Going data-centric in extremely heterogeneous programming
--- the tale of a quantum transport simulation ---

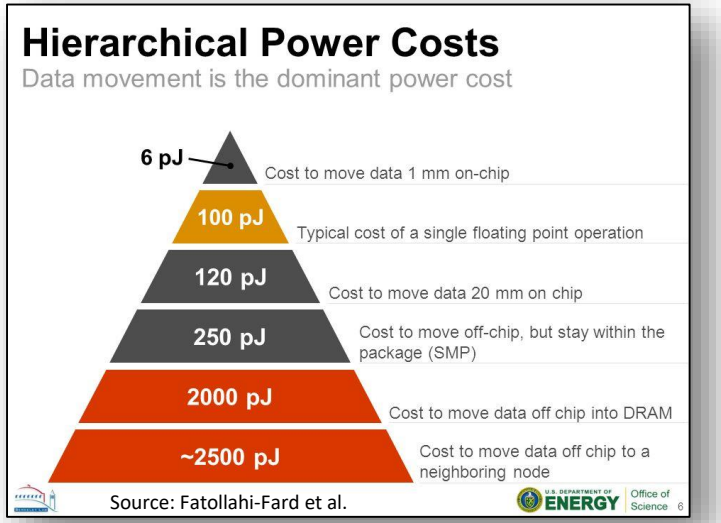
Torsten Hoefler, Random Access at The Salishan Conference on High Speed Computing, OR, USA

Alexandros Ziogas, Tal Ben-Nun, Guillermo Indalecio, Timo Schneider, Mathieu Luisier, and Johannes de Fine Licht
and the whole DAPP team

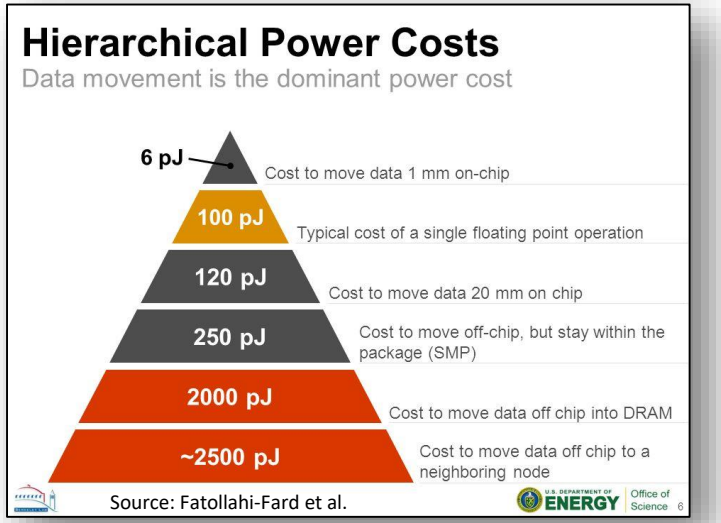
SPCL

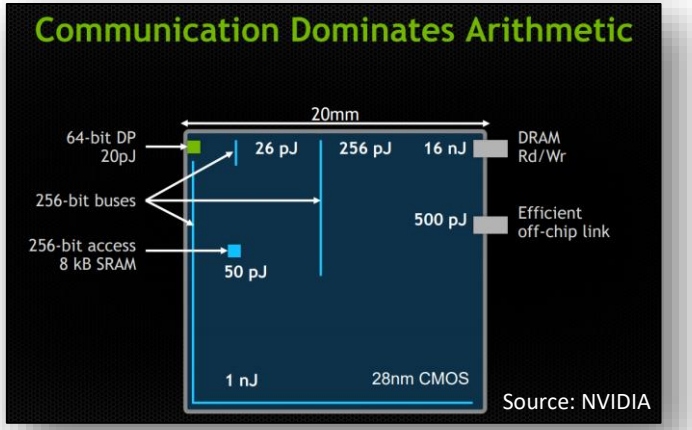
Data movement will dominate everything!

Data movement will dominate everything!

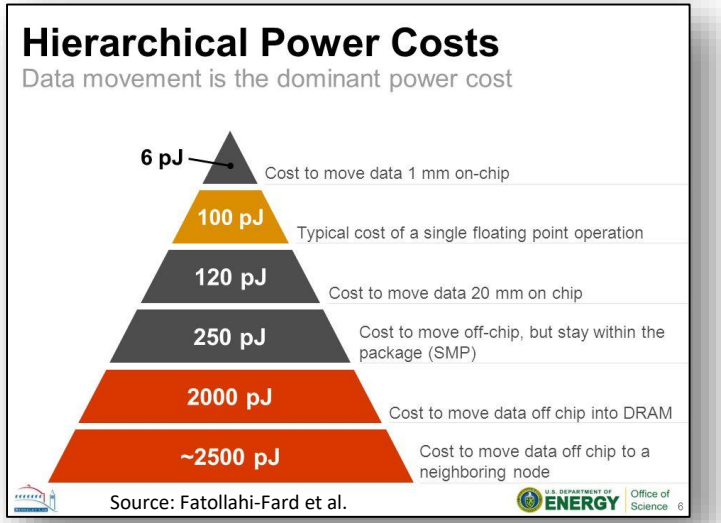


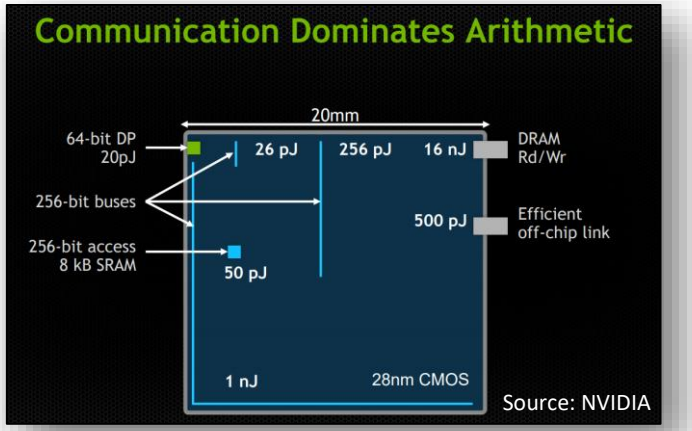
Data movement will dominate everything!

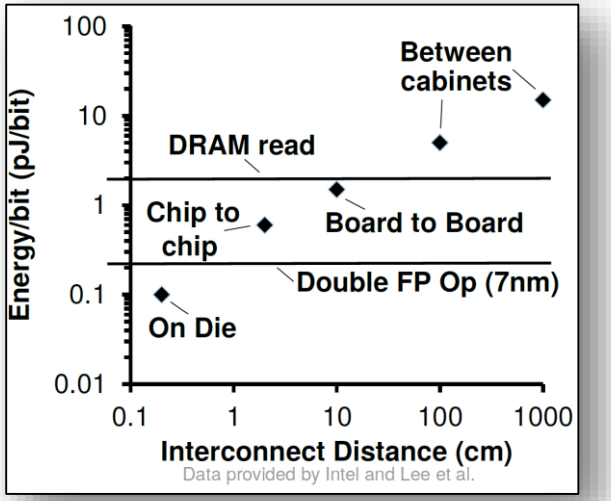




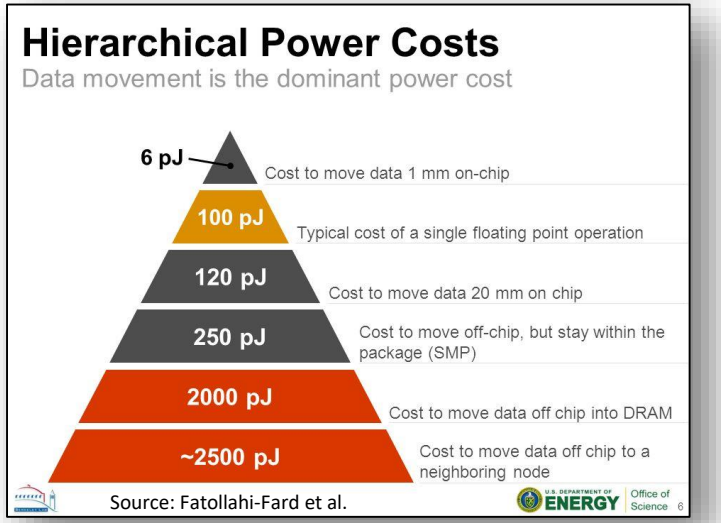
Data movement will dominate everything!

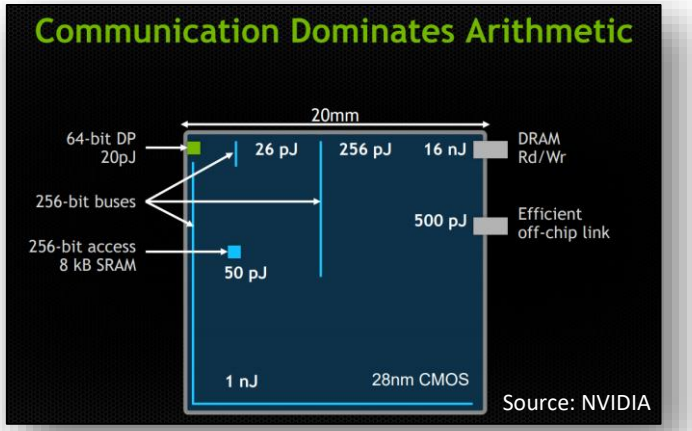


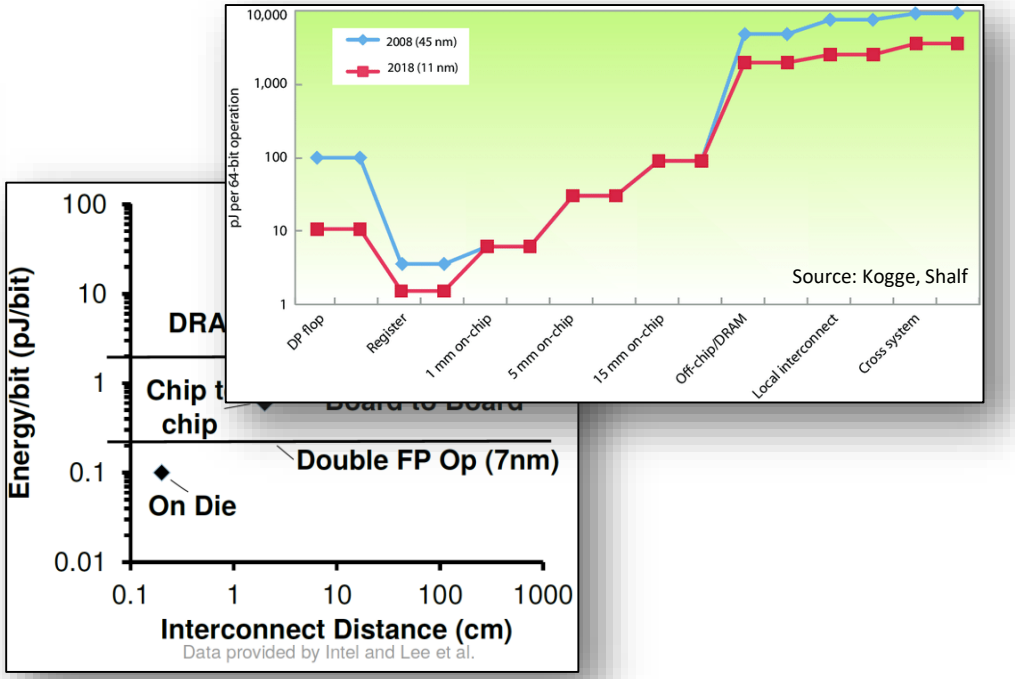




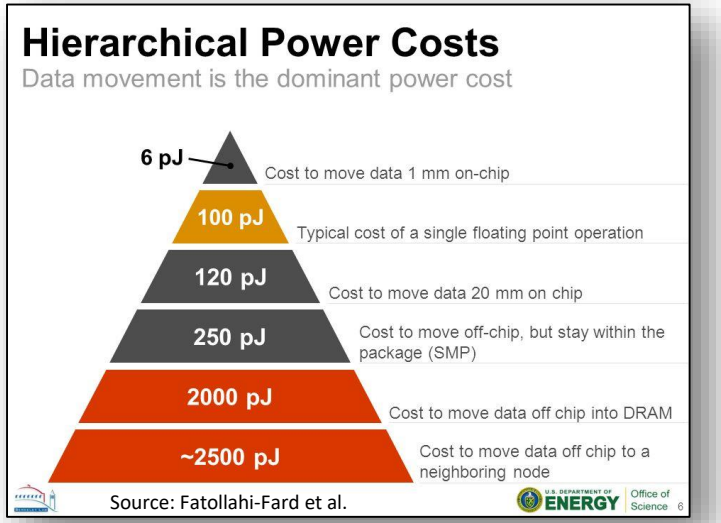
Data movement will dominate everything!

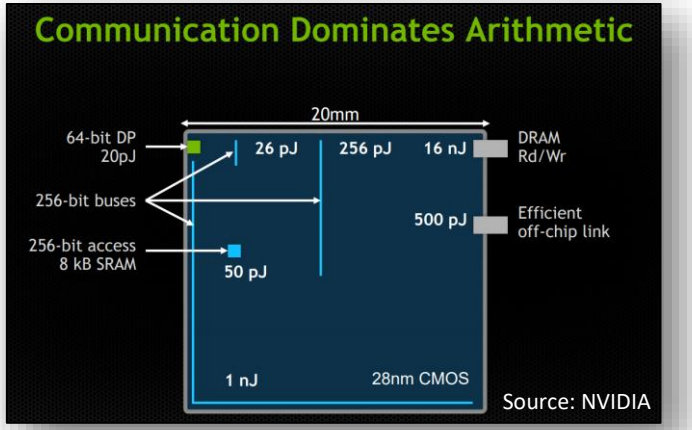


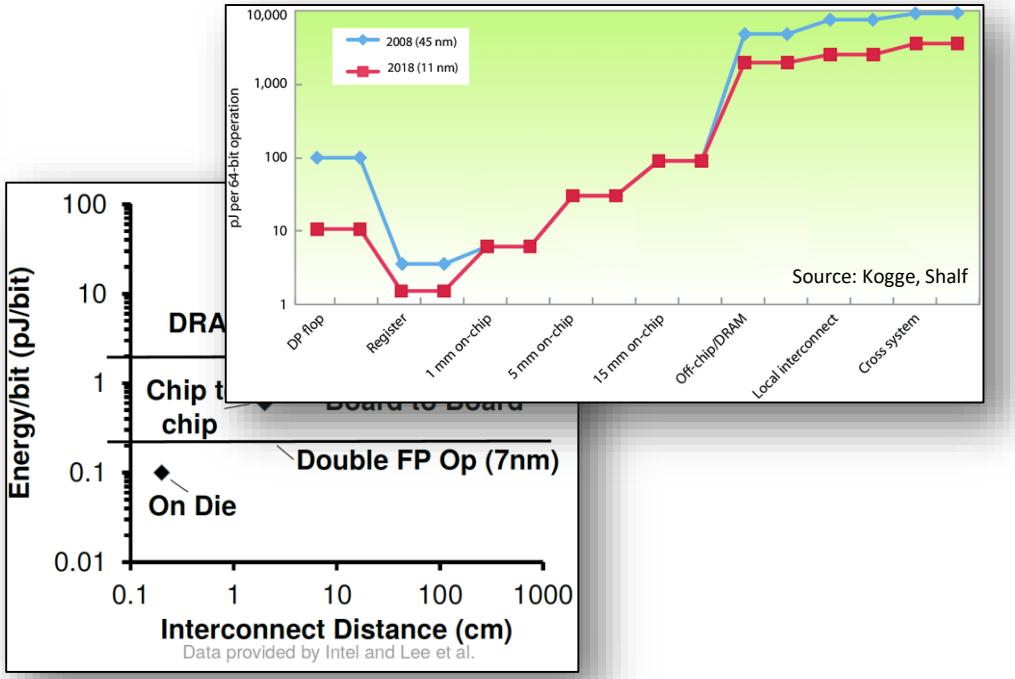




Data movement will dominate everything!







DOI:10.1145/1941487.1941507

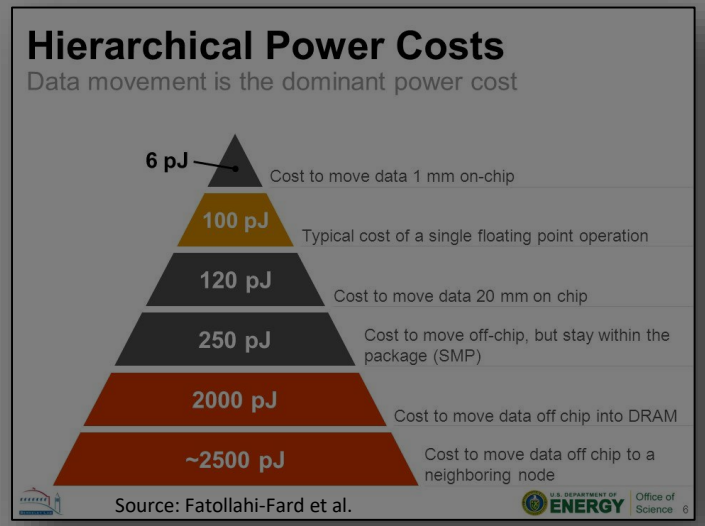
Energy efficiency is the new fundamental limiter of processor performance, way beyond numbers of processors.

