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

Alexandros Nikolaos Ziogas*, Tal Ben-Nun*, Guillermo Indalecio Fernández†, Timo Schneider*, Mathieu Luisier†, and Torsten Hoefler*
*Scalable Parallel Computing Laboratory, ETH Zurich, Switzerland
†Integrated Systems Laboratory, ETH Zurich, Switzerland

ABSTRACT

Designing efficient cooling systems for integrated circuits (ICs) relies on a deep understanding of the electro-thermal properties of transistors. To shed light on this issue in currently fabricated FinFETs, a quantum mechanical solver capable of revealing atomically-resolved electron and phonon transport phenomena from first-principles is required. In this paper, we consider a global, data-centric view of a state-of-the-art quantum transport simulator to optimize its execution on supercomputers. The approach yields coarse- and fine-grained data-movement characteristics, which are used for performance and communication modeling, communication-avoidance, and data-layout transformations. The transformations are tuned for the Piz Daint and Summit supercomputers, where each platform requires different caching and fusion strategies to perform optimally. The presented results make *ab initio* device simulation enter a new era, where nanostructures composed of over 10,000 atoms can be investigated at an unprecedented level of accuracy, paving the way for better heat management in next-generation ICs.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel and high-performance simulations**; **Parallel computing methodologies**; **Quantum mechanic simulation**.

ACM Reference Format:
Alexandros Nikolaos Ziogas*, Tal Ben-Nun*, Guillermo Indalecio Fernández†, Timo Schneider*, Mathieu Luisier†, and Torsten Hoefler*. 2019. Optimizing the Data Movement in Quantum Transport Simulations via Data-Centric Parallel Programming. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3295500.3356200

1 INTRODUCTION

Heat dissipation in microchips reached alarming peak values of 100 W/cm^2 already in 2006 [21, 28]. This led to the end of Dennard scaling and the beginning of the "multicore crisis", an era with energy-efficient parallel, but sequentially slower multicore CPUs. Now, more than ten years later, average power densities of up to 30 W/cm^2, about four times more than hot plates, are commonplace in modern high-performance CPUs, putting thermal management at the center of attention of circuit designers [20]. By scaling the dimensions of transistors more rapidly than their supply voltage, the semiconductor industry has kept increasing heat dissipation from one generation of microprocessors to the other. In this context, large-scale data and supercomputing centers are facing critical challenges regarding the design and cost of their cooling infrastructures. The price to pay for that has become exorbitant, as the cooling can be up to 40% of the total electicity consumed by data centers; a cumulative cost of many billion dollars per year.

Landauer's theoretical limit of energy consumption for non-reversible computing offers a glimmer of hope: today's processing units require orders of magnitude more energy than the $k_B T \ln 2$ Joule bound to (irreversibly) change one single bit. However, to approach this limit, it will be necessary to first properly understand the mechanisms behind nanoscale heat dissipation in semiconductor devices [21]. Fin field-effect transistors (FinFETs), as schematized in Fig. 1(a-c), build the core of all recent integrated circuits (ICs). Their dimensions do not exceed 100 nanometers along all directions, even 10 nm along one of them, with an active region composed of fewer than 1 million atoms. This makes them subject to strong quantum mechanical and peculiar thermal effects.

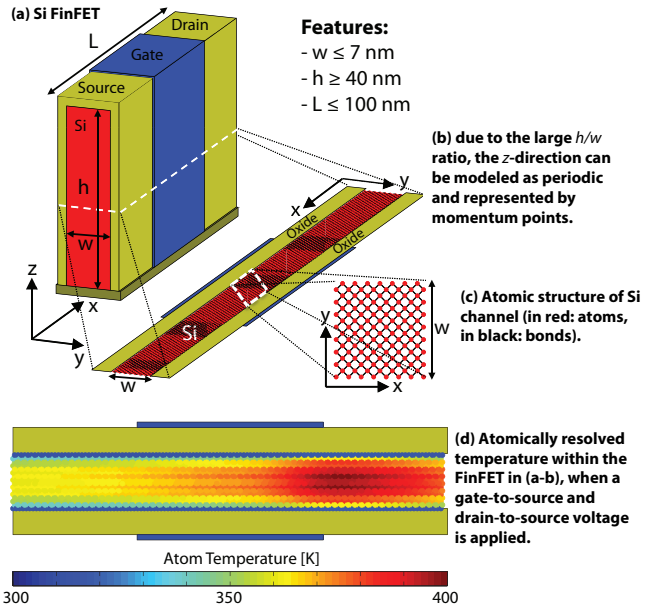


Figure 1: Schematic view of a 3-D Silicon FinFET.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC '19, November 17–22, 2019, Denver, CO, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6229-0/19/11…$15.00
https://doi.org/10.1145/3295500.3356200

When a voltage is applied across FinFETs, electrons start to flow from the source to the drain contact, giving rise to an electrical current whose magnitude depends on the gate bias. The potential difference between source and drain allows electrons to transfer part of their energy to the crystal lattice surrounding them. This energy is converted into atomic vibrations, called phonons, that can propagate throughout FinFETs. The more atoms vibrate, the "hotter" a device becomes. This phenomenon, known as self- or Joule-heating, plays a detrimental role in today's transistor technologies and has consequences up to the system level. It is illustrated in Fig. 1(d): a strong increase of the lattice temperature can be observed close to the drain contact of the simulated FinFET. The negative influence of self-heating on the CPU/GPU performance can be minimized by devising computer-assisted strategies to efficiently evacuate the generated heat from the active region of transistors.

Electro-thermal properties of nano-devices can be modeled and analyzed via *Quantum Transport (QT) simulation*, where electron and phonon currents are evaluated by taking quantum mechanics into account. Due to the large height/width ratio of FinFETs, these effects can be physically captured in a two-dimensional simulation domain comprising 10,000 to 15,000 thousand atoms. Such dissipative simulations involve solving the Schrödinger equation with open boundary conditions over several momentum and energy vectors that are coupled to each other through electron-phonon interactions. A straightforward algorithm to address this numerical problem consists of defining two loops, one over the momentum points and another one over the electron energies, which results in potentially extraneous execution dependencies and a complex communication pattern. The latter scales sub-optimally with the number of participating atoms and computational resources, thus limiting current simulations to the order of thousand atoms.

While the schedule (ordering) of the loops in the solver is natural from the physics perspective (§ 2), the data decomposition it imposes when parallelizing is not scalable from the computational perspective. To investigate larger device structures, it is crucial to reformulate the problem as a communication-avoiding algorithm — rescheduling computations across compute resources to minimize data movement.

Even when each node is operating at maximum efficiency, large-scale QT simulations are both bound by communication volume and memory requirements. The former inhibits strong scaling, as simulation time includes nanostructure-dependent point-to-point communication patterns, which becomes infeasible when increasing node count. The memory bottleneck is a direct result of the former. It hinders large simulations due to the increased memory requirements w.r.t. atom count. Transforming the QT simulation algorithm to minimize communication is thus the key to simultaneously model larger devices and increase scalability on different supercomputers.

The current landscape of supercomputing resources is dominated by heterogeneous nodes, where no two clusters are the same. Each setup requires careful tuning of application performance, focused mostly around data locality [26]. As this kind of tuning demands in-depth knowledge of the hardware, it is typically performed by a *Performance Engineer*, a developer who is versed in intricate system details, existing high-performance libraries, and capable of modeling performance and setting up optimized procedures independently. This role, which complements the *Domain Scientist*, has been increasingly important in scientific computing for the past three decades, but is now essential for any application beyond straightforward linear algebra to operate at extreme scales. Until recently, both Domain Scientists and Performance Engineers would work with one code-base. This creates a co-dependent situation [19], where the original domain code is tuned to a point that making modifications to the algorithm or transforming its behavior is difficult to one without the presence of the other, even if data locality or computational semantics are not changed.

In this paper, we propose a paradigm change by rewriting the problem from a data-centric perspective. We use OMEN, the current state-of-the-art quantum transport simulation application [17], as our baseline, and show that the key to formulating a communication-avoiding variant is tightly coupled with recovering local and global data dependencies of the application. We start from a reference Python implementation, using Data-Centric (DaCe) Parallel Programming [1] to express the computations separately from data movement. DaCe automatically constructs a *stateful dataflow* view that can be used to optimize data movement without modifying the original computation. This enables rethinking the communication pattern of the simulation, and tuning the data movement for each target supercomputer.

In sum, the paper makes the following contributions:

- Construction of the dissipative quantum transport simulation problem from a physical perspective;
- Definition of the stateful dataflow of the algorithm, making data and control dependencies explicit on all levels;
- Creation of a novel *tensor-free communication-avoiding variant* of the algorithm based on the data-centric view;
- Optimal choice of the decomposition parameters based on the modeling of the performance and communication of our variant, nano-device configuration, and cluster architecture;
- Demonstration of the algorithm's scalability on two vastly different supercomputers — Piz Daint and Summit — up to full-scale runs on 10k atoms, 21 momentum points and 1,000 energies per momentum;
- A performance increase of *1–2 orders of magnitude* over the previous state of the art, all from a data-centric Python implementation that reduces code-length by a factor of five.

2 STATEMENT OF THE PROBLEM

A technology computer aided design (TCAD) tool can shed light on the electro-thermal properties of nano-devices, provided that it includes the proper physical models:

- The dimensions of FinFETs calls for an atomistic quantum mechanical treatment of the device structures;
- The electron and phonon bandstructures should be accurately and fully described;
- The interactions between electrons and phonons, especially energy exchanges, should be accounted for.

