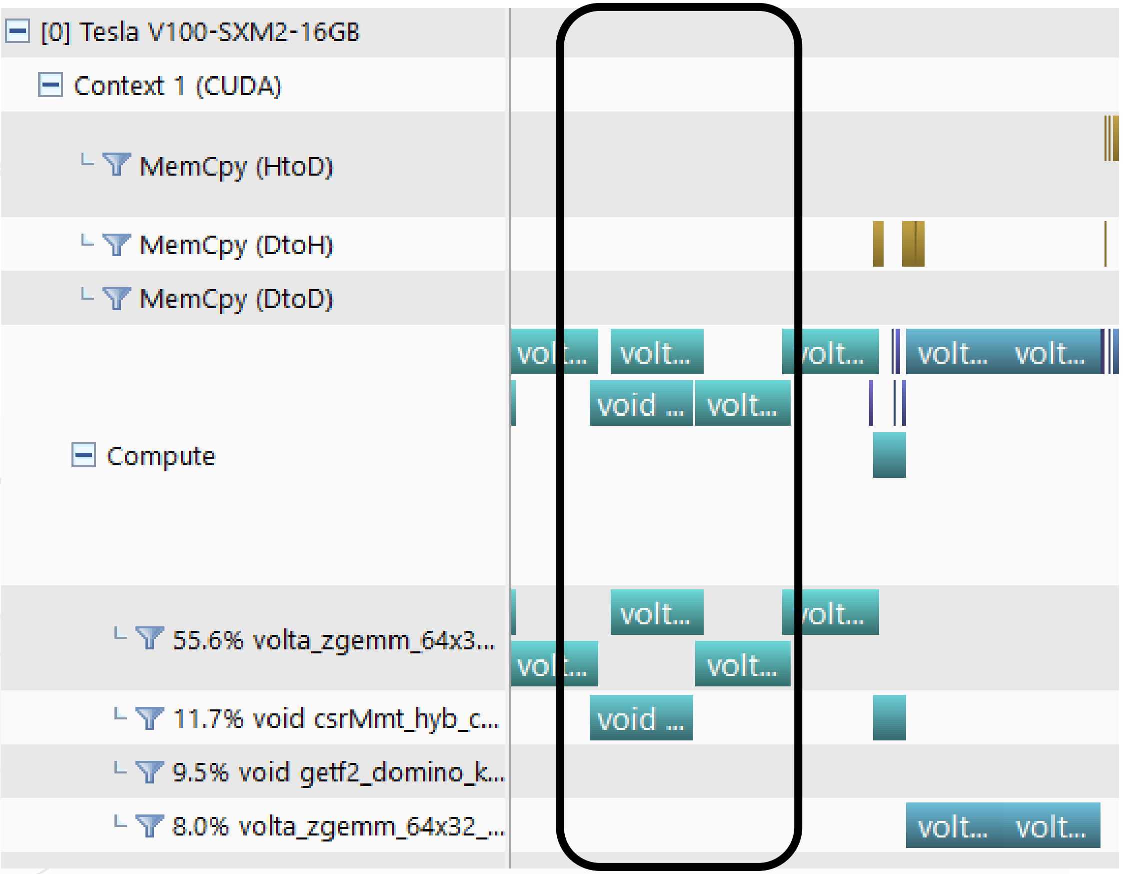
Extracting Parallelism



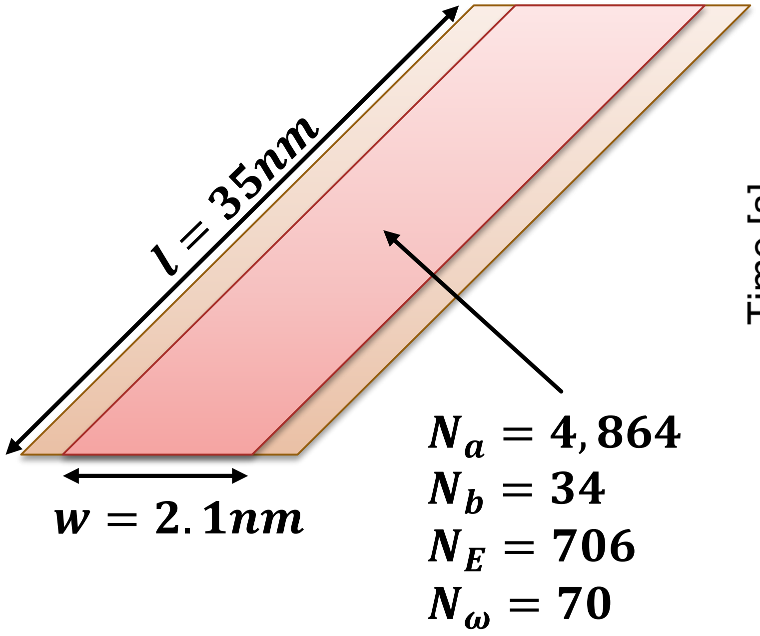
```
auto __a = dace::ArrayViewIn<dace::complex128, 2, 1> (gpu_tmpL, bsize, 1);
auto *a = __a.ptr<1>();
auto __b = dace::ArrayViewIn<dace::complex128, 2, 1> (gpu_hergR, bsize, 1);
auto *b = __b.ptr<1>();

auto __c = dace::ArrayViewOut<dace::complex128, 2, 1> (gpu_tmpL_R, bsize, 1);
auto *c = __c.ptr<1>();

int __dace_current_stream_id = 2;
cudaStream_t __dace_current_stream = dace::cuda::__streams[__dace_current_stream_id];

cublasSetStream(handle, __dace_current_stream);
cublasStatus_t status = cublasZgemm(
    handle,
    CUBLAS_OP_N, CUBLAS_OP_N,
    bsize, bsize, bsize,
    const_pone,
    (cuDoubleComplex*)b, bsize,
    (cuDoubleComplex*)a, bsize,
    const_zero,
    (cuDoubleComplex*)c, bsize
);

cudaEventRecord(dace::cuda::__events[5], dace::cuda::__streams[2]);
cudaStreamWaitEvent(dace::cuda::__streams[0], dace::cuda::__events[5], 0);
```