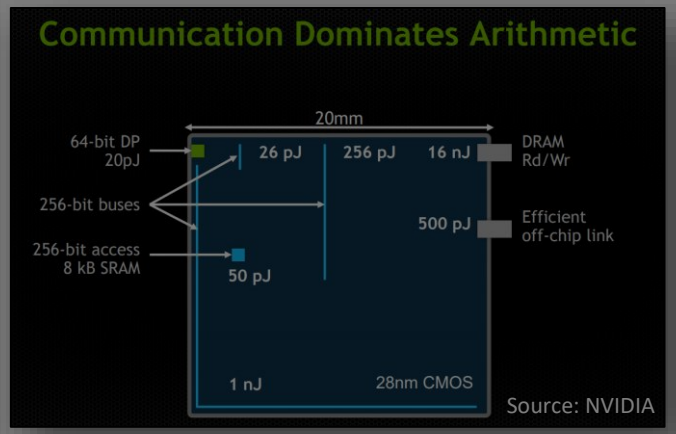
BY SHEKHAR BORKAR AND ANDREW A. CHIEN

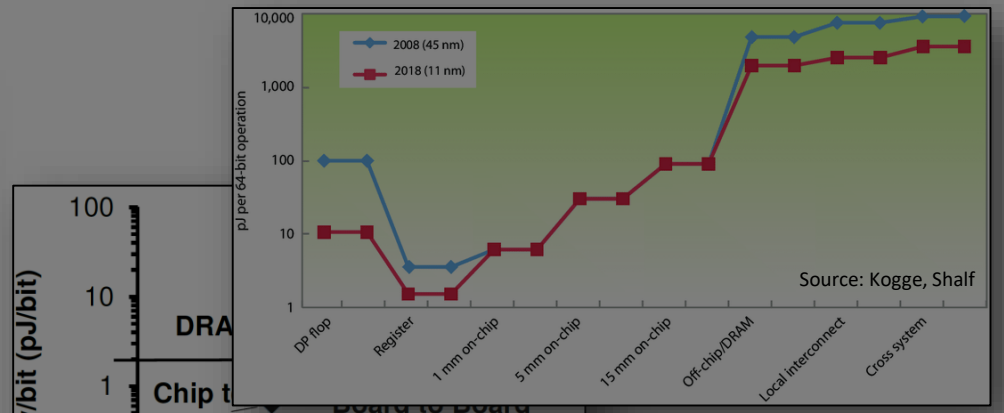
The Future of Microprocessors

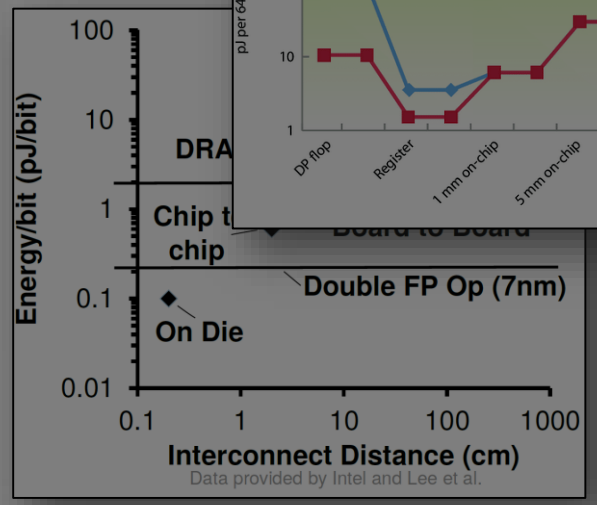
- "In future microprocessors, the energy expended for data movement will have a critical effect on achievable performance."
- "… movement consumes almost 58 watts with hardly any energy budget left for computation."
- "…the cost of data movement starts to dominate."
- "…data movement over these networks must be limited to conserve energy…"
- the phrase "data movement" appears 18 times on 11 pages (usually in concerning contexts)!
- "Efficient data orchestration will increasingly be critical, evolving to more efficient memory hierarchies and new types of interconnect tailored for locality and that **depend on sophisticated software to place computation and data so as to minimize data movement**."

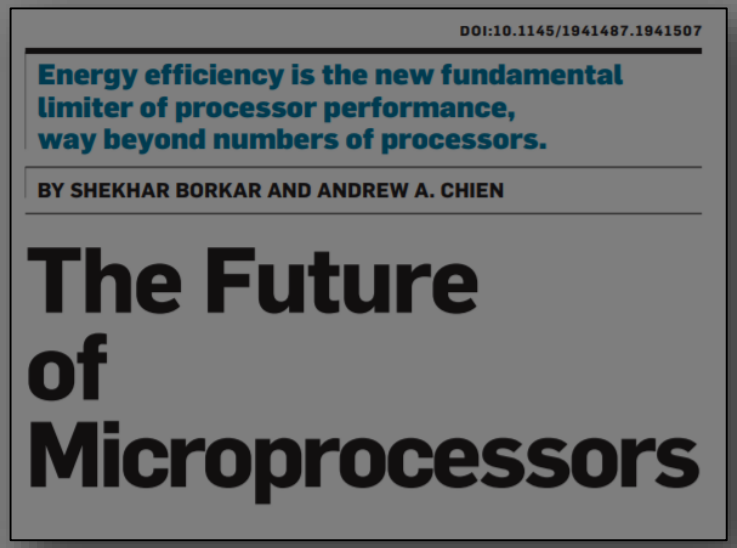
Data movement will dominate everything!










DOI:10.1145/1941487.1941507

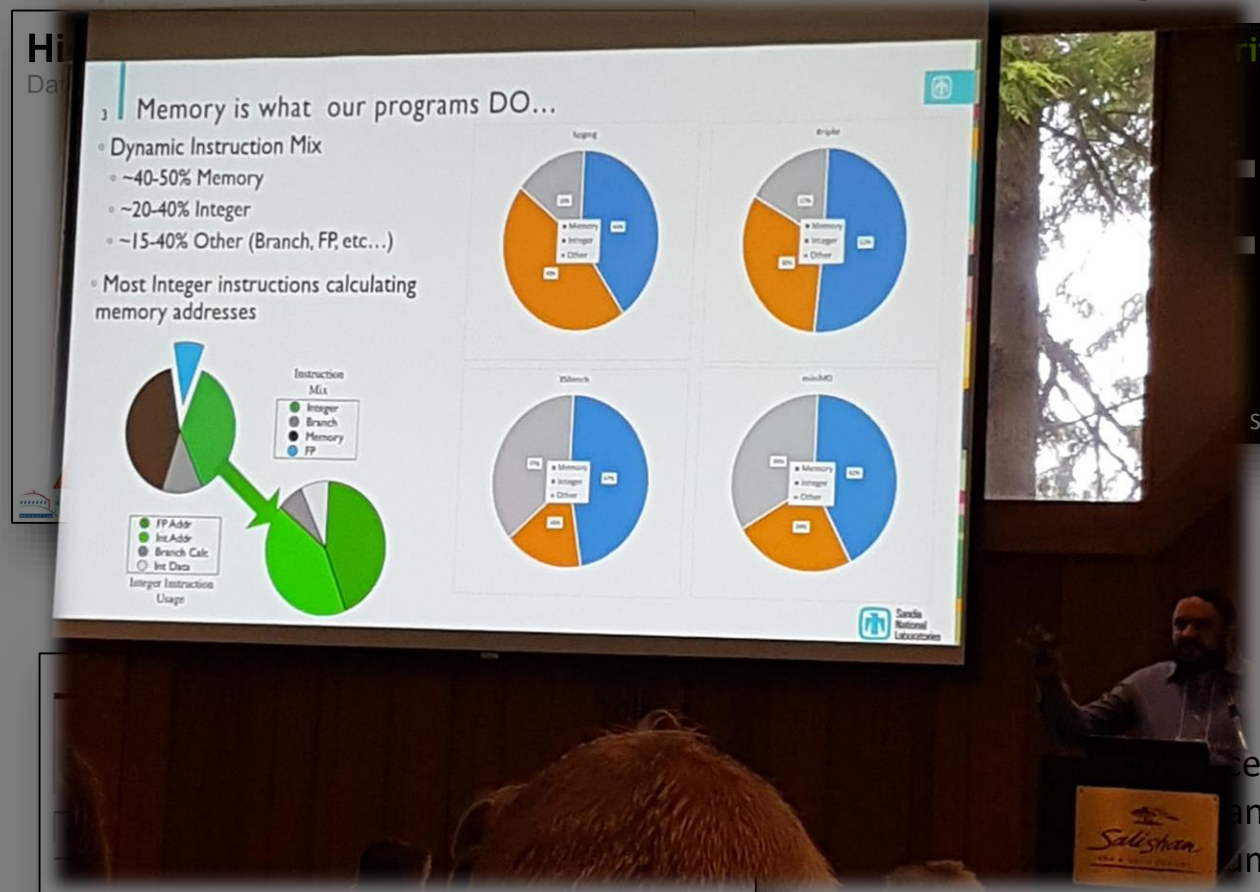
Energy efficiency is the new fundamental limiter of processor performance, way beyond numbers of processors.

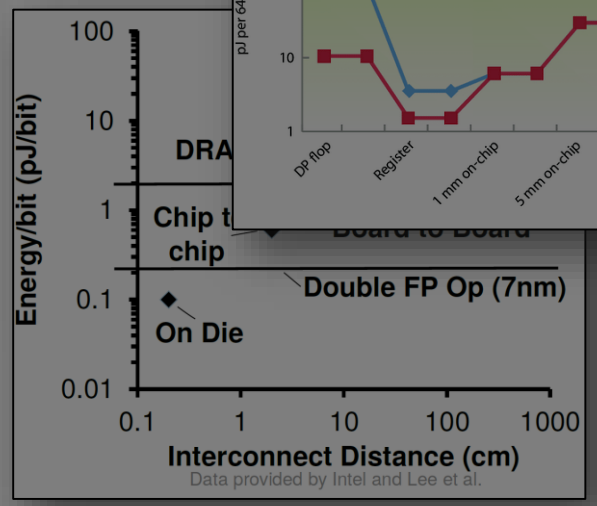
BY SHEKHAR BORKAR AND ANDREW A. CHIEN

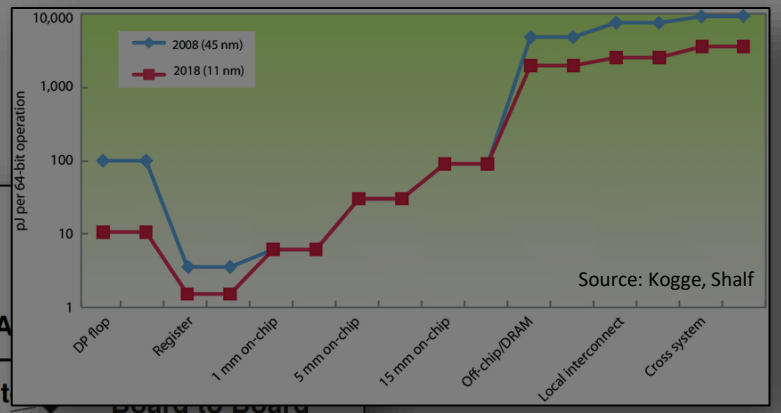
The Future of Microprocessors

- "In future microprocessors, the energy expended for data movement will have a critical effect on achievable performance."
- "… movement consumes almost 58 watts with hardly any energy budget left for computation."
- "…the cost of data movement starts to dominate."
- "…data movement over these networks must be limited to conserve energy…"
- the phrase "data movement" appears 18 times on 11 pages (usually in concerning contexts)!
- "Efficient data orchestration will increasingly be critical, evolving to more efficient memory hierarchies and new types of interconnect tailored for locality and that **depend on sophisticated software to place computation and data so as to minimize data movement**."

Data movement will dominate everything!





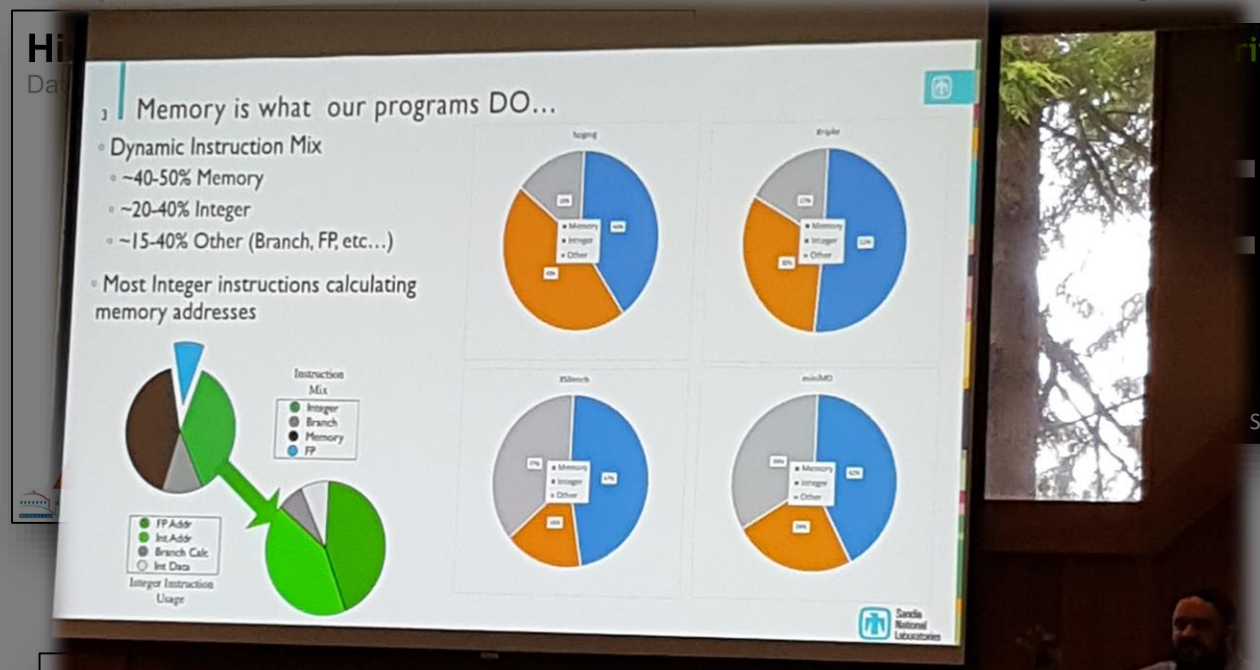


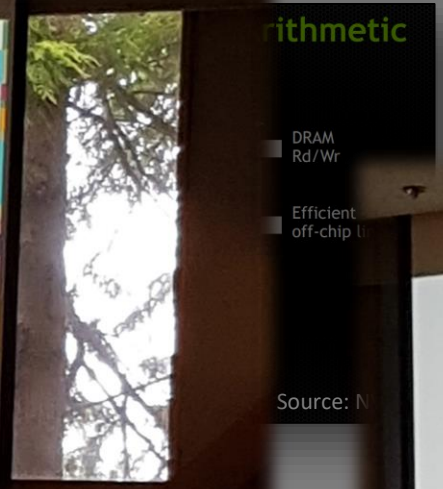
The Future of Microprocessors

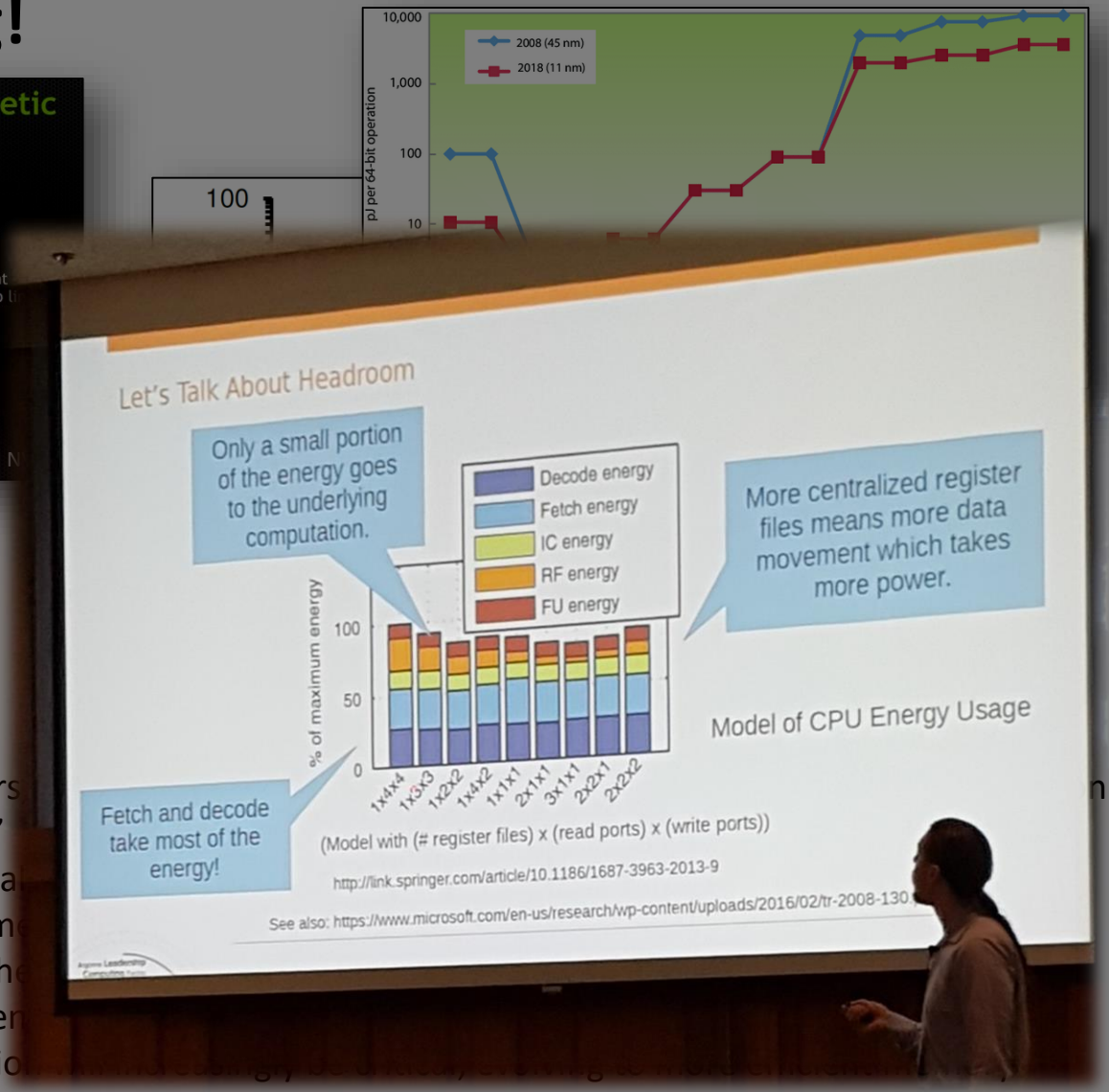
essors, the energy expended for data movement will have a critical effect on
nce."
mes almost 58 watts with hardly any energy budget left for computation."