The Non-equilibrium Green's Function (NEGF) formalism [5] combined with density functional theory (DFT) [13] fulfills all these requirements and lends itself optimally to the investigation of self-heating in arbitrary device geometries. With NEGF, both electron

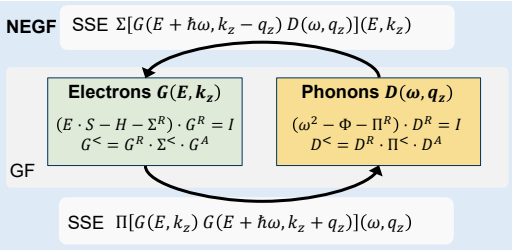


Figure 2: Self-consistent coupling between the GF and SSE phases (kernels) as part of the NEGF formalism.

and phonon transport can be described, together with their respective interactions. With the help of DFT, an *ab initio* method, any material (combination) can be handled at the atomic level without the need for empirical parameters.

FinFETs are essentially three-dimensional (3-D) components that can be approximated as 2-D slices in the x-y plane, whereas the height, aligned with the z-axis (see Fig. 1(a-b)), can be treated as a periodic dimension and represented by a momentum vector k_z or q_z in the range $[-\pi, \pi]$. Hence, the DFT+NEGF equations have the following form:

$$\begin{cases} \Big(E\cdot \mathbf{S}(k_z) - \mathbf{H}(k_z) - \Sigma^R(E,k_z)\Big)\cdot \mathbf{G}^R(E,k_z) = \mathbf{I} \\ \mathbf{G}^{\gtrless}(E,k_z) = \mathbf{G}^R(E,k_z)\cdot\Sigma^{\gtrless}(E,k_z)\cdot\mathbf{G}^A(E,k_z). \end{cases} \quad (1)$$

In Eq. (1), E is the electron energy, $\mathbf{S}(k_z)$ and $\mathbf{H}(k_z)$ the overlap and Hamiltonian matrices, respectively. They typically exhibit a block tri-diagonal structure and a size $N_A \times N_{orb}$ with N_A as the total number of atoms in the considered structure and N_{orb} the number of orbitals (basis components) representing each atom. The $\mathbf{S}(k_z)$ and $\mathbf{H}(k_z)$ matrices must be produced by a DFT package relying on a localized basis set, e.g., SIESTA [12] or CP2K [27]. The $\mathbf{G}(E,k_z)$'s refer to the electron Green's Functions (GF) at energy E and momentum k_z. They are of the same size as $\mathbf{S}(k_z)$, $\mathbf{H}(k_z)$, and $\mathbf{I}$, the identity matrix. The GF can be either retarded (R), advanced (A), lesser ($<$), or greater ($>$) with $\mathbf{G}^A(E,k_z)=\Big(\mathbf{G}^R(E,k_z)\Big)^T$. The same conventions apply to the self-energies $\Sigma(E,k_z)$ that include a boundary and a scattering term. The former connects the simulation domain to external contacts, whereas the latter encompasses all possible interactions of electrons with their environment.

To handle phonon transport, the following NEGF-based system of equations must be processed:

$$\begin{cases} \Big(\omega^2\cdot\mathbf{I} - \Phi(q_z) - \Pi^R(\omega,q_z)\Big)\cdot\mathbf{D}^R(\omega,q_z) = \mathbf{I} \\ \mathbf{D}^{\gtrless}(\omega,q_z) = \mathbf{D}^R(\omega,q_z)\cdot\Pi^{\gtrless}(\omega,q_z)\cdot\mathbf{D}^A(\omega,q_z), \end{cases} \quad (2)$$

where the $\mathbf{D}(\omega,q_z)$'s are the phonon Green's functions at frequency ω and momentum q_z and the $\mathbf{\Pi}(\omega,q_z)$'s the self-energies, while $\mathbf{\Phi}(q_z)$ refers to the dynamical (Hessian) matrix of the studied domain, computed with density functional perturbation theory (DFPT) [25]. The phonon Green's Function types are the same as for electrons (retarded, advanced, lesser, and greater). All matrices involved in Eq. (2) are of size $N_A \times N_{3D}$, N_{3D}=3 corresponding to the number of directions along which the crystal can vibrate (x, y, z).

Equations (1) and (2) must be solved for all possible electron energy (N_E) and momentum (N_{k_z}) points as well as all phonon frequencies (N_ω) and momentum (N_{q_z}). This can be done with a so-called recursive Green's Function (RGF) algorithm [23] that takes advantage of the block tri-diagonal structure of the matrices $\mathbf{H}$, $\mathbf{S}$, and $\mathbf{\Phi}$. All matrices can be divided into *bnum* blocks with $\frac{N_A}{bnum}$ atoms each, if the structure is homogeneous, as here. RGF then performs a forward and backward pass over the *bnum* blocks that compose the 2-D slice. Both passes involve a number of multiplications between matrices of size $\left(\frac{N_A}{bnum}N_{orb}\right)^2$ for electrons (or $\left(\frac{N_A}{bnum}N_{3D}\right)^2$ for phonons) for each block.

Table 1: Typical QT Simulation Parameters

Variable	Description	Range
N_{k_z}	Number of electron momentum points	[1, 21]
N_{q_z}	Number of phonon momentum points	[1, 21]
N_E	Number of energy points	[700, 1500]
N_ω	Number of phonon frequencies	[10, 100]
N_A	Total number of atoms per device structure	See Table 2
N_B	Neighbors considered for each atom	[4, 50]
N_{orb}	Number of orbitals per atom	[1, 30]
N_{3D}	Degrees of freedom for crystal vibrations	3

The main computational bottleneck does not come from RGF, but from the fact that in the case of self-heating simulations the energy-momentum (E,k_z) and frequency-momentum (ω,q_z) pairs are not independent from each other, but tightly coupled through the scattering self-energies (SSE) $\Sigma^{R\gtrless,S}$ and $\mathbf{\Pi}^{R\gtrless,S}$. These matrices are made of blocks of size $N_{orb}\times N_{orb}$ and $N_{3D}\times N_{3D}$, respectively, and are given by [22]:

$$\Sigma^{\gtrless S}_{aa}(E,k_z) = i\sum_{q_z ijl}\int \frac{d\hbar\omega}{2\pi}\Big[\nabla_i\mathbf{H}_{ab}\cdot\mathbf{G}^{\gtrless}_{bb}(E-\hbar\omega,k_z-q_z)\cdot \nabla_j\mathbf{H}_{ba}\cdot\Big(\mathbf{D}^{\gtrless ij}_{ba}(\omega,q_z) - \mathbf{D}^{\gtrless ij}_{bb}(\omega,q_z) - \mathbf{D}^{\gtrless ij}_{aa}(\omega,q_z) + \mathbf{D}^{\gtrless ij}_{ab}(\omega,q_z)\Big)\Big], \quad (3)$$

$$\mathbf{\Pi}^{\gtrless S}_{aa}(\omega,q_z) = -i\sum_{k_z l}\int \frac{dE}{2\pi}\mathrm{tr}\Big\{\nabla_i\mathbf{H}_{ba}\cdot\mathbf{G}^{\gtrless}_{aa}(E+\hbar\omega,k_z+q_z)\cdot \nabla_j\mathbf{H}_{ab}\cdot\mathbf{G}^{\lessgtr}_{bb}(E,k_z)\Big\}, \quad (4)$$

$$\mathbf{\Pi}^{\gtrless S}_{ab}(\omega,q_z) = i\sum_{k_z}\int \frac{dE}{2\pi}\mathrm{tr}\Big\{\nabla_i\mathbf{H}_{ba}\cdot\mathbf{G}^{\gtrless}_{aa}(E+\hbar\omega,k_z+q_z)\cdot \nabla_j\mathbf{H}_{ab}\cdot\mathbf{G}^{\lessgtr}_{bb}(E,k_z)\Big\}. \quad (5)$$

In Eqs. (3-5), all Green's Functions $\mathbf{G}_{ab}$ ($\mathbf{D}_{ab}$) are matrices of size $N_{orb}\times N_{orb}$ ($N_{3D}\times N_{3D}$). They describe the coupling between all orbitals (vibrational directions) of two neighbor atoms a and b situated at position $\mathbf{R}_a$ and $\mathbf{R}_b$. Each atom possesses N_B neighbors. Furthermore, $\nabla_i\mathbf{H}_{ab}$ is the derivative of the Hamiltonian block $\mathbf{H}_{ab}$ coupling atoms a and b w.r.t variations along the i=x, y, or z coordinate of the bond $\mathbf{R}_b-\mathbf{R}_a$. To obtain the retarded components of the scattering self-energies, the following relationship can be used: $\Sigma^R \approx (\Sigma^> - \Sigma^<)/2$, which is also valid for $\mathbf{\Pi}^R$[14]. Due to computational reasons, only the diagonal blocks of $\Sigma^{R\gtrless,S}$ are retained, while N_B non-diagonal connections are kept for $\mathbf{\Pi}^{R\gtrless,S}$.

Table 2: State of the Art Quantum Transport Simulators

Name	Maximum # of Computed Atoms						Scalability	
	Tight-binding-like*			DFT			Max. Cores	Using
	$GF_e^\dagger$	$GF_{ph}^\dagger$	$GF + SSE$	$GF_e^\dagger$	$GF_{ph}^\dagger$	$GF + SSE$	(Magnitude)	GPUs
GOLLUM [8]	1k	1k	—	100	100	—	N/A	✗
Kwant [10]	10k	—	—	—	—	—	N/A	✗
NanoTCAD ViDES [18]	10k	—	—	—	—	—	N/A	✗
QuantumATK [24]	10k	10k	—	1k	1k	—	1k	✗
TB_sim [9]	100k	—	10k‡	1k	—	—	10k	✓
NEMO5 [11]	100k	100k	10k‡	—	—	—	100k	✓
OMEN [17]	100k (1.44 Pflop/s [16])	100k	10k	10k	10k (15 Pflop/s [2])	1k (0.16 Pflop/s)	100k	✓
This work	N/A	N/A	N/A	10k	10k	**10k (19.71 Pflop/s)**	**1M**	✓

*: including Maximally-Localized Wannier Functions (MLWF), †: Ballistic, ‡: Simplified.