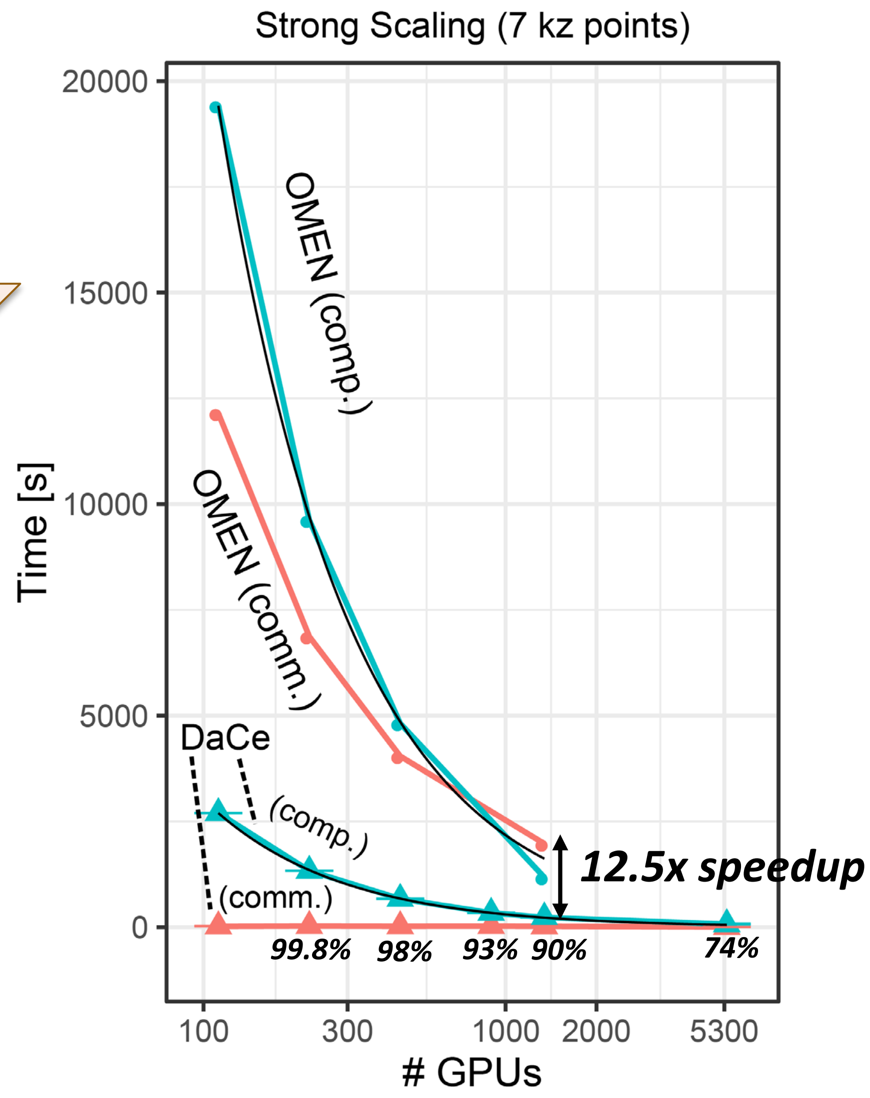
assignment to streams

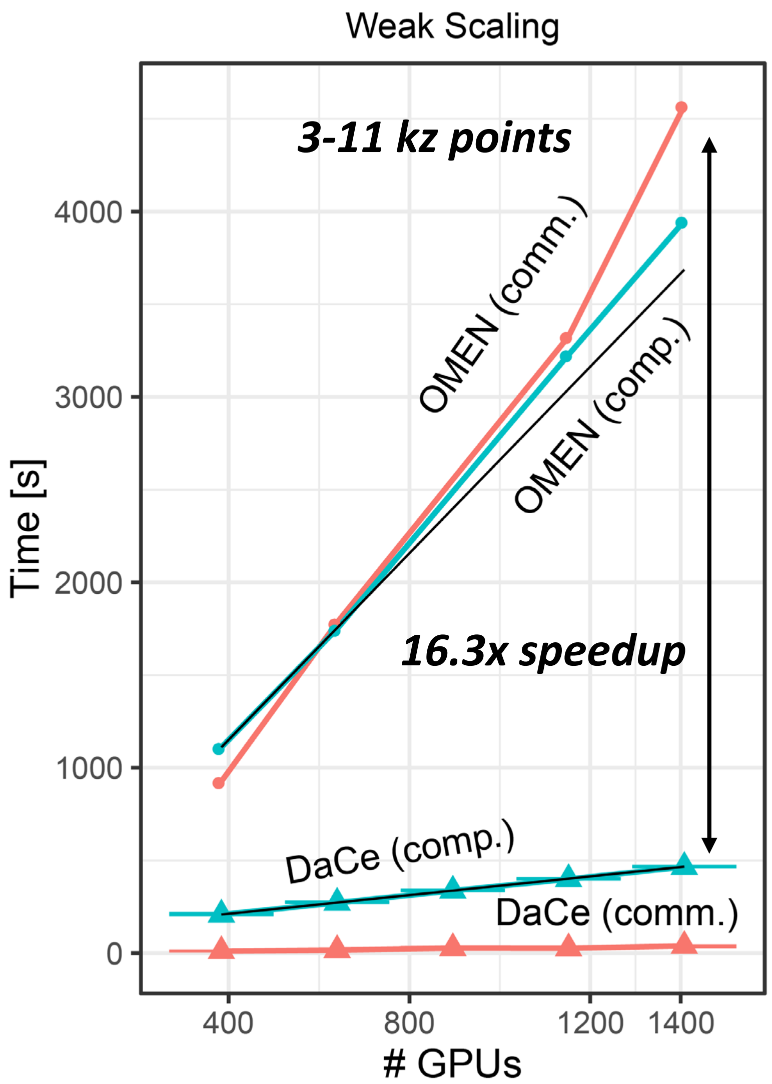
synchronization code



OMEN vs DaCe OMEN: Performance







Conclusions

Data-Centric Framework

Data-Centric Model

OMEN Application

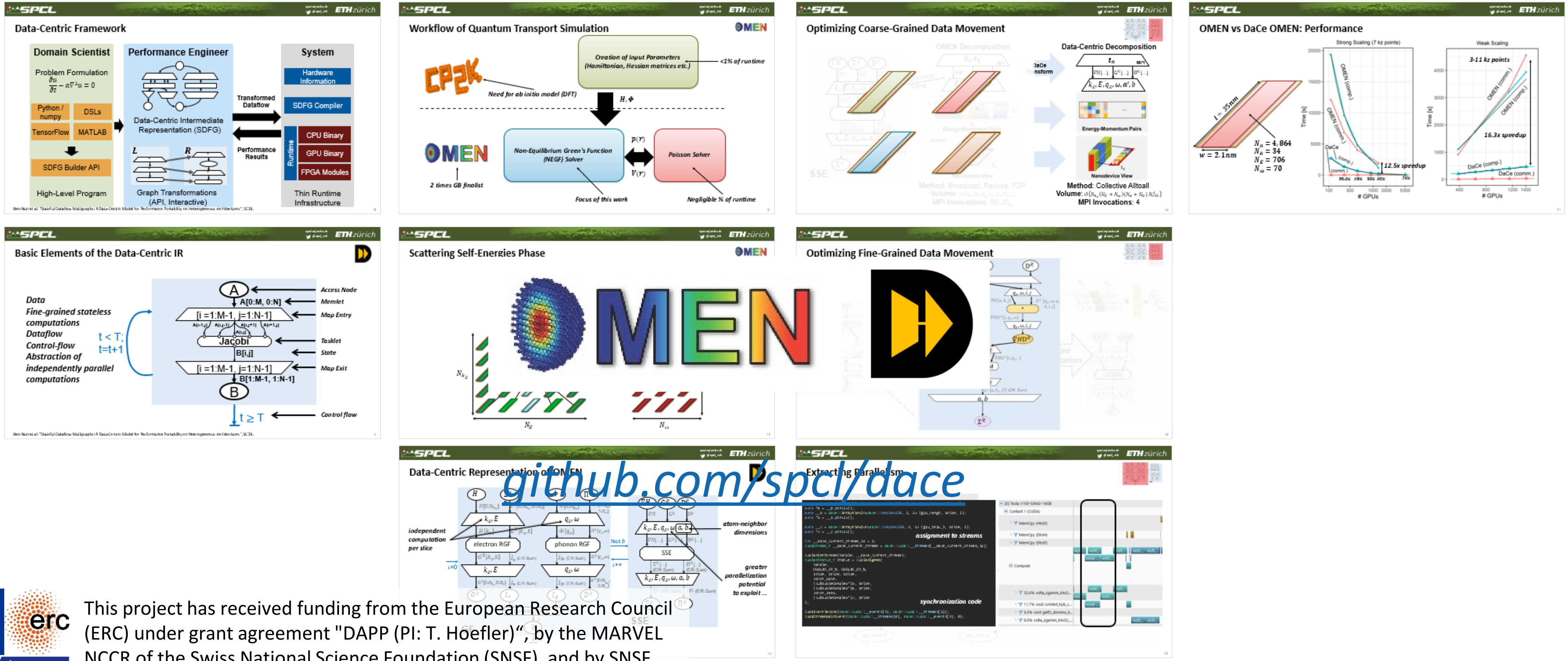
Domain Scientists' View

Data-Centric View

Optimizing Coarse-Grained and Fine-Grained Dataflow

Extracting Parallelism

Performance



github.com/spcl/dace

This project has received funding from the European Research Council (ERC) under grant agreement "DAPP (PI: T. Hoefler)", by the MARVEL NCCR of the Swiss National Science Foundation (SNSF), and by SNSF grant 175479 (ABIME).