- "…the cost of data movement starts to dominate."
- "…data movement over these networks must be limited to conserve energy…"
- the phrase "data movement" appears 18 times on 11 pages (usually in concerning contexts)!
- "Efficient data orchestration will increasingly be critical, evolving to more efficient memory hierarchies and new types of interconnect tailored for locality and that **depend on sophisticated software to place computation and data so as to minimize data movement**."

Data movement will dominate everything!







- "...the cost of data movem
- "...data movement over th
- the phrase "data movemen
- "Efficient data orchestratio hierarchies and new types of interconnect tailored for locality and that **depend on sophisticated software to place computation and data so as to minimize data movement**."

“Sophisticated software”: How do we program today?

“Sophisticated software”: How do we program today?

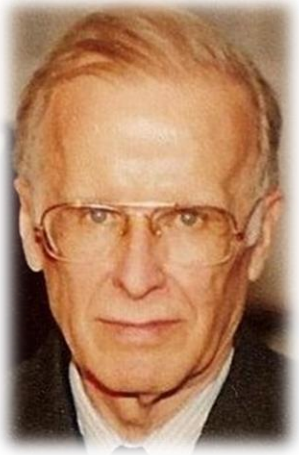
- **Well, to a good approximation how we programmed yesterday**

“Sophisticated software”: How do we program today?

- **Well, to a good approximation how we programmed yesterday**
  - Or last year?

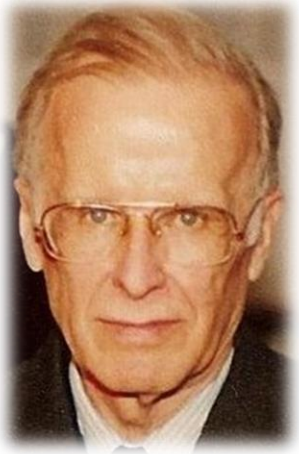
“Sophisticated software”: How do we program today?

- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?



Backus ‘77: *“The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer’s bottleneck does.”*

“Sophisticated software”: How do we program today?

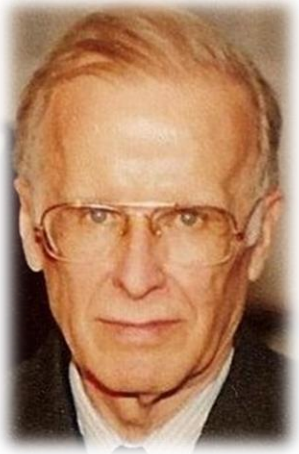


- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**

Backus ‘77: *“The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer’s bottleneck does.”*

“Sophisticated software”: How do we program today?

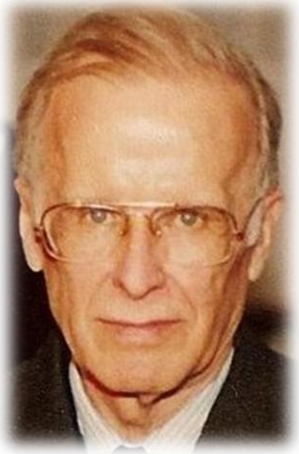


- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**
  - Worry about operation counts (flop/s is **the metric**, isn’t it?)

Backus ‘77: *“The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer’s bottleneck does.”*

“Sophisticated software”: How do we program today?

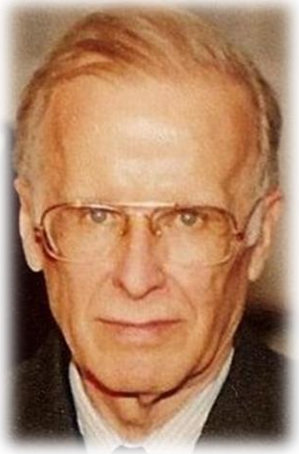


- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**
  - Worry about operation counts (flop/s is **the metric**, isn’t it?)
  - Data movement is at best implicit (or invisible/ignored)

Backus ‘77: *“The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer’s bottleneck does.”*

"Sophisticated software": How do we program today?



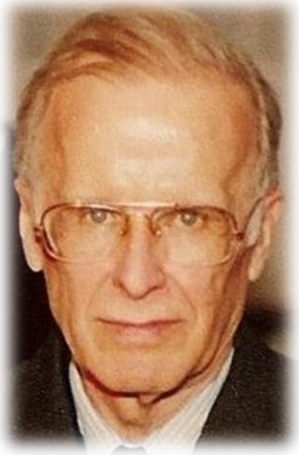
- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**
  - Worry about operation counts (flop/s is **the metric**, isn't it?)
  - Data movement is at best implicit (or invisible/ignored)

- **Legion is taking a good direction towards data-centric**

Backus '77: *"The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer's bottleneck does."*

"Sophisticated software": How do we program today?



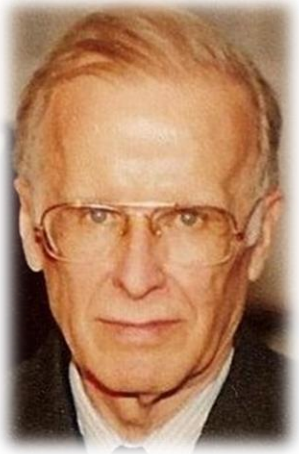
- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**
  - Worry about operation counts (flop/s is **the metric**, isn't it?)
  - Data movement is at best implicit (or invisible/ignored)

Backus '77: *"The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer's bottleneck does."*

- **Legion is taking a good direction towards data-centric**
  - Tasking relies on data placement but not really dependencies (not visible to tool-chain)

"Sophisticated software": How do we program today?



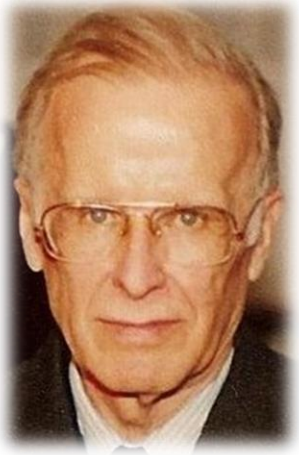
- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**
  - Worry about operation counts (flop/s is **the metric**, isn't it?)
  - Data movement is at best implicit (or invisible/ignored)

Backus '77: *"The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer's bottleneck does."*

- **Legion is taking a good direction towards data-centric**
  - Tasking relies on data placement but not really dependencies (not visible to tool-chain)
  - But it is still control-centric in the tasks – not (performance) portable between devices!

"Sophisticated software": How do we program today?



- **Well, to a good approximation how we programmed yesterday**
  - Or last year?
  - Or four decades ago?

- **Control-centric programming**
  - Worry about operation counts (flop/s is **the metric**, isn't it?)
  - Data movement is at best implicit (or invisible/ignored)

Backus '77: *"The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer's bottleneck does."*

- **Legion is taking a good direction towards data-centric**
  - Tasking relies on data placement but not really dependencies (not visible to tool-chain)
  - But it is still control-centric in the tasks – not (performance) portable between devices!

- **Let's go a step further towards an explicitly data-centric viewpoint**
  - For performance engineers at least!

Performance Portability with DataCentric (DaCe) Parallel Programming

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance Portability with DataCentric (DaCe) Parallel Programming

Domain Scientist

Problem Formulation

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

```
@dace.program
def program(A, B):

 @dace.map
 def heat(z: _[0:D],
          y: _[0:H],
          x: _[0:W]):
  a << A[z,y,x]
  b >> B[z,y,x]
  ...
```

numpy / MATLAB

TensorFlow

Explicit

Implicit

High-Level Program

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance Portability with DataCentric (DaCe) Parallel Programming

Domain Scientist

Problem Formulation

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

```
@dace.program
def program(A, B):

 @dace.map
 def heat(z: _[0:D],
          y: _[0:H],
          x: _[0:W]):
  a << A[z,y,x]
  b >> B[z,y,x]
  ...
```

numpy / MATLAB

TensorFlow

Explicit

Implicit

High-Level Program

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance Portability with DataCentric (DaCe) Parallel Programming

Domain Scientist

Problem Formulation

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

```
@dace.program
def program(A, B):

 @dace.map
 def heat(z: _[0:D],
          y: _[0:H],
          x: _[0:W]):
  a << A[z,y,x]
  b >> B[z,y,x]
  ...
```

numpy / MATLAB

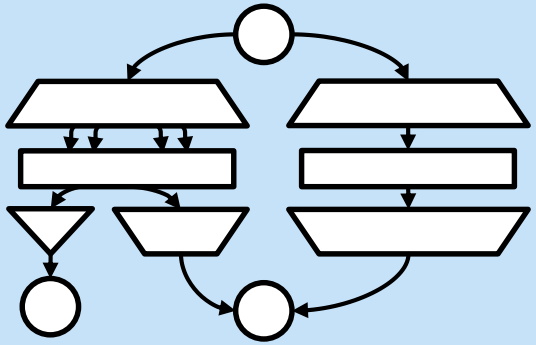
TensorFlow

Explicit

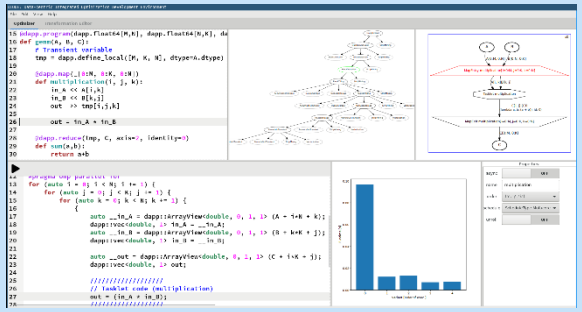
Implicit

High-Level Program

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Interactive Optimization Environment (DIODE)

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance Portability with DataCentric (DaCe) Parallel Programming

Domain Scientist

Problem Formulation

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

```
@dace.program
def program(A, B):

 @dace.map
 def heat(z: _[0:D],
          y: _[0:H],
          x: _[0:W]):
  a << A[z,y,x]
  b >> B[z,y,x]
  ...
```

numpy / MATLAB

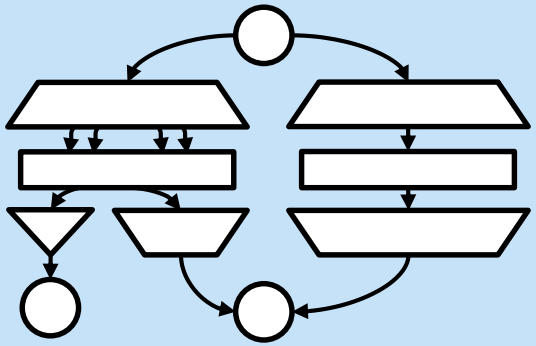
TensorFlow

Explicit

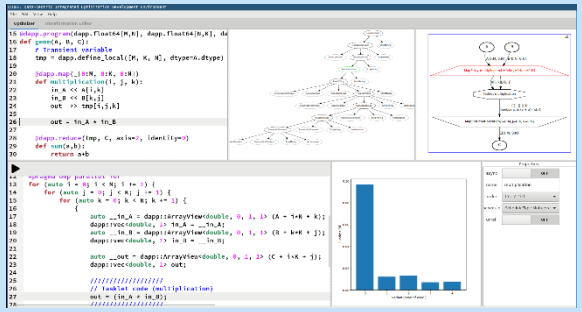
Implicit

High-Level Program

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Interactive Optimization Environment (DIODE)

Graph Transformation

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance Portability with DataCentric (DaCe) Parallel Programming

Domain Scientist

Problem Formulation

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

```
@dace.program
def program(A, B):

 @dace.map
 def heat(z: _[0:D],
          y: _[0:H],
          x: _[0:W]):
  a << A[z,y,x]
  b >> B[z,y,x]
  ...
```

numpy / MATLAB

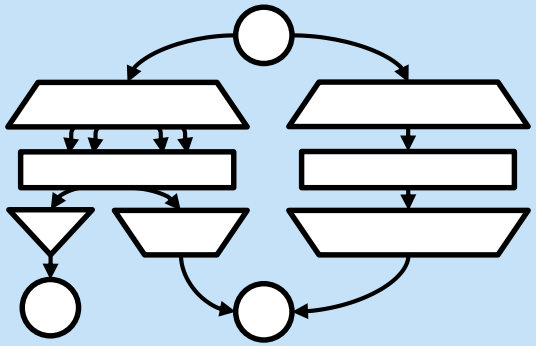
TensorFlow

Explicit

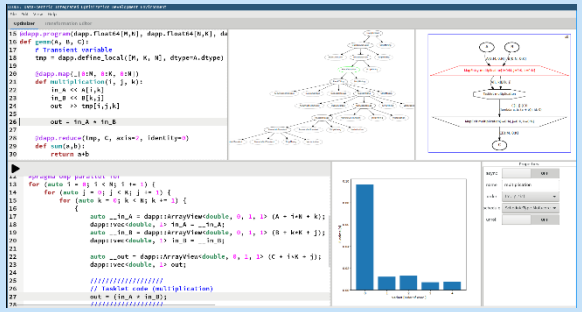
Implicit

High-Level Program

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Interactive Optimization Environment (DIODE)

Graph Transformation

System

Hardware Information

SDFG Compiler

Runtime

CPU Binary

GPU Binary

FPGA Modules

Thin Application Infrastructure

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance Portability with DataCentric (DaCe) Parallel Programming

Domain Scientist

Problem Formulation

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

```
@dace.program
def program(A, B):

 @dace.map
 def heat(z: _[0:D],
          y: _[0:H],
          x: _[0:W]):
  a << A[z,y,x]
  b >> B[z,y,x]
  ...
```

numpy / MATLAB

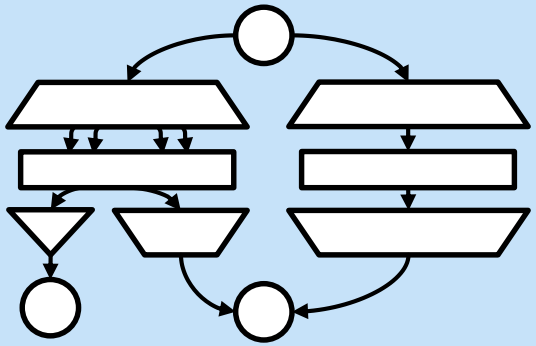
TensorFlow

Explicit

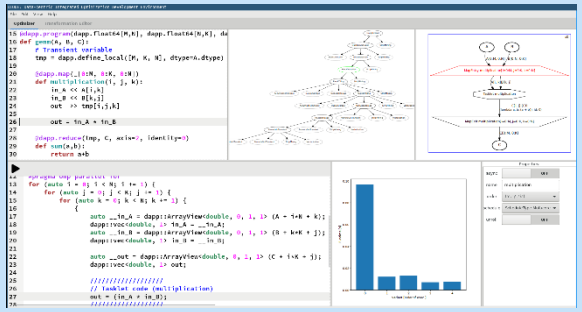
Implicit

High-Level Program

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Interactive Optimization Environment (DIODE)