The evaluation of Eqs. (3-5) does not require the knowledge of all entries of the **G** and **D** matrices, but of two (lesser and greater) 5-D tensors of shape $[N_{k_z}, N_E, N_A, N_{orb}, N_{orb}]$ for electrons and two 6-D tensors of shape $[N_{q_z}, N_\omega, N_A, N_B + 1, N_{3D}, N_{3D}]$ for phonons. Each $[k_z, E, N_A, N_{orb}, N_{orb}]$ and $[q_z, \omega, N_A, N_B + 1, N_{3D}, N_{3D}]$ combination is produced independently from the other by solving Eq. (1) and (2), respectively. The electron and phonon scattering self-energies can also be reshaped into multi-dimensional tensors that have exactly the same dimensions as their Green's functions counterparts. However, the self-energies cannot be computed independently, one energy-momentum or frequency-momentum pair depending on many others, as defined in Eqs. (3-5) and depicted in Fig. 2. Furthermore, $\Sigma^{\gtrless S}(E, k_z)$ is a function of $\mathbf{D}^{\gtrless}(\omega, q_z)$, while $\mathbf{G}^{\gtrless S}(E, k_z)$ is needed to calculate $\mathbf{\Pi}^{\gtrless}(\omega, q_z)$.

To obtain the electrical and energy currents that flow through a given device and the corresponding charge density, Eqs. (1-2) (GF) and Eqs. (3-5) (SSE) must be iteratively solved until convergence is reached, and all GF contributions must be accumulated [22]. The algorithm starts by setting $\Sigma^{\gtrless}(E, k_z)$=$\mathbf{\Pi}^{\gtrless}(\omega, q_z)$=0 and continues by computing all GFs under this condition. The latter then serve as inputs to the next phase, where the SSE are evaluated for all (k_z,E) and (q_z,ω) pairs. Subsequently, the SSE matrices are fed into the GF calculation and the process repeats itself until the GF variations do not exceed a pre-defined threshold. In terms of HPC, the main challenges reside in the distribution of these quantities over thousands of compute units, the resulting communication-intensive gathering of all data to handle the SSE phase, and the efficient solution of Eqs. (3-5) on hybrid nodes, as they involve many small matrix multiplications. Typical simulation parameters are listed in Table 1.

2.1 Current State of the Art

There exist several atomistic quantum transport simulators [8–11, 17, 18, 24] that can model the characteristics of nano-devices. Their performance is summarized in Table 2, where their estimated maximum number of atoms that can be simulated for a given physical model is provided. Only orders of magnitude are shown, as these quantities depend on the device geometries and bandstructure method. It should be mentioned that most tools are limited to tight-binding-like (TB) Hamiltonians, because they are computationally cheaper than DFT ones ($N_{orb,TB} < N_{orb,DFT}$ and $N_{B,TB} \ll N_{B,DFT}$). This explains the larger systems that can be treated with TB. However, such approaches lack accuracy when it comes to the exploration of material stacks, amorphous layers, metallic contacts, or interfaces. In these cases, only DFT ensures reliable results, but at much higher computational cost.

To the best of our knowledge, the only tool that can solve Eqs. (1) to (5) self-consistently, in structures composed of thousands of atoms, at the DFT level is OMEN, a two times Gordon Bell Prize finalist [2, 16].[1] The code is written in C++, contains 90,000 lines of code in total, and uses MPI as its communication protocol. Some parts of it have been ported to GPUs using the CUDA language and taking advantage of libraries such as cuBLAS, cuSPARSE, and MAGMA. The electron-phonon scattering model was first implemented based on the tight-binding method and a three-level MPI distribution of the workload (momentum, energy, and spatial domain decomposition). A first release of the model with equilibrium phonon ($\mathbf{\Pi}$=0) was validated up to 95k cores for a device with N_A=5,402, N_B=4, N_{orb}=10, N_{k_z}=21, and N_E=1,130. These runs showed that the application can reach a parallel efficiency of 57%, when going from 3,276 up to 95,256 cores, with the SSE phase consuming from 25% to 50% of the total simulation times. The reason for the SSE increase could be attributed to the communication time required to gather all Green's Function inputs for Eq. (3), which grew from 16 to 48% of the total simulation time [15] as the number of cores went from 3,276 to 95,256.

After extending the electron-phonon scattering model to DFT and adding phonon transport to it, it has been observed that the time spent in the SSE phase (communication and computation) explodes. Even for a small structure with N_A=2,112, N_{orb}=4, N_{k_z}=N_{q_z}=11, N_E=650, N_ω=30, and N_B=13, 95% of the total simulation time is dedicated to SSE, regardless of the number of used cores/nodes, among which ~60% for the communication between the different MPI tasks. To simulate self-heating in realistic FinFETs (N_A >10,000), with a high accuracy (N_{k_Z} >20, N_E >1,000), and within reasonable times (a couple of minutes for one GF-SSE iteration at machine scale), the algorithms involved in the solution of Eqs. (1) to (5) must be drastically improved: as compared to the state of the art, an improvement of at least one order of magnitude is needed in terms of the number of atoms that can be handled, and two orders of magnitude for what concerns the computational time.

[1] Previous achievements: development of parallel algorithms to deal with ballistic transport (Eq. (1) alone) expressed in a tight-binding (SC11 [16]) or DFT (SC15 [2]) basis.

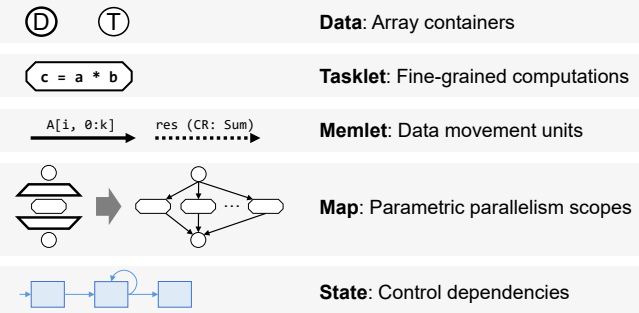


Figure 3: SDFG concepts and syntax.

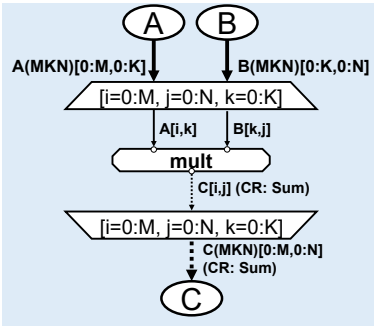


Figure 4: Matrix multiplication in DaCe.

3 DATA-CENTRIC PARALLEL PROGRAMMING

Communication-Avoiding (CA) algorithms [3, 6] are defined as algorithm variants and schedules (orders of operations) that minimize the total number of performed memory loads and stores, achieving lower bounds in some cases. To achieve such bounds, a subset of those algorithms is *matrix-free*[2], potentially reducing communication at the expense of recomputing parts of the data on-the-fly. A key requirement in modifying an algorithm to achieve communication avoidance is to explicitly formulate its data movement characteristics. The schedule can then be changed by reorganizing the data flow to minimize the sum of accesses in the algorithm. Recovering a Data-Centric (DaCe) view of an algorithm, which makes movement explicit throughout all levels (from a single core to the entire cluster), is thus the path forward in scaling up the creation of CA variants to more complex algorithms and multi-level memory hierarchies as one.

DaCe defines a development workflow where the original algorithm is independent from its data movement representation, enabling symbolic analysis and transformation of the latter without modifying the scientific code. This way, a CA variant can be formulated and developed by a performance engineer, while the original algorithm retains readability and maintainability. At the core of the DaCe implementation is the Stateful DataFlow multiGraph (SDFG) [1], an intermediate representation that encapsulates data movement and can be generated from high-level code in Python. The syntax (node and edge types) of SDFGs is listed in Fig. 3. The workflow is as follows: The domain scientist designs an algorithm and implements it as linear algebra operations (imposing dataflow implicitly), or using Memlets and Tasklets (specifying dataflow explicitly). This implementation is then parsed into an SDFG, where performance engineers may apply graph transformations to improve data locality. After transformation, the optimized SDFG is compiled to machine code for performance evaluation. It may be further transformed interactively and tuned for different target platforms and memory hierarchy characteristics.

An example of a naïve matrix multiplication SDFG (`C = A @ B` in Python) is shown in Fig. 4. In the figure, we see that data flows from Data nodes `A` and `B` through a Map scope. This would theoretically expand to `M*N*K` multiplication Tasklets (`mult`), where the contribution of each Tasklet (i.e., a multiplied pair) will be summed in Data node `C` (due to conflicting writes that are resolved by `CR: Sum`). The

[2]The term is derived from solvers that do not need to store the entire matrix in memory.

Memlet edges define all data movement, which is seen in the input and output of each Tasklet, but also entering and leaving the Map with its overall requirements (in brackets) and number of accesses (in parentheses). The accesses and ranges are symbolic expressions, which can be summed to obtain the algorithm's data movement characteristics. The SDFG representation allows the performance engineer to add transient (local) arrays, reshape and nest Maps (e.g., to impose a tiled schedule), fuse multiple scopes, map computations to accelerators (GPUs and FPGAs), and other transformations that may modify the overall number of accesses.

```
1  # Declaration of symbolic variables
2  Nkz, NE, Nqz, Nw, N3D, NA, NB, Norb = (
3      dace.symbol(name)
4      for name in ['Nkz', 'NE', 'Nqz', 'Nw',
5                   'N3D', 'NA', 'NB', 'Norb'])
6
7  @dace.program
8  def sse_sigma(neigh_idx: dace.int32[NA, NB],
9                dH: dace.float64[NA, NB, N3D, Norb, Norb],
10               G: dace.complex128[Nkz, NE, NA, Norb, Norb],
11               D: dace.complex128[Nqz, Nw, NA, NB, N3D, N3D],
12               Sigma: dace.complex128[Nkz, NE, NA, Norb, Norb]):
13
14     # Declaration of Map scope
15     for k, E, q, w, i, j, a, b in dace.map[0:Nkz, 0:NE,
16                                            0:Nqz, 0:Nw,
17                                            0:N3D, 0:N3D,
18                                            0:NA, 0:NB]:
19         f = neigh_idx[a, b]
20         dHG = G[k-q, E-w, f] @ dH[a, b, i]
21         dHD = dH[a, b, j] * D[q, w, a, b, i, j]
22         Sigma[k, E, a] += dHG @ dHD
23
24 if __name__ == '__main__':
25     # Initialize symbolic variables
26     Nkz.set(21)
27     NE.set(1000)
28     ...
29     # Initialize input/output arrays
30     idx = numpy.ndarray((NA.get(), NB.get()), numpy.int32)
31     ...
32     # Call dace program
33     sse_sigma(neigh_idx=idx, dH=dH, G=G, D=D, Sigma=Sigma)
```

Figure 5: $\Sigma^{\gtrless}$ computation in Python

Fig. 5 shows the computation of $\Sigma^{\gtrless}$ in DaCe, implemented with linear algebra operations in a Python-based frontend, while the resulting SDFG is presented in Fig. 8. Symbolic variables, such as the number of atoms, momentums and energies, are declared in lines 2-5. The `dace.program` decorator (line 7) is used to define the function to be converted to an SDFG. Type annotations in the function signature (lines 8-12) are used to define the datatype and shape of the input and output arrays. For-loop statements using the `dace.map` iterator (lines 15-18) define a Map scope. Linear algebra operations (lines 20-22) are automatically parsed to Tasklets. The

latter can be subsequently lowered to nested SDFGs that implement these operations in fine-grained dataflow, such as the matrix multiplication SDFG in Fig. 4. Alternatively, they can be mapped to optimized BLAS calls when generating code. The DaCe program is executed through Python host code (lines 24-33), where the symbolic variables, input and output arrays are initialized.

Our main innovation in optimizing the OMEN QT simulator lies in the use of the DaCe parallel programming framework. In the following section, we show how the data-centric view provided by DaCe is used to identify and implement a *tensor-free* CA variant of OMEN, achieving optimal communication for the first time in this scientific domain.

4 TRANSFORMING OMEN

To understand the dataflow of the OMEN implementation, its 90,000 lines of code, or 15,798 lines[3] of core RGF and SSE computations can be examined. Alternatively, the SDFG could be used to obtain a hierarchical view of the application, where States and Map scopes can be collapsed. A deeper dive allows optimization of certain regions. Below, we take a methodological top-down approach to transform the OMEN SDFG, starting from its high-level decomposition, which generates the communication, through individual computational kernels, to small-scale linear algebra operations. We instrument the code in order to find bottlenecks and critical subgraphs to "cut out" and transform. Furthermore, we support our decisions with communication and performance models obtained using the data-centric representation.

The top-level view of the QT simulation algorithm can be seen in Fig. 6. The SDFG shows that the simulation iterates over two states, GF and SSE. The former computes the Green's Functions, boundary conditions, and the electrical current. The state consists of two concurrent Maps, one for the electrons and one for the phonons (§ 2). The SSE state computes the scattering Self-Energies $\Sigma^{\gtrless}$ and $\mathbf{\Pi}^{\gtrless}$. At this point, we opt to represent the RGF solvers and SSE kernel as Tasklets, i.e., collapsing their dataflow, so as to focus on high-level aspects of the algorithm. This view indicates that the RGF solver cannot compute the Green's Functions for a specific atom separately from the rest of the material (operating on all atoms for a specific

[3]generated using David A. Wheeler's 'SLOCCount'.

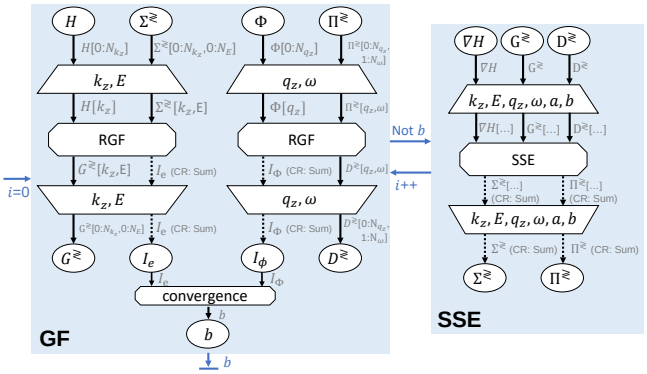


Figure 6: SDFG of QT simulation: high-level performance engineer view of the problem.

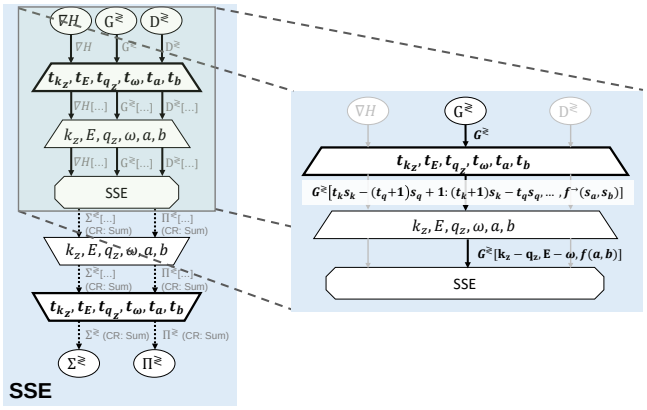


Figure 7: Map-tiling SSE (left) and resulting Memlets (right).

energy-momentum pair), and that SSE outputs the contribution of a specific $(k_z, E, q_z, \omega, a, b)$ point to $\Sigma^{\gtrless}$ and $\mathbf{\Pi}^{\gtrless}$. These contributions are then accumulated to the output tensors, as indicated by the dotted Memlet edges. The accumulation is considered associative; therefore the map can compute all dimensions of the inputs and outputs in parallel.

4.1 Communication Avoidance

The applications shown in Table 2, including OMEN, have been developed mainly by domain scientists, and thus use the "natural" decomposition construction of momentum points and energies, as shown in Fig. 6. As a result, the communication scheme for SSE in OMEN is split to $N_{q_z} N_\omega$ rounds. In each round:

- The phonon Green's Functions $\mathbf{D}^{\gtrless}(\omega, q_z)$ are broadcast to all processes;
- Each process iterates over its assigned electron Green's Functions $\mathbf{G}^{\gtrless}(E, k_z)$, and sends the corresponding $\mathbf{G}^{\gtrless}(E \pm \hbar\omega, k_z + q_z)$ to the processes that need them;
- Each process iterates over its assigned electron Green's Functions $\mathbf{G}^{\gtrless}(E, k_z)$ and receives the corresponding $\mathbf{G}^{\gtrless}(E \pm \hbar\omega, k_z - q_z)$;
- The partial phonon self-energies $\Pi_p^{\gtrless}(\omega, q_z)$ produced by each process are reduced to $\mathbf{\Pi}^{\gtrless}(\omega, q_z)$.

Based on the above, we make the following observations:

- The full 6-D tensors $\mathbf{D}^{\gtrless}$ are broadcast to all processes;
- The full 5-D tensors $\mathbf{G}^{\gtrless}$ are replicated through point-to-point communication $2N_{q_z} N_\omega$ times.

We use DaCe to transform the SSE state and find optimal data distributions and communication schemes in the following manner: First, we tile the SSE map (Fig. 7 left, differences highlighted in bold) across all dimensions, with the intention of assigning each tile to a different process. The tiling graph transformation splits a map to two nested ones, where each dimension of the original map is partitioned to n_d approximately equal ranges of size s_d. For example, the electron momentum dimension is partitioned to n_{k_z} ranges of size s_{k_z} each. The corresponding symbol t_{k_z} in the outer map spans the partitions, whereas the inner symbol k_z takes

values in the range $\left[t_{k_z} s_{k_z}, \left(t_{k_z}+1\right) s_{k_z}\right)$. Likewise, q_z iterates over $\left[t_{q_z} s_{q_z}, \left(t_{q_z}+1\right) s_{q_z}\right)$.

Subsequently, the DaCe framework propagates the data access expressions in Memlets from the Tasklets outwards, through scopes. DaCe automatically computes contiguous and strided ranges, but can only over-approximate some irregular accesses. In these cases, performance engineers can manually provide the additional information to the SDFG, creating new optimization opportunities.

In particular, the propagation of the access $\mathbf{G}^{\gtrless}[k_z - q_z, E - \omega, f(a,b)]$ is shown in Fig. 7 (right). The propagated range of the index expression $k_z - q_z$ is computed automatically as $[t_{k_z} s_{k_z} - \left(t_{q_z}+1\right) s_{q_z}+1, \left(t_{k_z}+1\right) s_{k_z} - t_{q_z} s_{q_z})$. The total number of accesses over this range is $s_{k_z} + s_{q_z} - 1$, while the length, which coincides with the number of unique accesses, is $\min\left(N_{k_z}, s_{k_z} + s_{q_z} - 1\right)$. However, the expression $f(a,b)$, which represents the index of the b-th neighbor of atom a, is an indirection through a matrix of the atom couplings. DaCe cannot propagate such indices and thus the performance engineer must provide a model or expression manually.