Graph Transformation

Performance Results

System

Hardware Information

SDFG Compiler

Runtime

CPU Binary

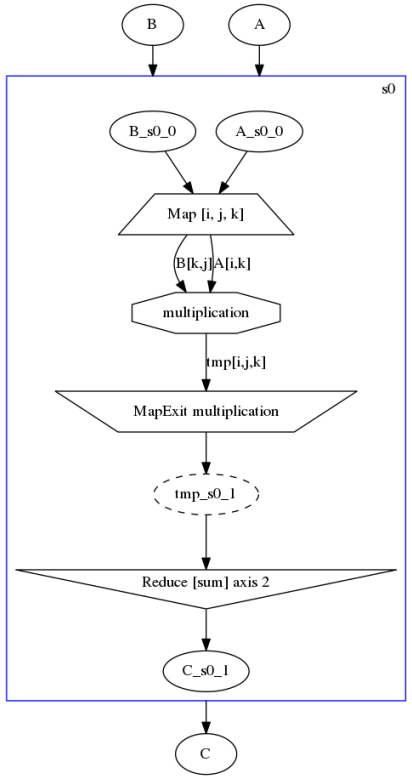
GPU Binary

FPGA Modules

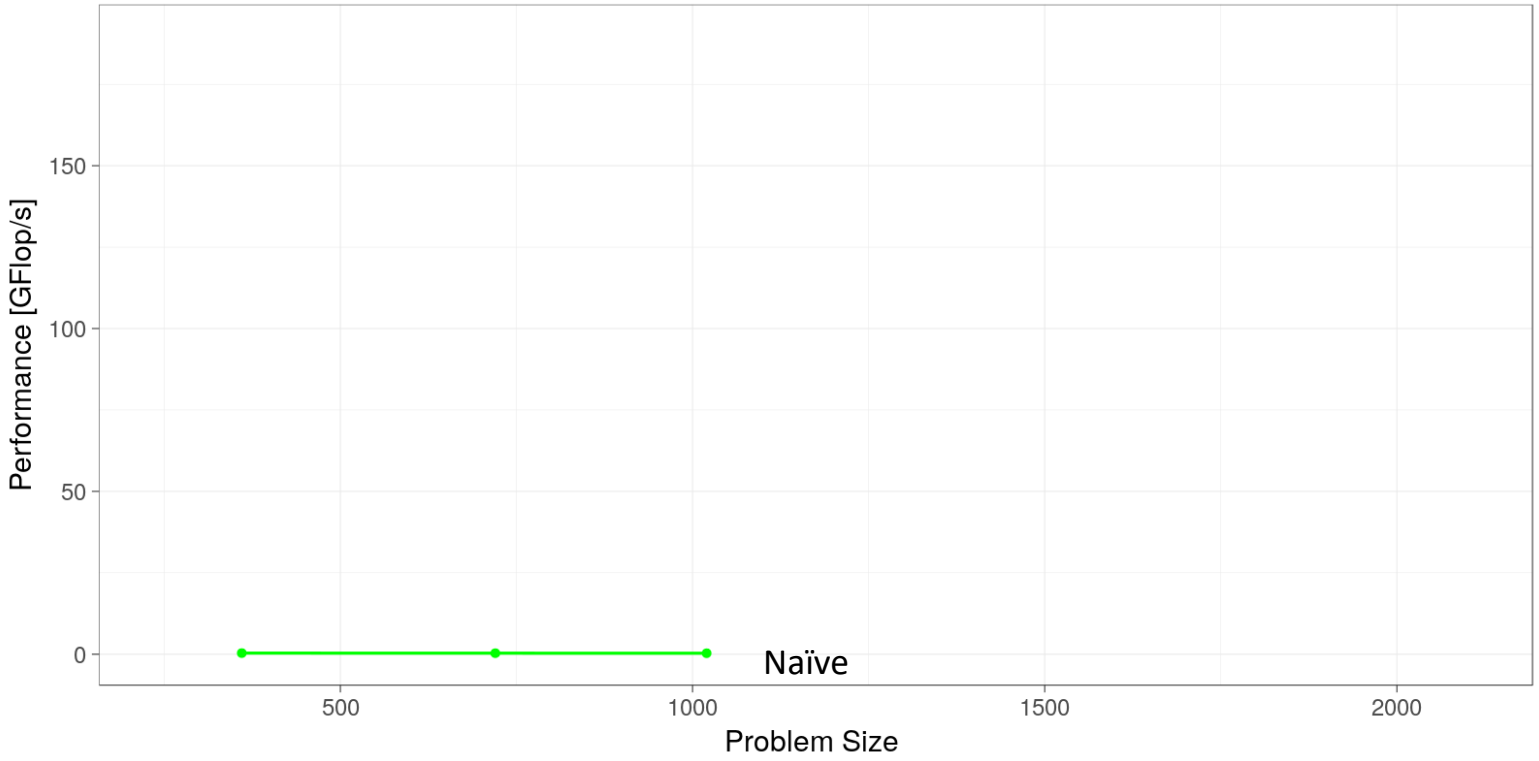
Thin Application Infrastructure

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86

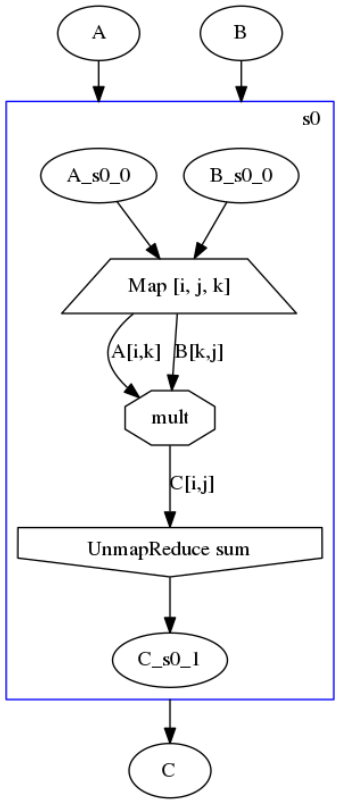


SDFG

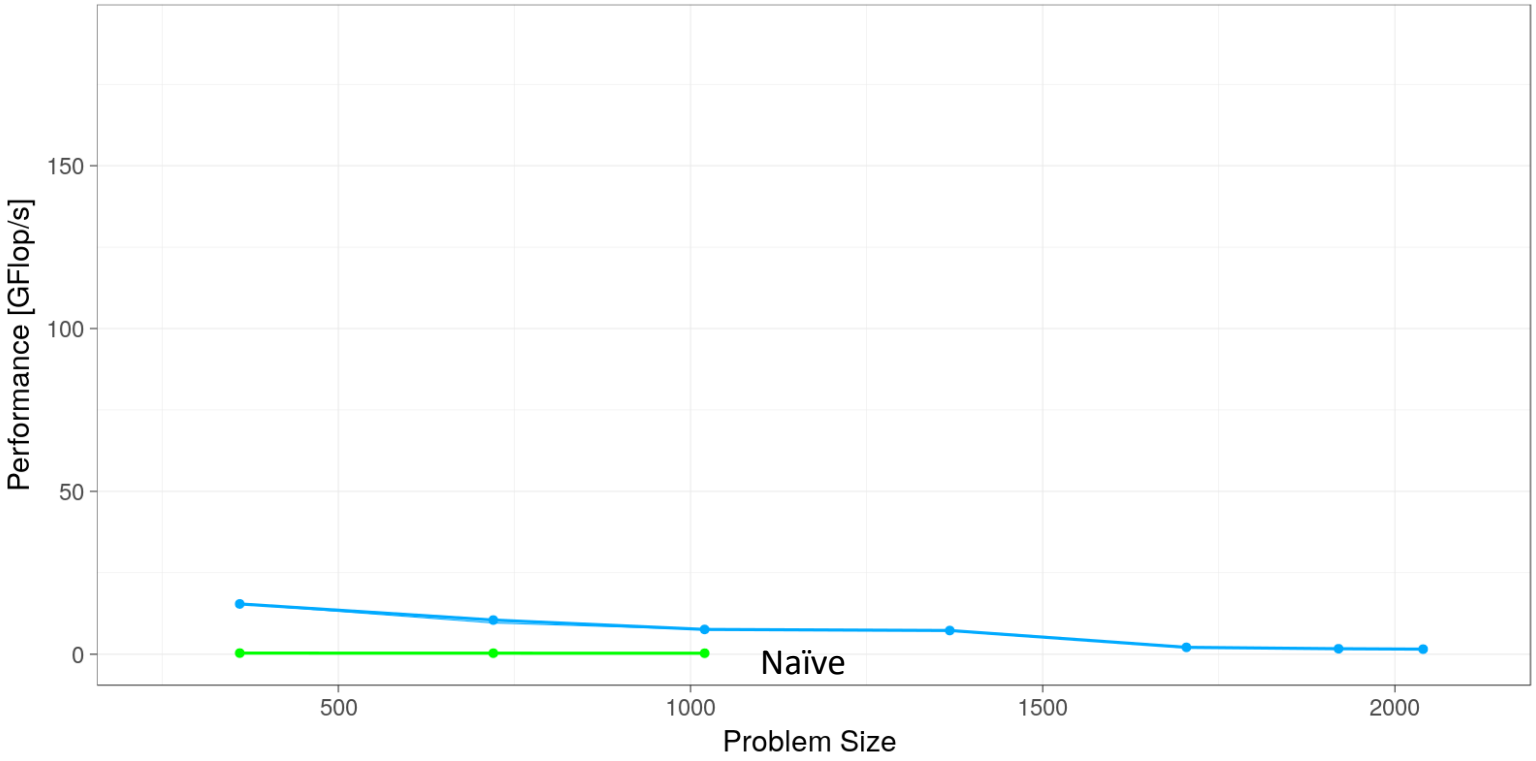


Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



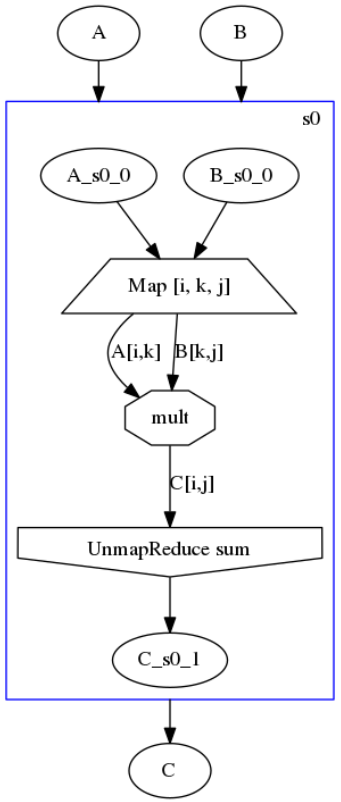
SDFG



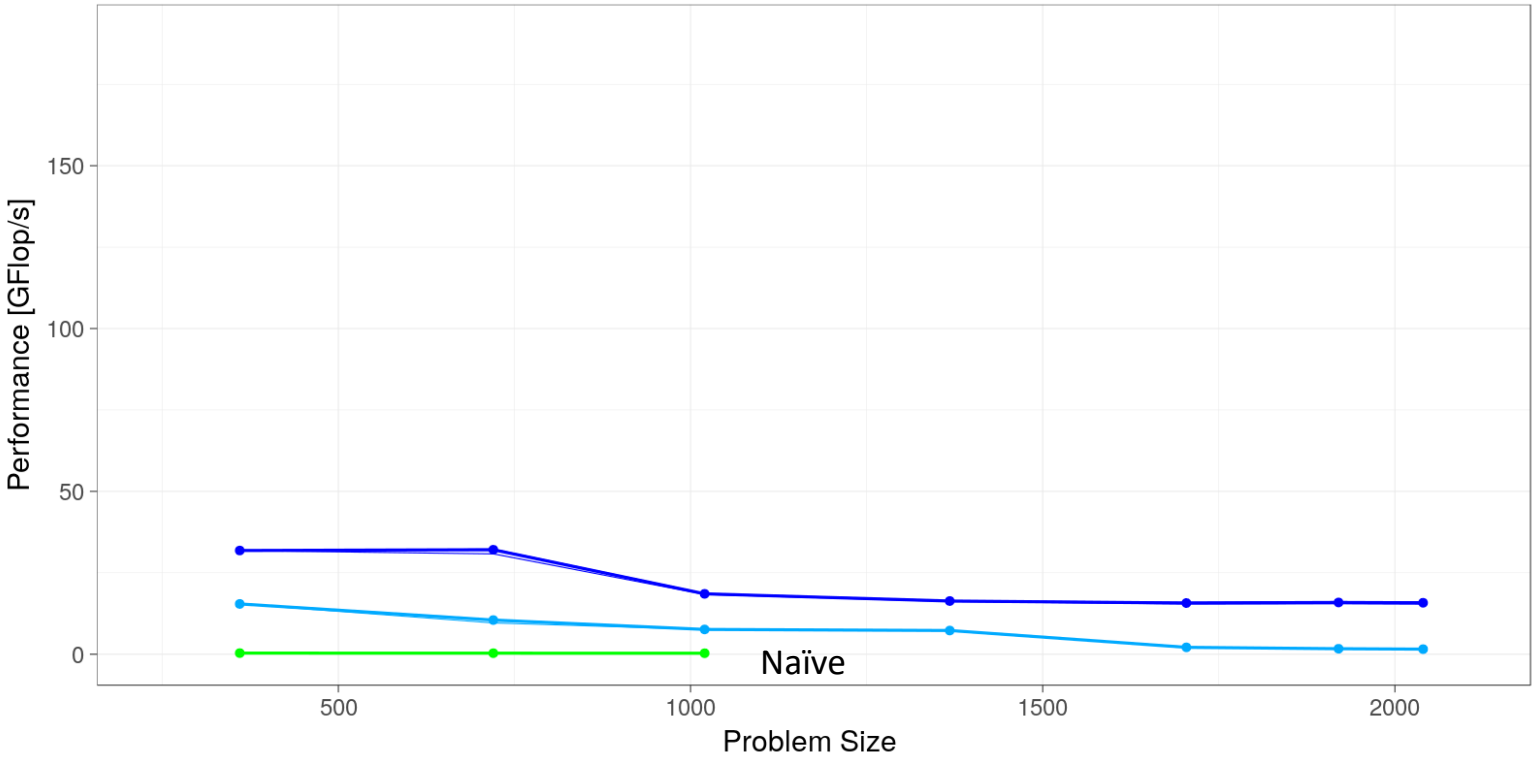
MapReduceFusion

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



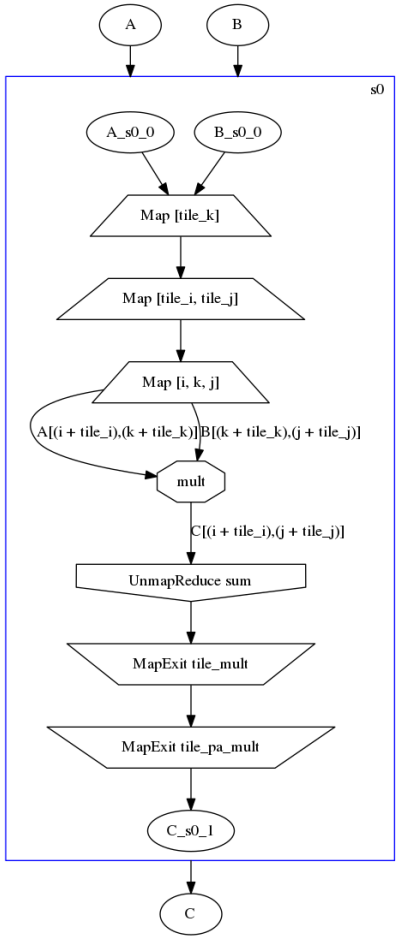
SDFG



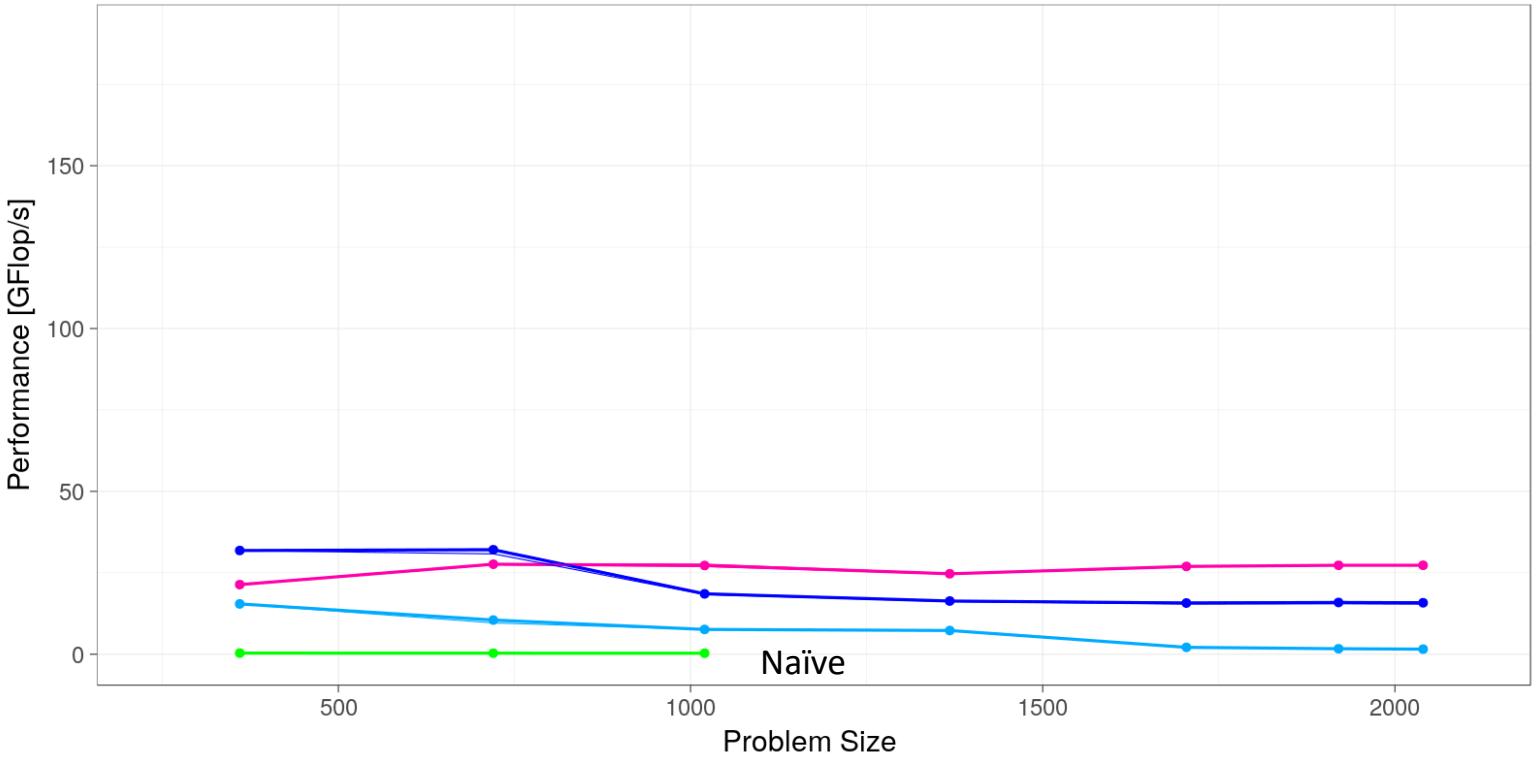
LoopReorder
MapReduceFusion

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



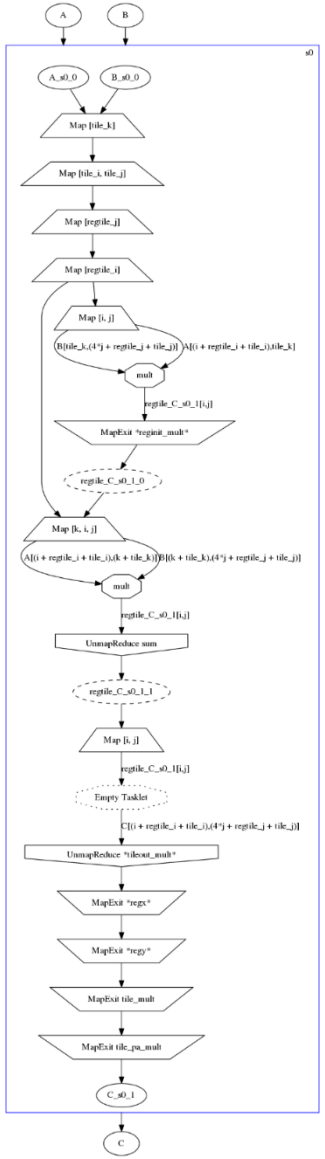
SDFG

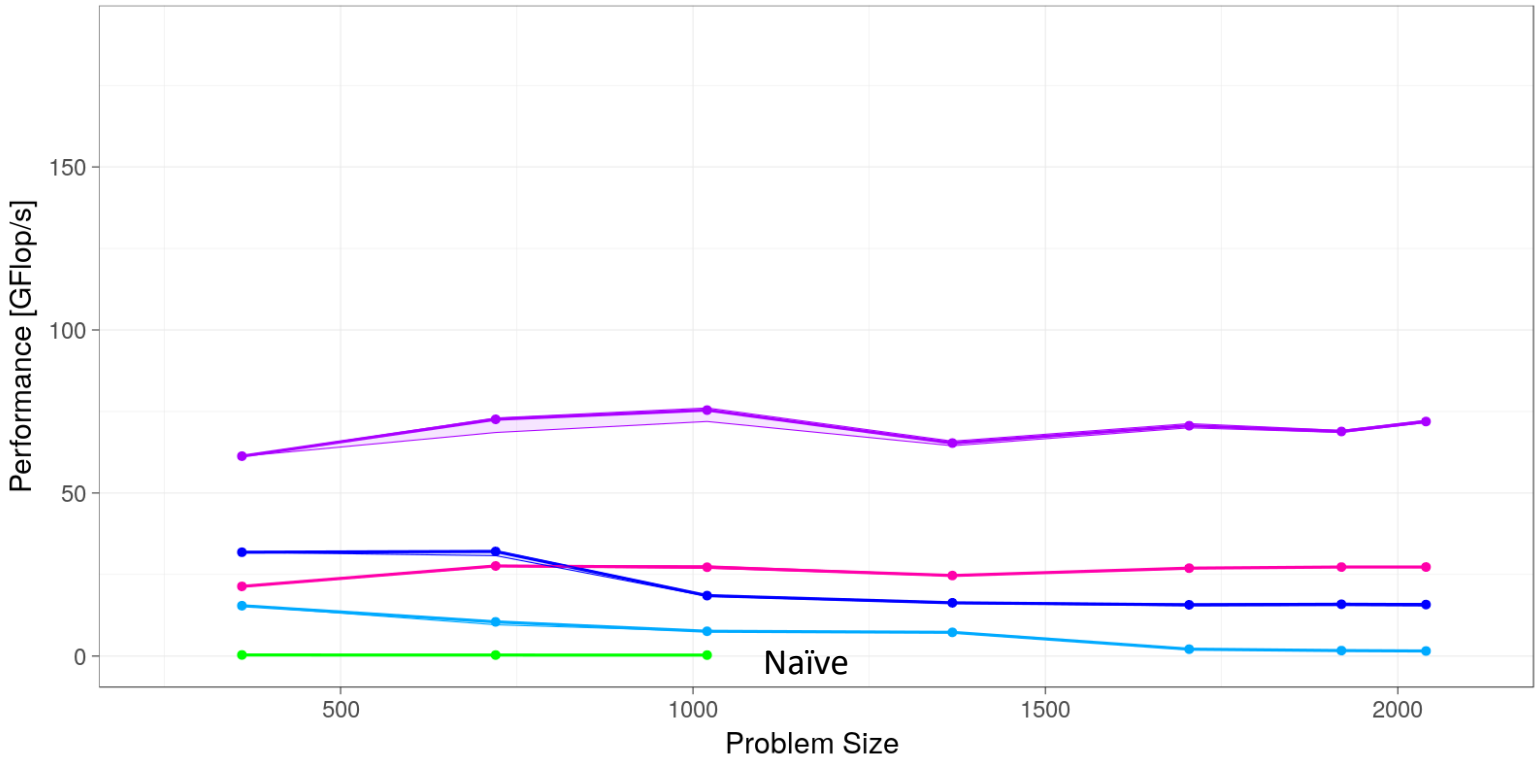


BlockTiling
LoopReorder
MapReduceFusion

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



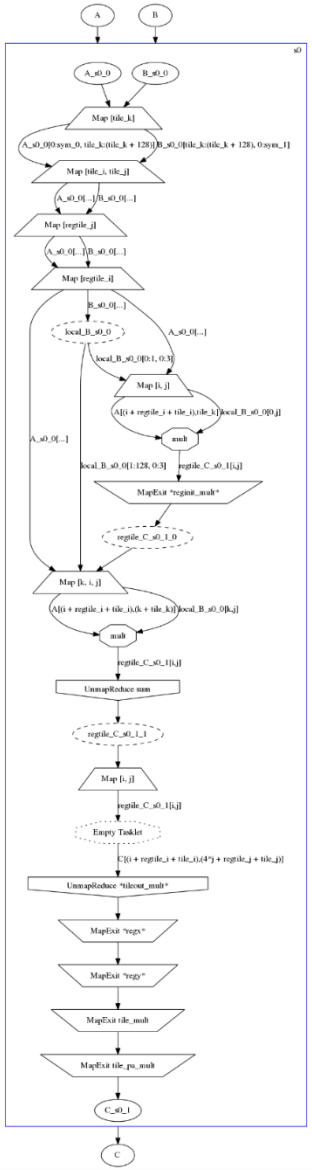


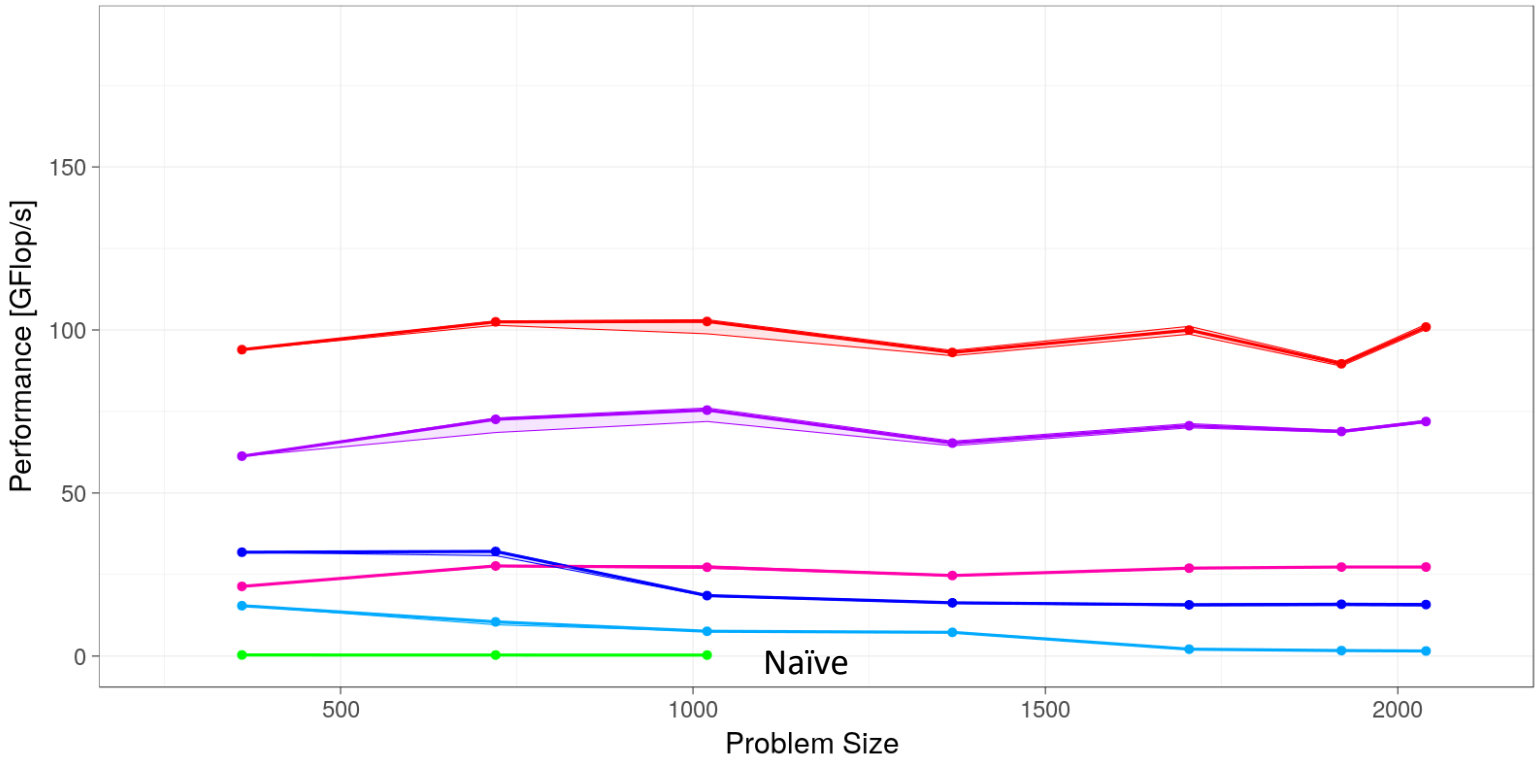
RegisterTiling

BlockTiling
LoopReorder
MapReduceFusion

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86





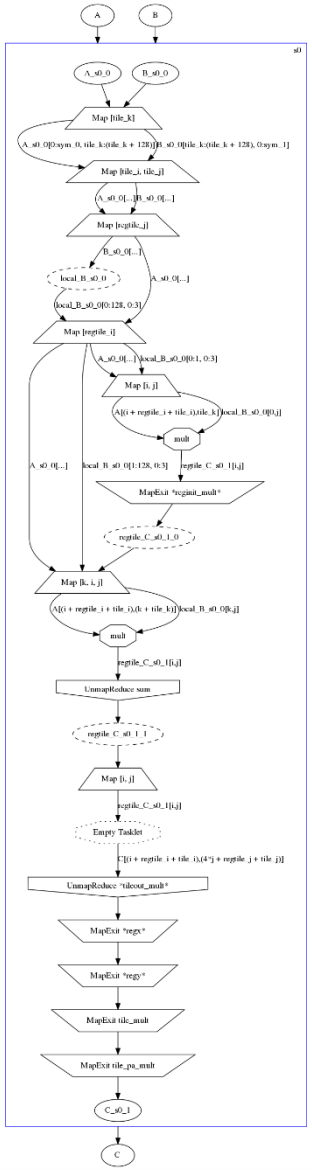
LocalStorage

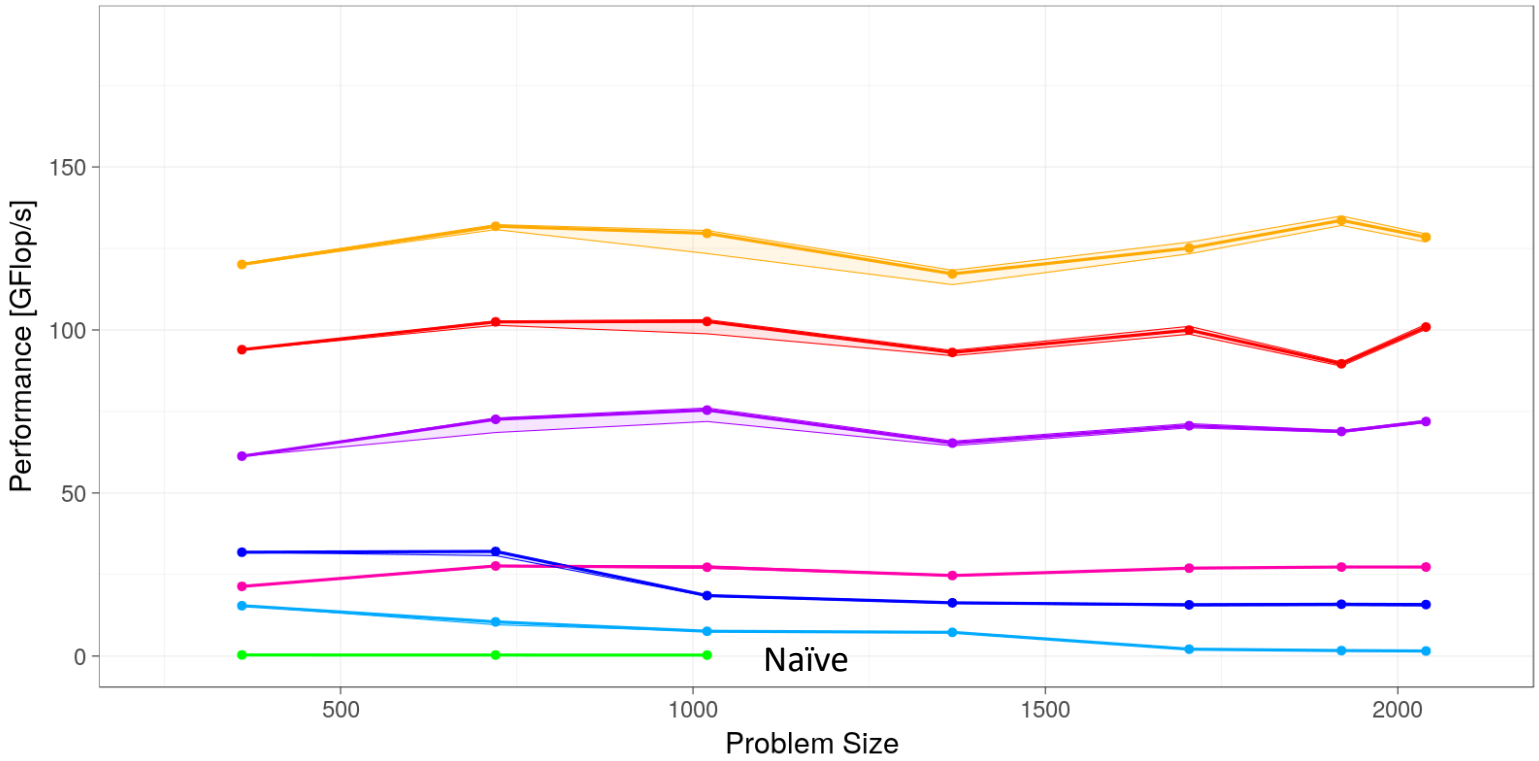
RegisterTiling

BlockTiling
LoopReorder
MapReduceFusion

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86