For this work, we do not to tile the dimensions of the atom neighbors. Instead, we make use of the observation that atoms with neighboring indices are very often neighbors in the coupling matrix. A good approximation to the propagation of $f(a,b)$ over the range $[t_a s_a, (t_a+1) s_a) \times [0, N_B)$ is then $\Big[\min\left(0, t_a s_a - \frac{N_B}{2}\right), \max\left(N_A, (t_a+1) s_a + \frac{N_B}{2}\right)\Big)$. The total number of accesses increases to $s_a N_B$, while the length of this range is $\min(N_A, s_a + N_B)$.

After Memlet propagation is complete, the total length of the Memlet ranges between the two maps provides the amount of data that each process must load/store or communicate over the network. An ***optimal communication scheme*** can subsequently be found by minimizing these expressions. For this work, we perform exhaustive search over the feasible tile sizes. Since the combinations of the latter are in the order of 10^6 for most simulation parameters and number of processes, the search completes in just a few seconds.

We demonstrate the power of the above approach by comparing the OMEN communication scheme against partitioning the atom and electron-energy dimensions. Using the original OMEN data distribution, each process:

- receives $64 \frac{N_{k_z} N_E}{P} N_{q_z} N_\omega N_A N_{orb}^2$ bytes for the electron Green's Functions $\mathbf{G}^{\gtrless}$;
- sends and receives a total of $64 N_{q_z} N_\omega N_A N_B N_{3D}^2$ bytes for the phonon Green's functions $\mathbf{D}^{\gtrless}$ and self-energies $\mathbf{\Pi}^{\gtrless}$;

where P is the number of processes. The DaCe-transformed SDFG changes the distribution of the data between the GF and SSE states, which yields all-to-all collective operations (`alltoallv` in the MPI standard). Specifically, each process contributes:

- $64 N_{k_z} \left(\frac{N_E}{T_E} + 2N_\omega\right)\left(\frac{N_A}{T_A} + N_B\right) N_{orb}^2$ bytes for the electron Green's functions $\mathbf{G}^{\gtrless}$ and self-energies $\mathbf{\Sigma}^{\gtrless}$;
- $64 N_{q_z} N_\omega \left(\frac{N_A}{T_A} + N_B\right) N_B N_{3D}^2$ bytes for $\mathbf{D}^{\gtrless}$ and $\mathbf{\Pi}^{\gtrless}$.

T_E and T_A are the number of partitions of the energies and atoms respectively, with $P = T_E T_A$. For $\mathbf{D}^{\gtrless}$ and $\mathbf{\Pi}^{\gtrless}$, the DaCe-based communication scheme reduces the factor $N_A N_B$ to $\frac{N_A}{T_A} + N_B$. In the case of $\mathbf{G}^{\gtrless}$, this scheme eliminates the quadratic factor over the number of momentum points exhibited by OMEN.

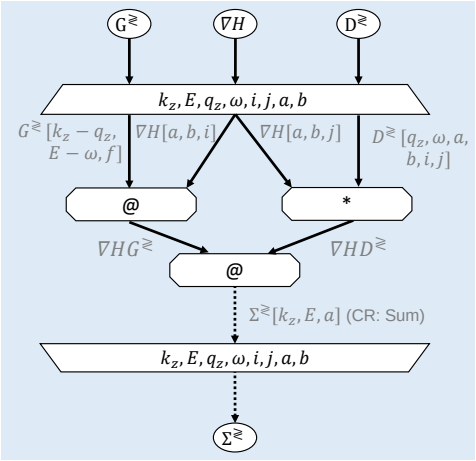


Figure 8: Initial SDFG of $\Sigma^{\gtrless}$ computation in SSE.

4.2 Dataflow Optimizations

The data-centric view not only encompasses macro dataflow that imposes communication, but also data movement within compute devices. We use DaCe to transform all computations in the communication-avoiding variant of OMEN, including the RGF algorithm, SSE, and boundary conditions, and automatically generate GPU code. Below we cut-out and showcase a subset of these transformations, focusing on a bottleneck subgraph of the QT simulator, which is found within the SSE kernel: computing $\Sigma^{\gtrless}$ as in Eq. (3). We note that computation of $\mathbf{\Pi}^{\gtrless}$ is transformed in a similar manner.

Fig. 8 gives the initial representation of the computation, generated from a reference Python implementation. The inputs are:

- $\mathbf{G}^{\gtrless}$: Electron Green's Functions, a 3-D array of N_{orb}^2 matrices and size $N_{k_z} \times N_E \times N_A$;
- $\nabla\mathbf{H}$: Derivative of the Hamiltonian, a 3-D array of N_{orb}^2 matrices and size $N_A \times N_B \times N_{3D}$;
- $\mathbf{D}^{\gtrless}$: Phonon Green's Functions, a 6-D array of scalar values and size $N_{q_z} \times N_\omega \times N_A \times N_B \times N_{3D}^2$. Prior to the kernel, the Green's Functions have been preprocessed to contain the values $\mathbf{D}_{ln}^{\gtrless ij}(\omega, q_z) - \mathbf{D}_{ll}^{\gtrless ij}(\omega, q_z) - \mathbf{D}_{nn}^{\gtrless ij}(\omega, q_z) + \mathbf{D}_{nl}^{\gtrless ij}(\omega, q_z)$, as described in Eq. (3).

The outputs are the electron self-energies $\mathbf{\Sigma}^{\gtrless}$, which are also a 3-D array of N_{orb}^2 matrices with the same dimensions as $\mathbf{G}^{\gtrless}$. The SDFG consists of a map over the 8-D space $\left[0, N_{k_z}\right) \times [0, N_E) \times \left[0, N_{q_z}\right) \times [0, N_\omega) \times [0, N_{3D}) \times [0, N_{3D}) \times [0, N_A) \times [0, N_B)$. For each $(k_z, E, q_z, \omega, i, j, a, b)$ point in this space, the following computations must be performed:

(1) The matrices at indices $\mathbf{G}^{\gtrless}[k_z - q_z, E - \omega, f]$ and $\nabla\mathbf{H}[a, b, i]$ are multiplied ("@" symbol) and the result is stored in the temporary matrix $\nabla\mathbf{H}\mathbf{G}^{\gtrless}$. The index f in the array $\mathbf{G}^{\gtrless}$ is an indirection $f(a,b)$ in the space $[0, N_A)$;

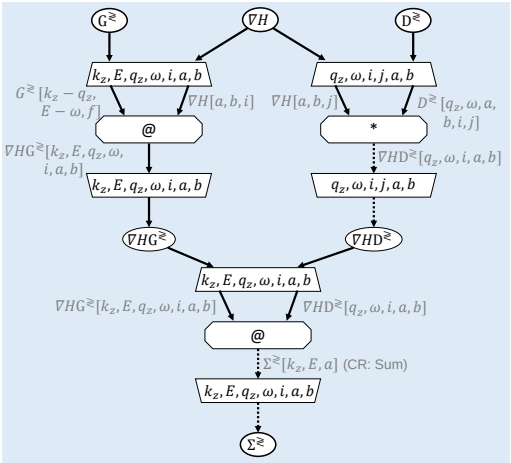


Figure 9: $\Sigma^{\gtrless}$ SDFG after applying Map Fission.

(2) The matrix at index $\nabla\mathbf{H}[a, b, j]$ is multiplied by the scalar value $\mathbf{D}^{\gtrless}[q_z, \omega, a, b, i, j]$ ("$*$" symbol) and the result is stored in the temporary matrix $\nabla\mathbf{HD}^{\gtrless}$;
(3) The product of the temporary matrices $\nabla\mathbf{HG}^{\gtrless}$ and $\nabla\mathbf{HD}^{\gtrless}$ is accumulated (dotted edges) to the matrix $\Sigma^{\gtrless}[k_z, E, a]$.

To optimize the SDFG, we first isolate the three computations described above. This is achieved by applying the Map Fission (distribution) transformation, as shown in Fig. 9. The transformation splits the map into three separate ones, where each one operates over a subset of the original space. As a result, it automatically detects that the top-left and bottom maps are independent of the j symbol, and removes it from them. Likewise, k_z and E are excluded from the top-right map. Furthermore, it substitutes the temporary matrices $\nabla\mathbf{HG}^{\gtrless}$ and $\nabla\mathbf{HD}^{\gtrless}$ with multi-dimensional tensors, that store all the intermediate results of two top maps.

We proceed with the optimization of the top-left map, enlarged in Fig. 10a. In the subgraph, the symbols (q_z, ω) (highlighted) are only used as offsets to the indices (k_z, E) of $\mathbf{G}^{\gtrless}$. Therefore, the subspace $[0, N_{k_z}) \times [0, N_E)$ already covers all $(k_z - q_z, E - \omega)$ points. The iteration over the subspace $[0, N_{q_z}) \times [0, N_\omega)$ (q_z and ω) results in redundant computation, and is removed in Fig. 10b. The two corresponding dimensions are also removed from $\nabla\mathbf{HG}^{\gtrless}$.