PromoteTransient

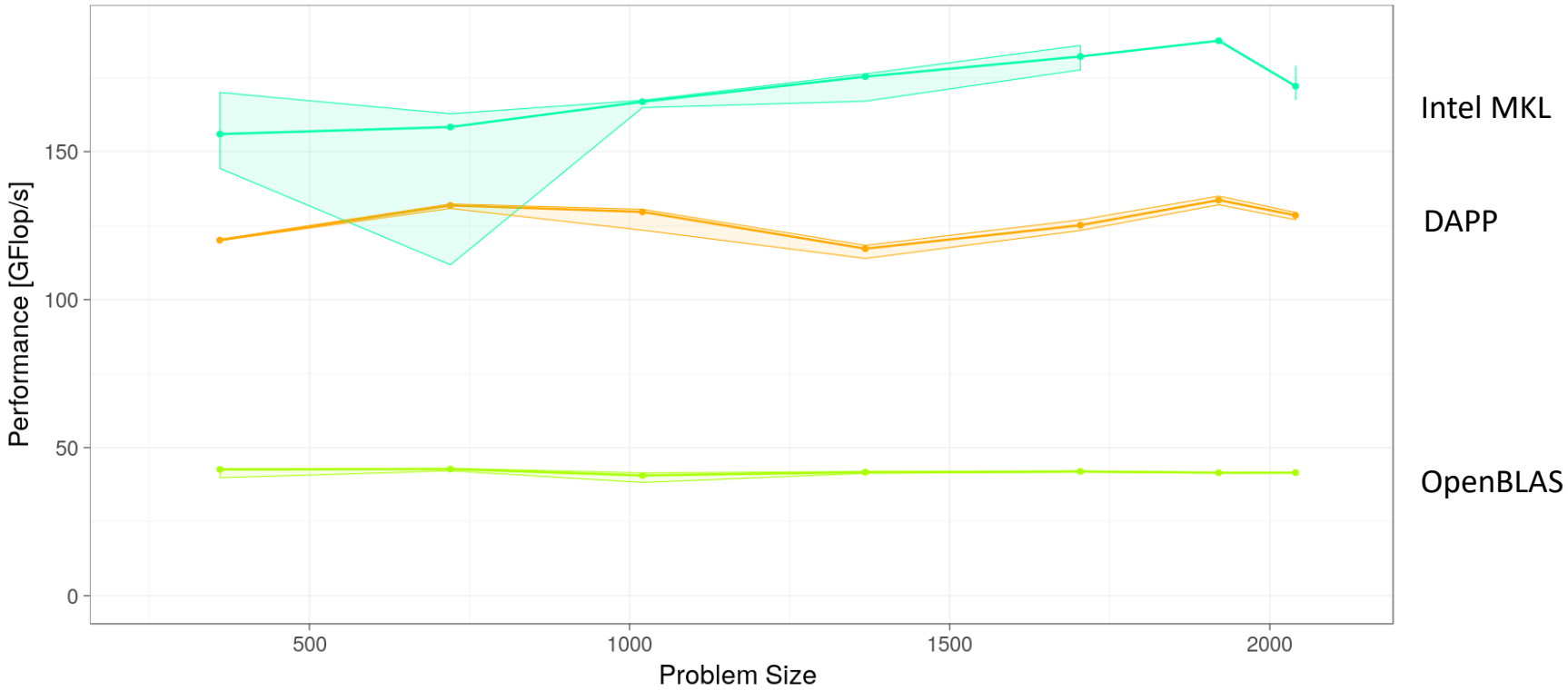
LocalStorage

RegisterTiling

BlockTiling
LoopReorder
MapReduceFusion

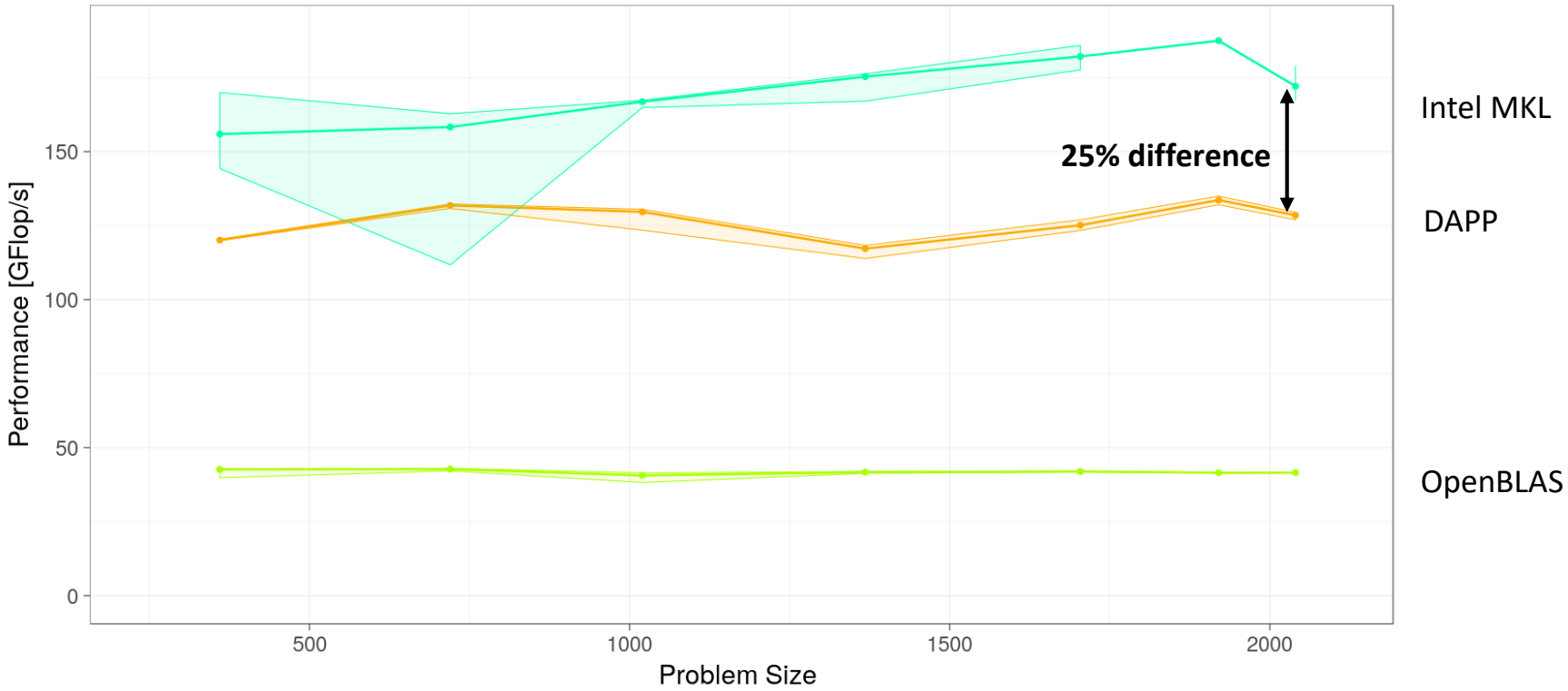
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



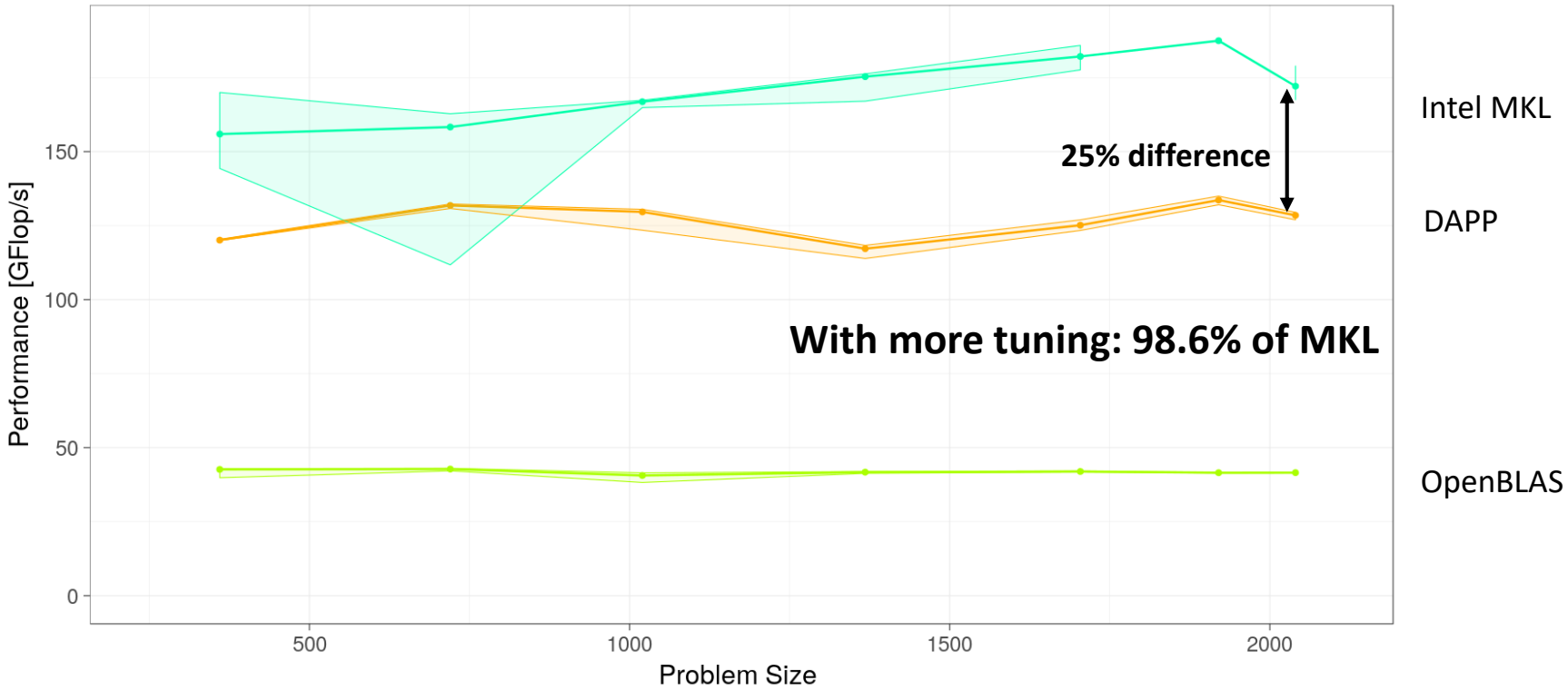
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



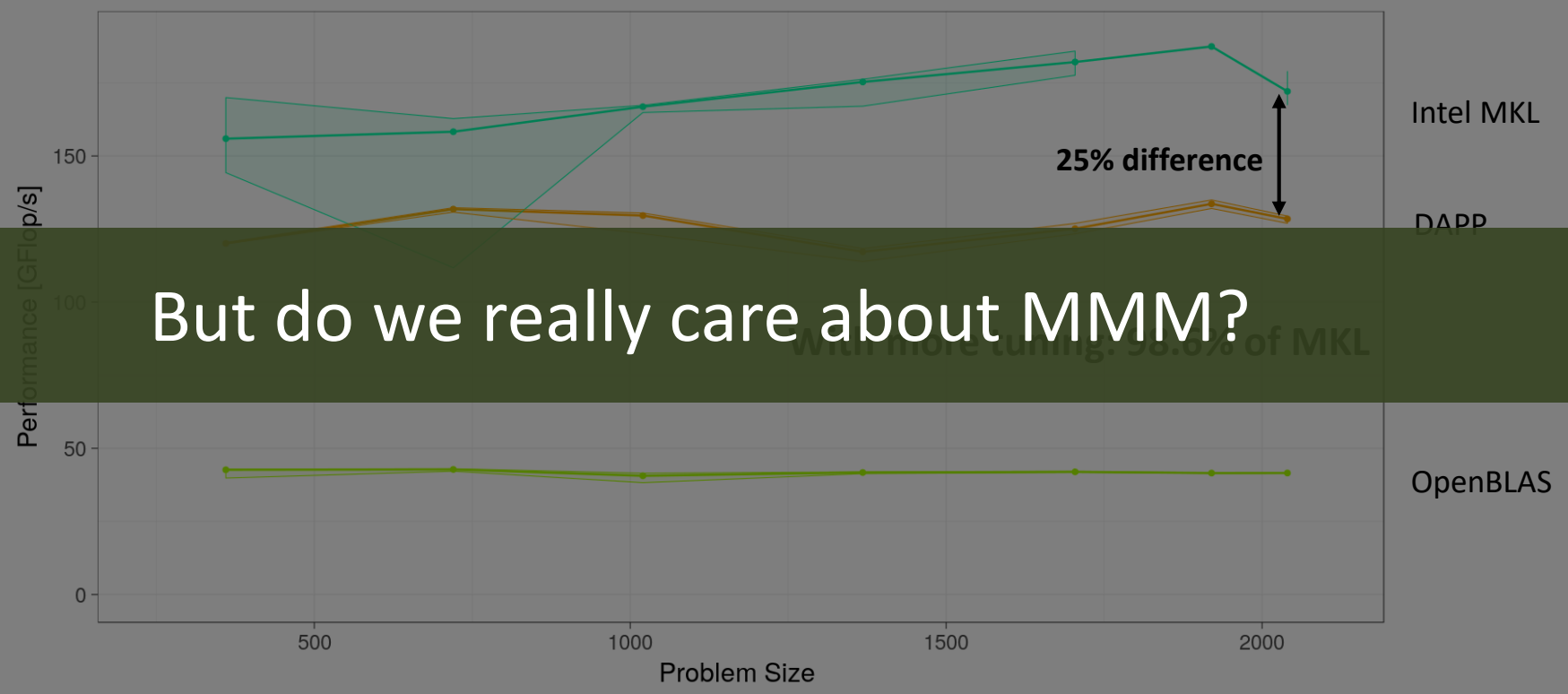
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Performance for matrix multiplication on x86



Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

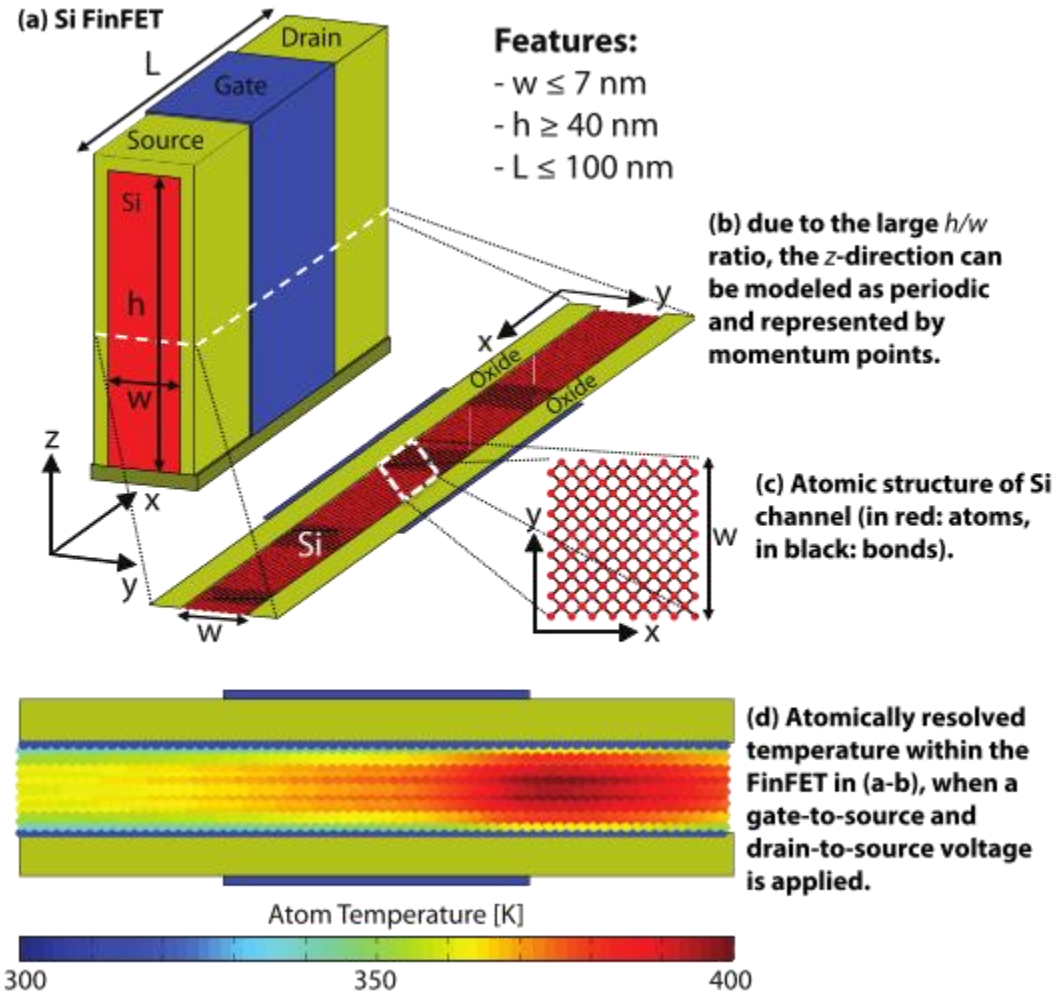
Performance for matrix multiplication on x86



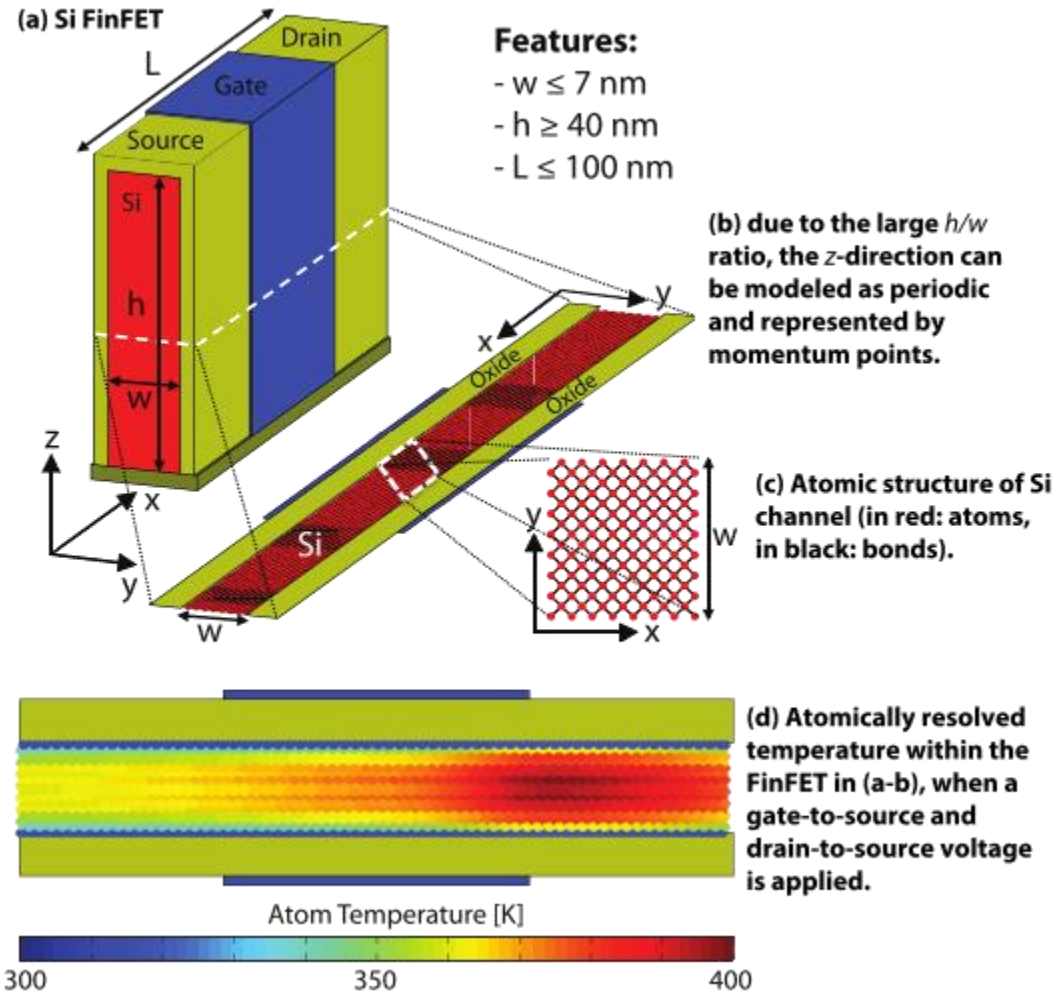
But do we really care about MMM?