At this point, the matrices $\nabla\mathbf{H}[a, b, i]$ are used multiple times inside the map (highlighted in Fig. 10b), a fact that can be exploited. However, the matrices $\mathbf{G}^{\gtrless}[k_z, E, f]$ are accessed irregularly, since f is in this case an indirection $f(a, b)$. This irregularity is treated by a data-layout transformation on $\mathbf{G}^{\gtrless}$ and $\nabla\mathbf{HG}^{\gtrless}$ (Fig. 10c). Now that the inner dimensions of the arrays are accessed continuously over (k_z, E) (highlighted), we combine the $N_{k_z}N_E$ matrix multiplications of size $N_{orb} \times N_{orb} \times N_{orb}$ in Fig. 10d to a single $N_{orb} \times N_{orb} \times N_{k_z}N_EN_{orb}$ operation, with better performance characteristics.

Our next optimization target is the third computation ($\Sigma^{\gtrless}$) in the SSE kernel, found in the bottom map enlarged in Fig. 11a. In the figure, both input tensors are accessed in a continuous manner over ω. In Fig. 11b we apply Map Expansion to create a nested map over the space $[0, N_\omega)$. The nested map performs the accumulation (showing only the inner indices) $\Sigma^{\gtrless}[E]$ += $\sum_\omega\{\nabla\mathbf{HG}^{\gtrless}[E - \omega] \cdot \nabla\mathbf{HD}^{\gtrless}[\omega]\}$, which can be rewritten as $\Sigma^{\gtrless}[E]$ += $\nabla\mathbf{HG}^{\gtrless}[E - \omega : E] \cdot \nabla\mathbf{HD}^{\gtrless}[:]^T$. In Fig. 11c we substitute the nested map with a single $N_{orb} \times N_{orb}N_\omega \times N_{orb}$ GEMM operation, which typically performs better than the individual small matrix multiplications.

Since the introduced transient tensors consist of multiple dimensions, their overall size may grow rapidly. Therefore, the last optimization step involves reducing the memory footprint of the kernel. We achieve this in two steps. First, we expand the $[0, N_A) \times [0, N_B)$ space in each of the separate maps; and second, we combine the three separate (outer) maps back to a single one with Map Fusion. The result is illustrated in Fig. 12. The transformation reduces the size of the transient arrays to only three dimensions, which are accessed for each iteration (a, b).

Table 3: Single Iteration Computational Load (Pflop Count)

	N_{k_z}				
Kernel	3	5	7	9	11
Contour Integral	8.45	14.12	19.77	25.42	31.06
RGF	52.95	88.25	123.55	158.85	194.15
SSE (OMEN)	24.41	67.80	132.89	219.67	328.15
SSE (DaCe)	12.38	34.19	66.85	110.36	164.71

4.3 Performance Model

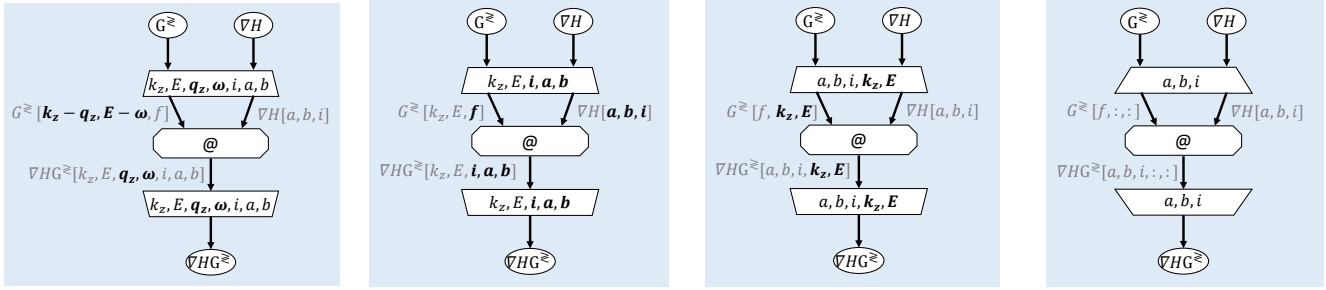
The majority of computations in the SDFG revolves around three kernels: (a) Contour Integration, computation of the open boundary conditions; (b) Recursive Green's Function (RGF); and (c) the SSE kernel. The first two kernels represent most of the computational load in the GF phase, while the SSE phase comprises the SSE kernel.

The kernels of the GF phase involve mostly matrix multiplications. Therefore, the computational complexity of the RGF algorithm is $O\left(N_A^3N_{orb}^3\right)$ for each (E,k_z) pair and $O\left(N_{k_z}N_EN_A^3N_{orb}^3\right)$ for the entire grid. Due to the GF phase kernels using both dense and sparse matrices, it is difficult to obtain an exact flop count using analytical expressions. We overcome this issue by counting GPU flop with the NVIDIA profiler `nvprof`, since most of the computations occur on the GPU.

For the SSE phase, described in detail in the previous section, the complexity of the multitude of small matrix multiplications (sized $N_{orb} \times N_{orb}$) is equal to $O\left(N_{k_z}N_EN_{q_z}N_\omega N_AN_BN_{orb}^3\right)$. Since the operations only involve dense matrices, the flop count for the original OMEN algorithm is $64N_AN_BN_{3D}N_{k_z}N_{q_z}N_EN_\omega N_{orb}^3$. The data-centric transformations performed on the algorithm reduce it to $32N_AN_BN_{3D}N_{k_z}N_{q_z}N_EN_\omega N_{orb}^3 + 32N_AN_BN_{3D}N_{k_z}N_EN_{orb}^3$. Table 3 shows the flop values, empirical and analytical, for a Silicon structure with $N_A = 4{,}864$, $N_B = 34$, $N_E = 706$ and $N_\omega = 70$, for varying N_{k_z} values, corresponding to a structure with $W = 2.1$nm and $L = 35$nm.

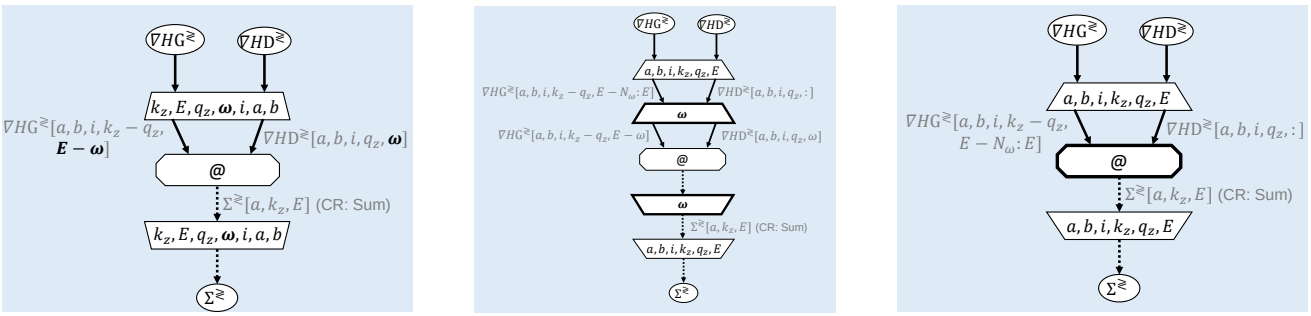
5 PERFORMANCE EVALUATION

We proceed to evaluate the performance of the data-centric OMEN algorithm. Starting with microbenchmarks, we demonstrate the necessity of a high-performance implementation and that critical portions of the algorithm deliver close-to-optimal performance on the underlying systems. We then measure performance aspects of OMEN and the DaCe variant on a large-scale problem consisting



(a) $\nabla HG^{\gtrless}$ computation. (b) Redundancy removal. (c) Data-layout transformation. (d) Multiplication fusion.

Figure 10: Transformation progression on the first part of the SSE kernel (computing $\nabla HG^{\gtrless}$).



(a) Continuous access over ω. (b) Map expansion. (c) Nested map substitution.

Figure 11: Transformation progression on the third part of the SSE kernel (computing $\Sigma^{\gtrless}$).

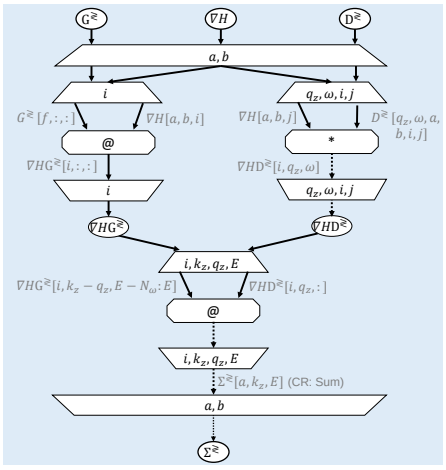


Figure 12: Optimized SSE $\Sigma^{\gtrless}$ SDFG.

of 4,864 atoms, between 22 and 5,400 nodes. Lastly, we run on the full extent of a supercomputer, measuring the heat dissipation of a 10,240 atom nanodevice with $W = 4.8$nm and $L = 35$nm. All DFT input parameters in Eqs. (1-2) were created with CP2K and rely therefore on Gaussian-type orbitals (GTO). A 3SP basis set was used to model all atoms. The choice of the exchange correlation function (LDA) has no influence on the computational efficiency.