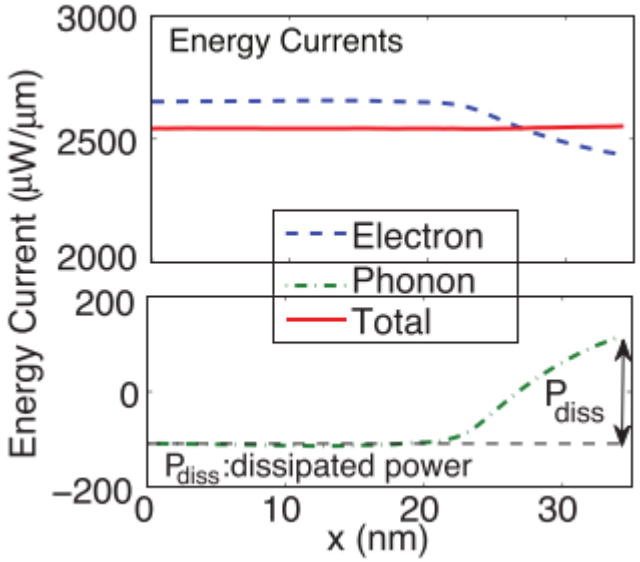
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Next-Generation Transistors need to be cooler – addressing self-heating

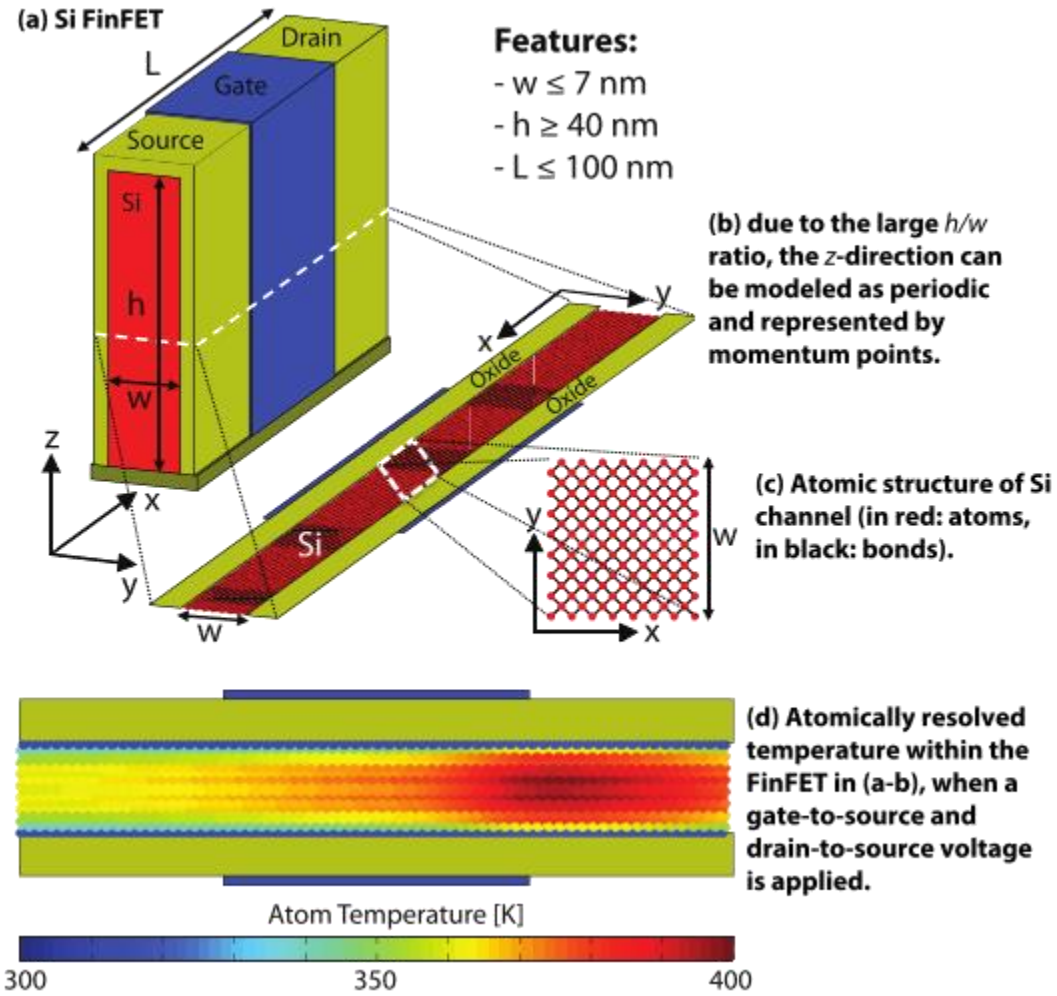


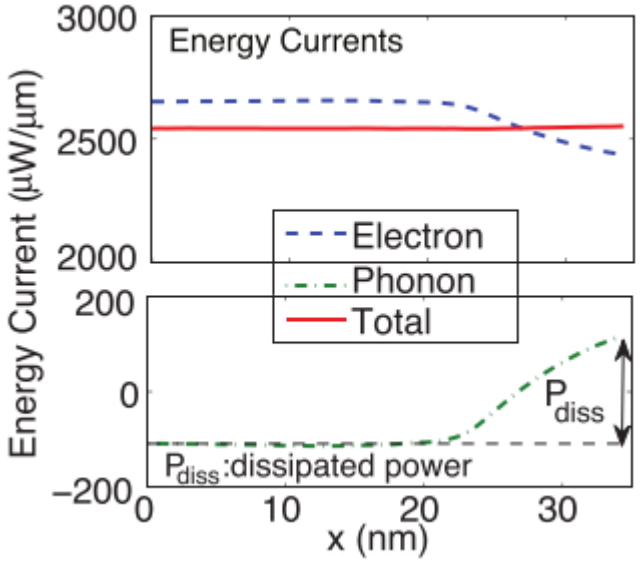
Next-Generation Transistors need to be cooler – addressing self-heating

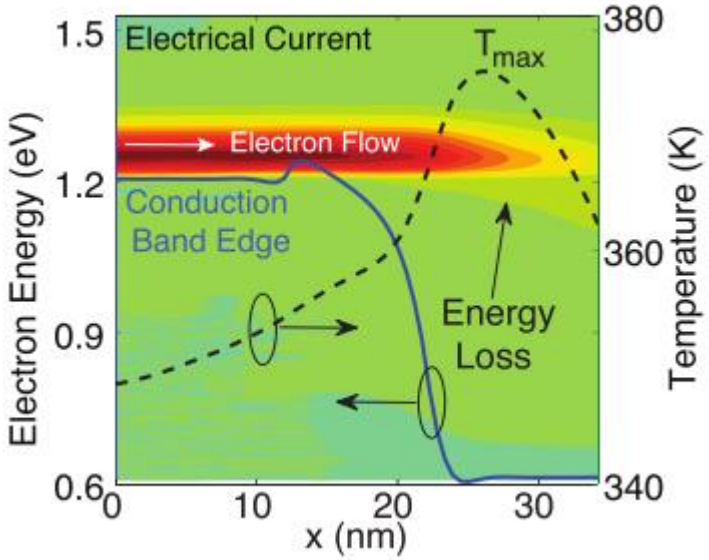




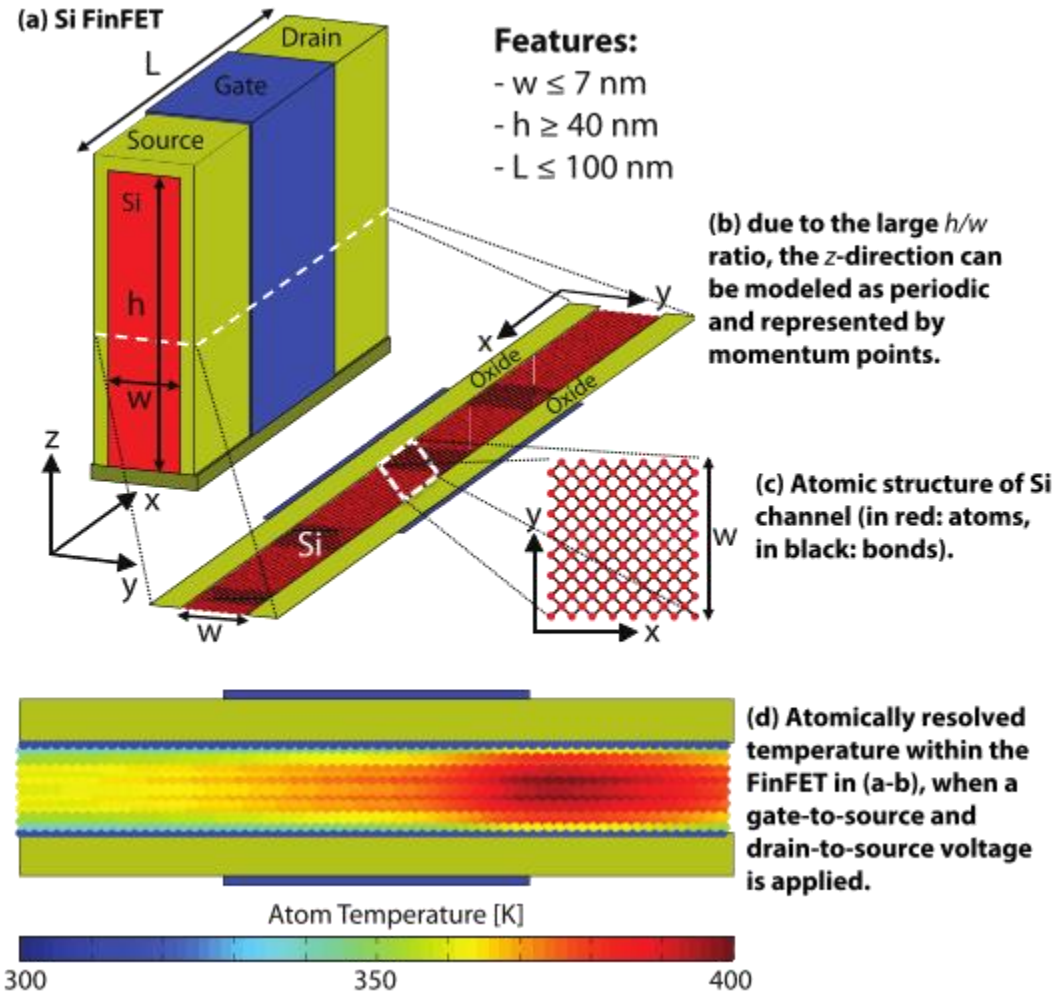
Next-Generation Transistors need to be cooler – addressing self-heating

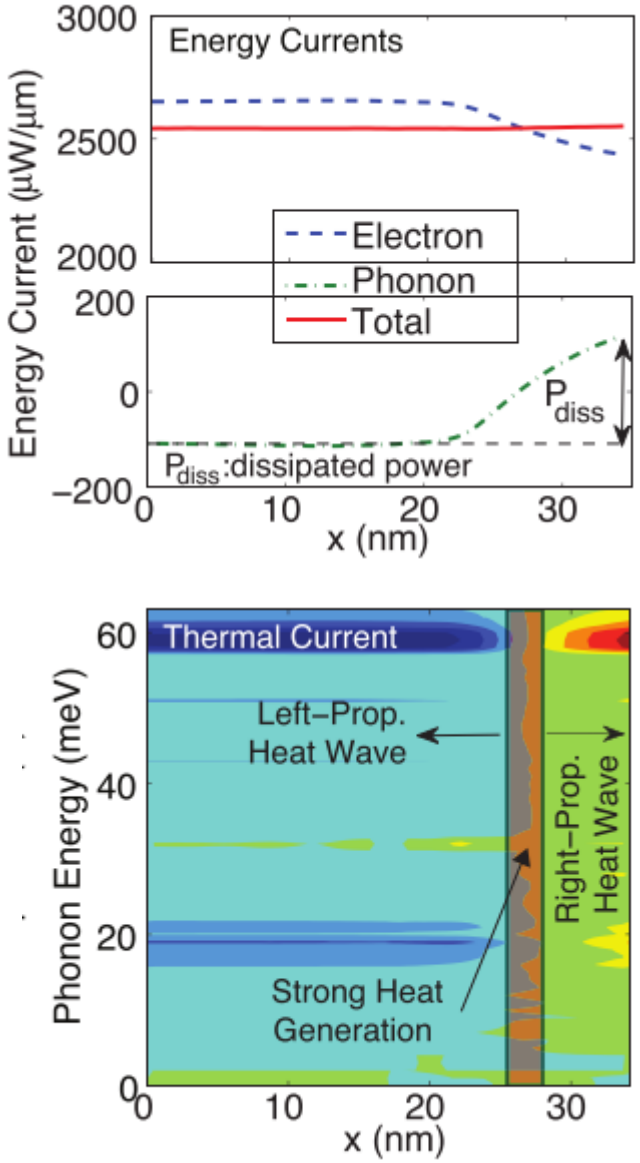


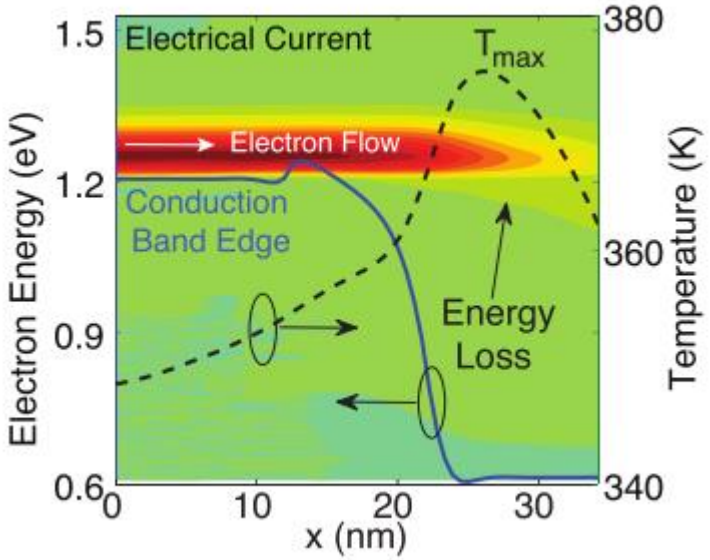




Next-Generation Transistors need to be cooler – addressing self-heating







Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

GF

Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

Electrons $G(E, k_z)$

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

GF

Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

GF

Electrons $G(E, k_z)$

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

Phonons $D(\omega, q_z)$

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

Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

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

Electrons $G(E, k_z)$

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

Phonons $D(\omega, q_z)$

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

GF

Quantum Transport Simulations with OMEN

- **OMEN Code (Luisier et al., Gordon Bell award finalist 2011 and 2015)**
  - 90k SLOC, C, C++, CUDA, MPI, OpenMP, …