The two systems we use are CSCS Piz Daint [4] (6th place in June's 2019's Top500 supercomputer list) and OLCF Summit [7] (1st place). Piz Daint is composed of 5,704 Cray XC50 compute nodes,

Table 4: Weak Scaling of SSE Communication Volume (TiB)

Algorithm / Variant	N_{k_z} (Processes) 3 (768)	5 (1280)	7 (1792)	9 (2304)	11 (2816)
OMEN	32.11	89.18	174.80	288.95	431.65
DaCe	0.54	1.22	2.17	3.38	4.86

$N_A = 4{,}864$, $N_B = 34$, $N_{orb} = 12$, $N_E = 706$, $N_\omega = 70$.

each equipped with a 12-core HT-enabled (2-way SMT) Intel Xeon E5-2690 CPU with 64 GiB RAM, and one NVIDIA Tesla P100 GPU. The nodes communicate using Cray's Aries interconnect. Summit comprises 4,608 nodes, each containing two IBM POWER9 CPUs (21 usable physical cores with 4-way SMT) with 512 GiB RAM and six NVIDIA Tesla V100 GPUs. The nodes are connected using Mellanox EDR 100G InfiniBand organized in a Fat Tree topology. For Piz Daint, we run our experiments with two processes per node (sharing the GPU), apart from a full-scale run on 5,400 nodes, where the simulation parameters do not produce enough workload for more than one process per node. In Summit we run with six processes per node, each consuming 7 physical cores.

We conduct every experiment at least 5 times (barring extreme-scale runs), and report the median result and 95% Confidence Interval as error bars.

5.1 Microbenchmarks

Below we discuss the communication aspect of SSE, followed by computational aspects of GF. We also evaluate the single-node performance of the different OMEN implementations.

Table 5: Strong Scaling of SSE Communication Volume (TiB)

Algorithm Variant	Processes				
	224	448	896	1792	2688
OMEN	108.24	117.75	136.76	174.80	212.84
DaCe	0.95	1.13	1.48	2.17	2.87

$N_A = 4{,}864$, $N_B = 34$, $N_{orb} = 12$, $N_{k_z} = 7$, $N_E = 706$, $N_\omega = 70$.

5.1.1 SSE Communication Pattern. In Tables 4 and 5 the total communication load for the different implementations is shown, for a Silicon material with $N_A = 4{,}864$, $N_B = 34$, $N_E = 706$ and $N_\omega = 70$. In Table 4, the number of processes increases relatively to N_{k_z}. The tiling parameters (§ 4.1) of the DaCe implementation are $T_E = N_{k_z}$ and $T_A = 7$. In Table 5, we fix N_{k_z} to 7 and vary the number of processes. T_E is always 7 and T_A is equal to $32\frac{P}{112}$. The tables both show clear advantage of using the communication-avoiding variant of the algorithm, with up to two orders of magnitude speedup.

5.1.2 Green's Functions and Sparsity. Since the RGF algorithm uses a combination of sparse and dense matrices, there are several paths that can be taken for computing their multiplication with each other. In particular, a common operation in RGF is `F[n] @ gR[n + 1] @ E[n + 1]` — multiplying two sparse blocks of the block tri-diagonal Hamiltonian matrix (`E`, `F`) with a retarded Green's Functions block (`gR`). To perform this operation, one might (a) use CSR-to-dense conversion followed by dense multiplication (*Dense-MM*); (b) multiply the first CSR matrices with the dense to obtain a dense matrix, followed by a transposed dense-CSR multiplication (*CSRMM*); or (c) multiply all matrices as sparse, keeping the result (and thus `gR`) sparse (*CSRGEMM*). The first two options can be interchanged via data-centric transformations.

Table 6: Sparse vs. Dense 3-Matrix Multiplication in RGF

Approach	Dense-MM	CSRMM	CSRGEMM
Time [ms]	203.59 ± 5.95	**47.06** ± 0.15	93.02 ± 0.21

In Table 6 we study the performance of all three approaches for representative sizes and matrix sparsity, using cuSPARSE for the operations. All implementations use multiple CUDA streams (as a result of SDFG scheduling) and thus pipeline CPU-to-GPU copies and computation. From the table, the best performance is attained with the *CSRMM* approach, with 1.98–4.33× speedup.

5.1.3 Single-Node Performance. We evaluate the performance of OMEN, the DaCe variant, and the Python reference implementation (using the numpy module implemented over MKL), on a Silicon nanostructure with $N_A = 4{,}864, N_B = 34, N_{k_z} = 3, N_E = 706$ and $N_\omega = 70$. In Table 7 the runtime of the GF and SSE SDFG states is shown, for $\frac{1}{112}$ of the total computational load, executed by a single node on Piz Daint. Although Python uses optimized routines, it exhibits very slow performance on its own. This is a direct result of using an interpreter for mathematical expressions, where arrays are allocated at runtime and each operation incurs high overheads. This can especially be seen in SSE, which consists of many small multiplication operations. The table also shows that the data-centric transformations made on the Python code using DaCe outperforms the manually-tuned C++ OMEN on both phases, where the performance-oriented reconstruction of SSE generates a speedup of 9.97×.

Table 7: Single-Node Runtime (Seconds)

Phase	Algorithm Variant		
	OMEN	Python	DaCe
GF	144.14	1,342.77	**111.25**
SSE	965.45	30,560.13	**96.79**

5.2 Scalability

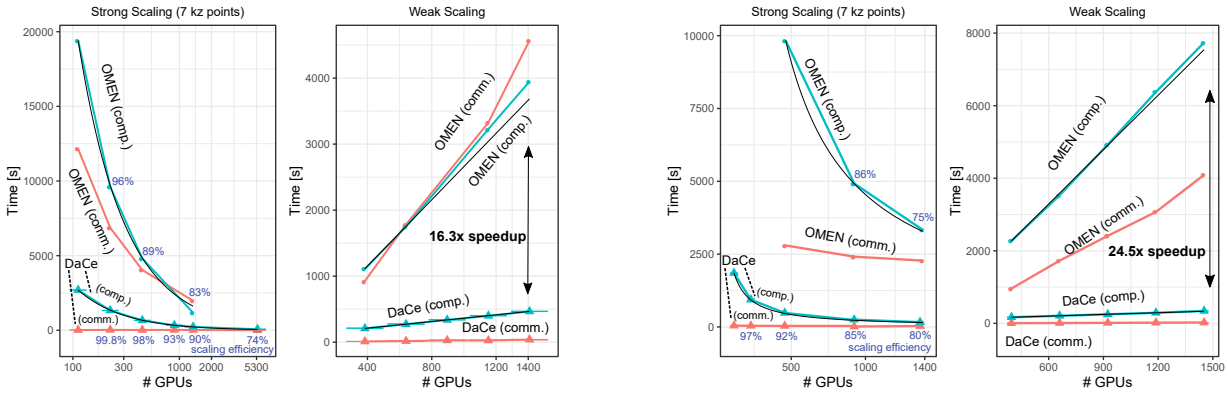
The communication-avoiding variant of OMEN (*DaCe OMEN*) exhibits strong scaling on both supercomputers. In Fig. 13, we measure the runtime and scalability of a single iteration of OMEN and the DaCe variant on Piz Daint and Summit. For strong scaling, we set a fixed nanostructure with 4,864 atoms and $N_{k_z} = 7$ (so that OMEN can treat it), using 112–5,400 nodes on Piz Daint and 19–228 nodes (114–1,368 GPUs) on Summit. Instead of linear scaling, we annotate ideal weak scaling (in black) with proportional increases in the number of k_z points and nodes, since the GF and SSE phases scale differently relative to the simulation parameters (by N_{k_z} and $N_{k_z}N_{q_z} = N_{k_z}^2$ respectively). We measure the same nanostructure with varying k_z points: $N_{k_z} \in \{3, 5, 7, 9, 11\}$, using 384–1,408 nodes on Piz Daint and 66–242 nodes (396–1,452 GPUs) on Summit.

Compared with the original OMEN, the DaCe variant is efficient, both from the computation and communication aspects. On Piz Daint, the total runtime of the reduced-communication variant **outperforms OMEN, the current state of the art, up to a factor of 16.3×**, while the communication time **improves by up to 417.2×**. On Summit, the total runtime **improves by up to factor of 24.5×**, while communication is **sped up by up to 79.7×**.

Observe that on Summit, the speedup of the computational runtime is higher than on Piz Daint. This is the result of OMEN depending on multiple external libraries, some of which are not necessarily optimized for every architecture (e.g., IBM POWER9). On the other hand, SDFGs are compiled on the target architecture and depend only on a few optimized libraries provided by the architecture vendor (e.g., MKL, cuBLAS, ESSL), whose implementations can be replaced by SDFGs for further tuning and transformations.

As for scaling, on Summit DaCe OMEN achieves a total speedup of 9.68× on 12 times the nodes in the strong scaling experiment (11.23× for computation alone). Piz Daint yields similar results with 10.69× speedup. The algorithm weakly scales with N_{k_z} on both platforms, again an order of magnitude faster than the state of the art. We can thus conclude that the data-centric transformed version of OMEN is strictly desirable over the C++ version.

5.2.1 Extreme-Scale Run. We run DaCe OMEN on a setup that is not possible on the original OMEN, due to infeasible memory requirements of the algorithm. We simulate a 10,240 atom, Silicon-based nanostructure — *a size never-before-simulated with DFT/SSE at the ab initio level* — using the DaCe variant of OMEN. For this purpose, we use up to 76.5% of the Summit supercomputer: 21,150 GPUs, and run our proposed Python code with up to 21 k_z points, which are necessary to produce accurate results. produces accurate results and costs 7,363 petaflop/iteration. This achieves 12% of effective peak performance (44.5% for the GF state and 6.2% for the SSE one), including communication. The simulation costs 7,363 Pflop/iteration, achieving a sustained performance of 19.71 Pflop/s (12.83% of the effective peak), including communication. The results



(a) Piz Daint (b) Summit

Figure 13: DaCe OMEN simulation scalability (N_a = 4,864, black lines: ideal scaling).

are listed in Table 8, proving that the electro-thermal properties of nano-devices of this magnitude can be computed in under 7 minutes per iteration, as required for practical applications. A full-scale run on Summit, with further optimizations, is described by Ziogas et al. [29].

Table 8: Summit Performance on 10,240 Atoms

N_{k_z}	Computation				Comm.
	GF state		SSE state		
(Nodes)	Pflop	Time [s]	Pflop	Time [s]	Time [s]
11 (1852)	2,922	75.84	490	95.46	44.02
15 (2580)	3,985	75.90	910	116.67	43.93
21 (1763)	5,579	150.38	1,784	346.56	121.91
21 (3525)	5,579	76.09	1,784	175.15	122.35

N_A = 10,240, N_B = 34, N_{orb} = 12, N_E = 1,000, N_ω = 70.

6 CONCLUSIONS

This paper shows that modifications to data movement alone can transform a dissipative quantum transport simulation algorithm to become communication-efficient. Through rigorous modeling made possible by a data-centric intermediate representation, and graph transformations on the underlying macro and micro dataflow, this work is the first to introduce communication-avoiding principles to a full application. The algorithm is run on two of the fastest supercomputers, where the performance is increased by up to *two orders of magnitude* over the previous state of the art, measuring heat dissipation of nanodevices with scattering self-energies, 10,240 atoms, and 21 k_z points *for the first time*. These results were obtained from a Python source code containing 3,155 lines of code[4] and an SDFG with 2,015 nodes, all without modifying the original operations. Applying the contributions of this paper on the state of the art C++ code, on the other hand, would require its complete rewrite, due to its tightly-coupled computation and communication modules.

The presented results imply that optimizing data movement separately from the source code can be used to further adapt this algorithm, as well as other physics simulations, to future supercomputers; and to augment quantum transport simulations with additional features, without undoing existing optimizations, enabling better cooling system designs in future microprocessors.

[4]Generated using David A. Wheeler's 'SLOCCount'.

ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 programme (grant agreement DAPP, No. 678880), by the MARVEL NCCR of the Swiss National Science Foundation (SNSF), by SNSF grant 175479 (ABIME), and by a grant from the Swiss National Supercomputing Centre, Project No. s876. This work used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. The authors would like to thank Maria Grazia Giuffreda, Nick Cardo (CSCS), Don Maxwell, Christopher Zimmer, and especially Jack Wells (ORNL) for access and support of the computational resources.

REFERENCES

[1] T. Ben-Nun, J. de Fine Licht, A. N. Ziogas, T. Schneider, and T. Hoefler. 2019. Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures. In *Proc. Int'l Conference for High Performance Computing, Networking, Storage and Analysis*.

[2] Mauro Calderara, Sascha Brück, Andreas Pedersen, Mohammad H. Bani-Hashemian, Joost VandeVondele, and Mathieu Luisier. 2015. Pushing Back the Limit of Ab-initio Quantum Transport Simulations on Hybrid Supercomputers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15)*. ACM, New York, NY, USA, Article 3, 12 pages.

[3] E. Carson, J. Demmel, L. Grigori, N. Knight, P. Koanantakool, O. Schwartz, and H. V. Simhadri. 2016. Write-Avoiding Algorithms. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 648–658. https://doi.org/10.1109/IPDPS.2016.114

[4] Swiss National Supercomputing Centre. 2019. Piz Daint. https://www.cscs.ch/computers/piz-daint/

[5] Supriyo Datta. 1995. *Electronic Transport in Mesoscopic Systems*. Cambridge University Press. https://doi.org/10.1017/CBO9780511805776

[6] J. Demmel. 2013. Communication-avoiding algorithms for linear algebra and beyond. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. 585–585. https://doi.org/10.1109/IPDPS.2013.123

[7] Oak Ridge Leadership Computing Facility. 2019. Summit. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/

[8] Jaime Ferrer, Colin J Lambert, Víctor Manuel García-Suárez, D Zs Manrique, D Visontai, L Oroszlany, Rubén Rodríguez-Ferradás, Iain Grace, SWD Bailey, Katalin Gillemot, et al. 2014. GOLLUM: a next-generation simulation tool for electron, thermal and spin transport. *New Journal of Physics* 16, 9 (2014), 093029.

[9] CEA Grenoble. 2013. TB_Sim. http://inac.cea.fr/L_Sim/TB_Sim/

[10] Christoph W Groth, Michael Wimmer, Anton R Akhmerov, and Xavier Waintal. 2014. Kwant: a software package for quantum transport. *New Journal of Physics* 16, 6 (2014), 063065.

[11] The Nanoelectronic Modeling Group and Gerhard Klimeck. 2018. NEMO5. https://engineering.purdue.edu/gekcogrp/software-projects/nemo5/

[12] J. Izquierdo, A. Vega, L. C. Balbás, Daniel Sánchez-Portal, Javier Junquera, Emilio Artacho, Jose M. Soler, and Pablo Ordejón. 2000. Systematic ab initio study of the electronic and magnetic properties of different pure and mixed iron systems. *Phys. Rev. B* 61 (May 2000), 13639–13646. Issue 20. https://doi.org/10.1103/PhysRevB.61.13639

[13] W. Kohn and L. J. Sham. 1965. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.* 140 (Nov 1965), A1133–A1138. Issue 4A. https://doi.org/10.1103/PhysRev.140.A1133

[14] Roger Lake, Gerhard Klimeck, R. Chris Bowen, and Dejan Jovanovic. 1997. Single and multiband modeling of quantum electron transport through layered semiconductor devices. *Journal of Applied Physics* 81, 12 (1997), 7845–7869. https://doi.org/10.1063/1.365394

[15] M. Luisier. 2010. A Parallel Implementation of Electron-Phonon Scattering in Nanoelectronic Devices up to 95k Cores. In *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–11. https://doi.org/10.1109/SC.2010.6

[16] Mathieu Luisier, Timothy B. Boykin, Gerhard Klimeck, and Wolfgang Fichtner. 2011. Atomistic Nanoelectronic Device Engineering with Sustained Performances Up to 1.44 PFlop/s. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. ACM, New York, NY, USA, Article 2, 11 pages.

[17] Mathieu Luisier, Andreas Schenk, Wolfgang Fichtner, and Gerhard Klimeck. 2006. Atomistic simulation of nanowires in the $sp^3d^5s^*$ tight-binding formalism: From boundary conditions to strain calculations. *Phys. Rev. B* 74 (Nov 2006), 205323. Issue 20. https://doi.org/10.1103/PhysRevB.74.205323

[18] NanoTCAD. 2017. ViDES. http://vides.nanotcad.com/vides/

[19] Patrick McCormick. 2019. Yin & Yang: Hardware Heterogeneity & Software Productivity . Talk at SOS23 meeting, Asheville, NC.

[20] Robert Pawlik. 2016. Current CPUs produce 4 times more heat than hot plates. https://www.cloudandheat.com/blog/current-cpus-produce-4-times-more-heat-than-hot-plates-future-performance-increases-only-possible-by-using-direct-water-cooling/

[21] E. Pop, S. Sinha, and K. E. Goodson. 2006. Heat Generation and Transport in Nanometer-Scale Transistors. *Proc. IEEE* 94, 8 (Aug 2006), 1587–1601. https://doi.org/10.1109/JPROC.2006.879794

[22] Christian Stieger, Aron Szabo, Teutë Bunjaku, and Mathieu Luisier. 2017. Ab-initio quantum transport simulation of self-heating in single-layer 2-D materials. *Journal of Applied Physics* 122, 4 (2017), 045708. https://doi.org/10.1063/1.4990384

[23] A. Svizhenko, M. P. Anantram, T. R. Govindan, B. Biegel, and R. Venugopal. 2002. Two-dimensional quantum mechanical modeling of nanotransistors. *Journal of Applied Physics* 91, 4 (2002), 2343–2354. https://doi.org/10.1063/1.1432117

[24] Synopsys. 2019. QuantumATK. https://www.synopsys.com/silicon/quantumatk.html

[25] Atsushi Togo, Fumiyasu Oba, and Isao Tanaka. 2008. First-principles calculations of the ferroelastic transition between rutile-type and $CaCl_2$-type SiO_2 at high pressures. *Phys. Rev. B* 78 (Oct 2008), 134106. Issue 13. https://doi.org/10.1103/PhysRevB.78.134106

[26] D. Unat, A. Dubey, T. Hoefler, J. Shalf, M. Abraham, M. Bianco, B. L. Chamberlain, R. Cledat, H. C. Edwards, H. Finkel, K. Fuerlinger, F. Hannig, E. Jeannot, A. Kamil, J. Keasler, P. H. J. Kelly, V. Leung, H. Ltaief, N. Maruyama, C. J. Newburn, and M. Pericás. 2017. Trends in Data Locality Abstractions for HPC Systems. *IEEE Transactions on Parallel and Distributed Systems* 28, 10 (Oct 2017), 3007–3020. https://doi.org/10.1109/TPDS.2017.2703149

[27] Joost VandeVondele, Matthias Krack, Fawzi Mohamed, Michele Parrinello, Thomas Chassaing, and Jürg Hutter. 2005. Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach. *Computer Physics Communications* 167, 2 (2005), 103 – 128. https://doi.org/10.1016/j.cpc.2004.12.014

[28] J. Wei. 2008. Challenges in Cooling Design of CPU Packages for High-Performance Servers. *Heat Transfer Engineering* 29, 2 (2008), 178–187. https://doi.org/10.1080/01457630701686727

[29] A. N. Ziogas, T. Ben-Nun, G. Indalecio Fernandez, T. Schneider, M. Luisier, and T. Hoefler. 2019. A Data-Centric Approach to Extreme-Scale *Ab initio* Dissipative Quantum Transport Simulations. In *Proc. Int'l Conference for High Performance Computing, Networking, Storage and Analysis*.