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

Electrons $G(E, k_z)$

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

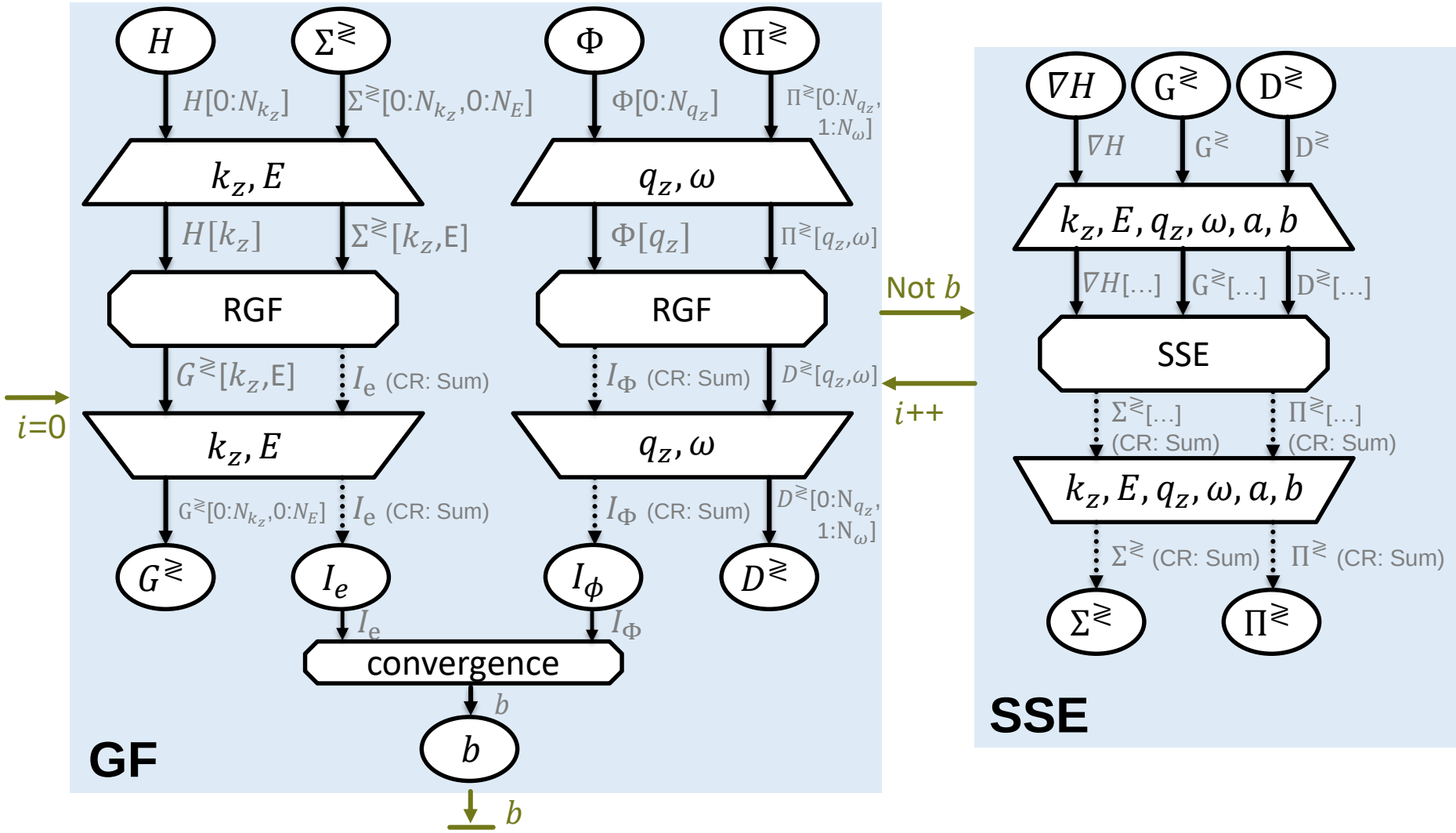
Phonons $D(\omega, q_z)$

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

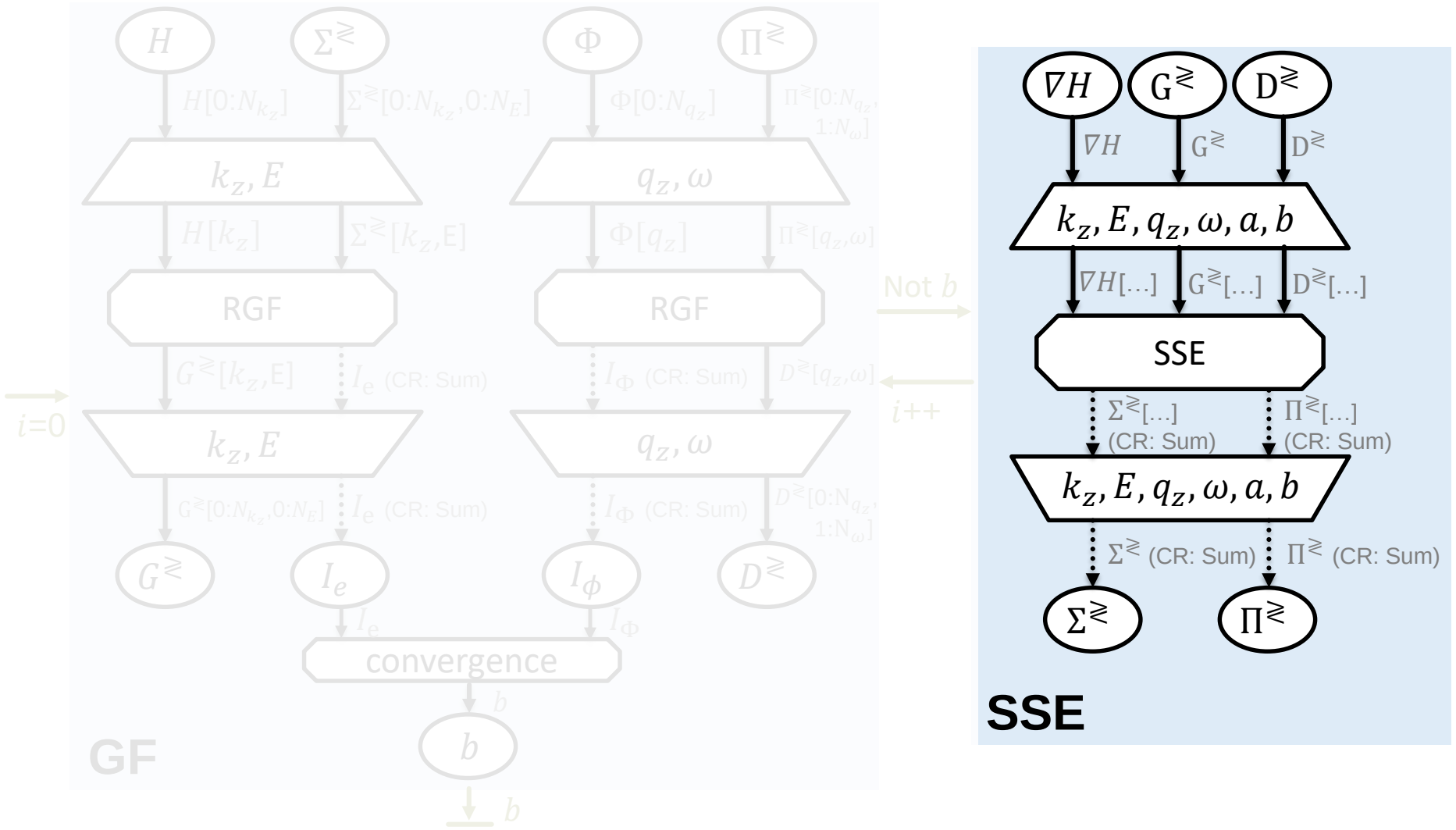
GF

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

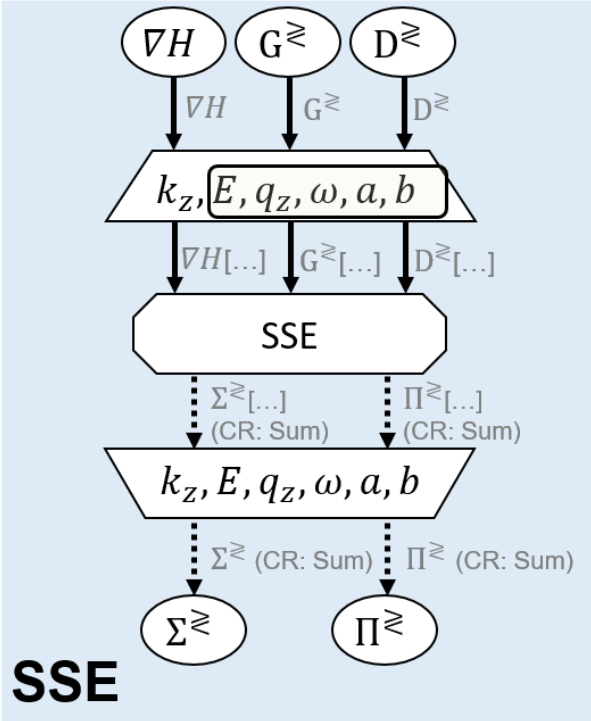
All of OMEN (90k SLOC) in a single SDFG – (collapsed) tasklets contain more SDFGs

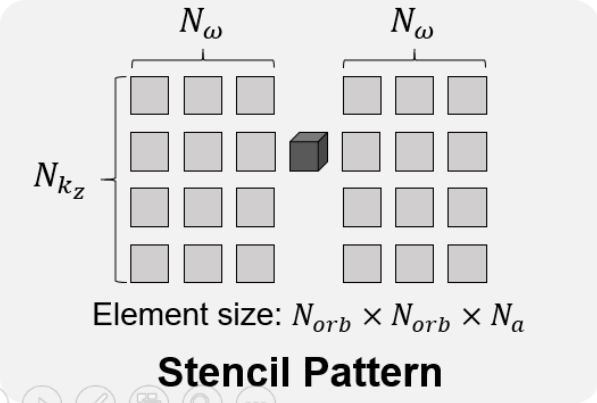


All of OMEN (90k SLOC) in a single SDFG – (collapsed) tasklets contain more SDFGs

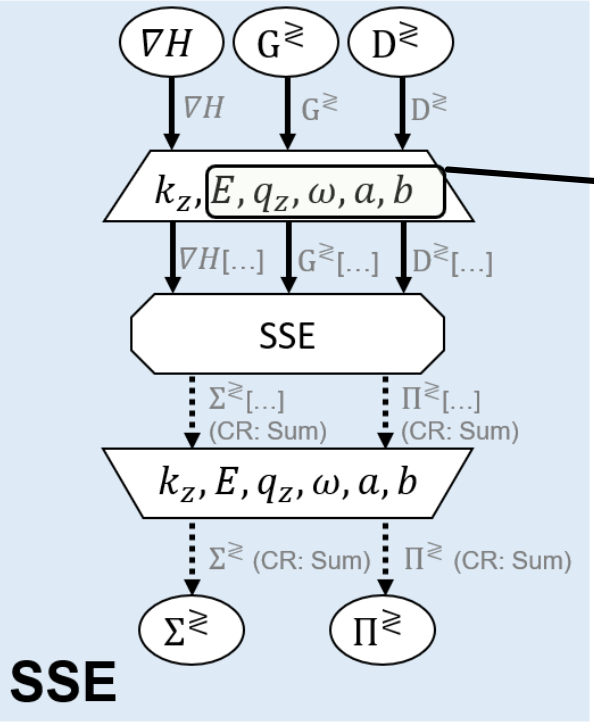


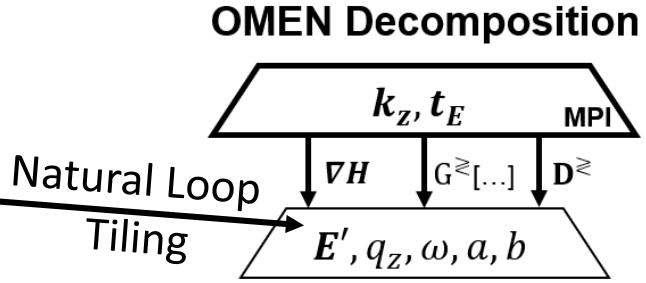
Zooming into SSE (large share of the runtime)

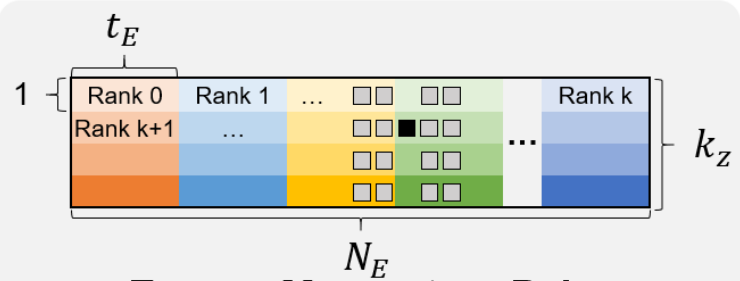




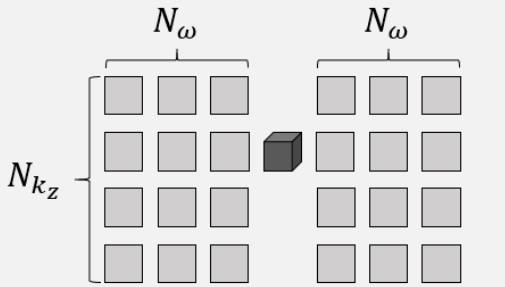
Zooming into SSE (large share of the runtime)





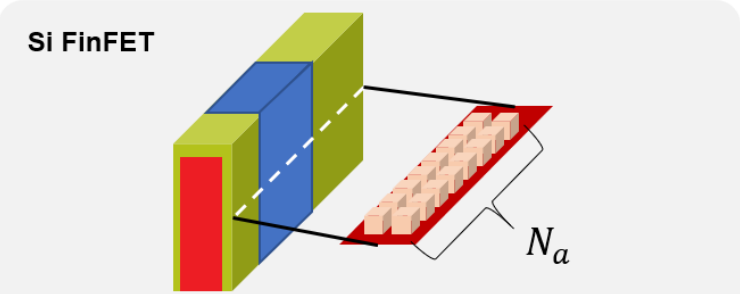


Energy-Momentum Pairs



Element size: $N_{orb} \times N_{orb} \times N_a$

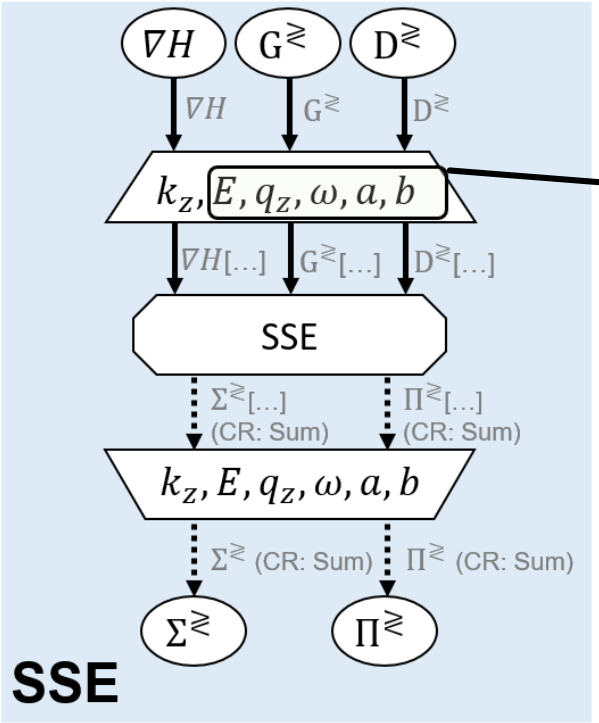
Stencil Pattern

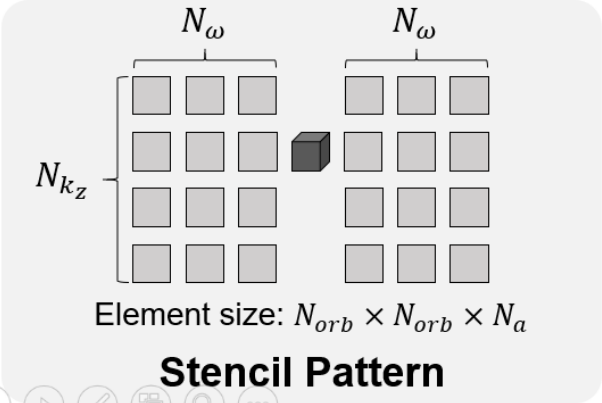


Nanodevice View

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

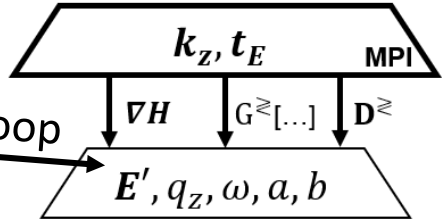
Zooming into SSE (large share of the runtime)

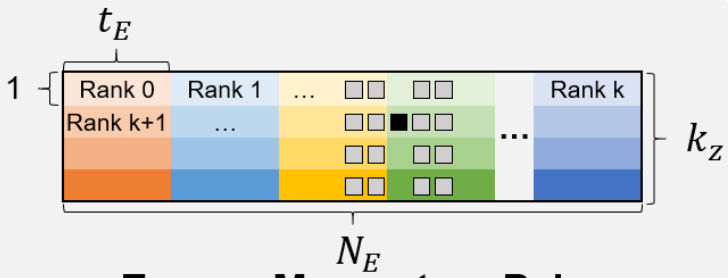




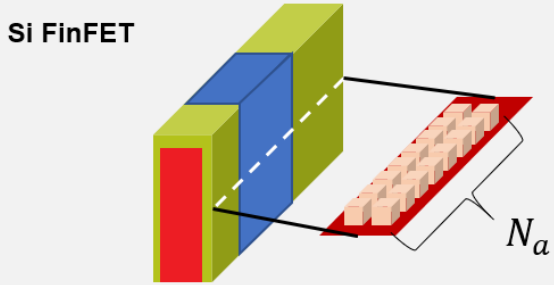
Natural Loop Tiling

OMEN Decomposition





Energy-Momentum Pairs

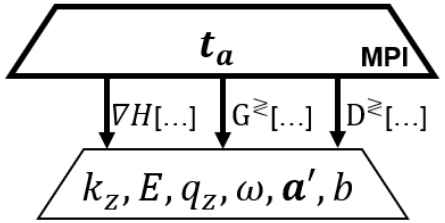


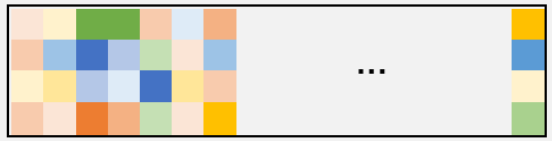
Nanodevice View

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

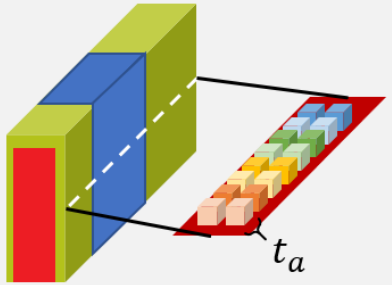
DaCe Transform

Data-Centric Decomposition





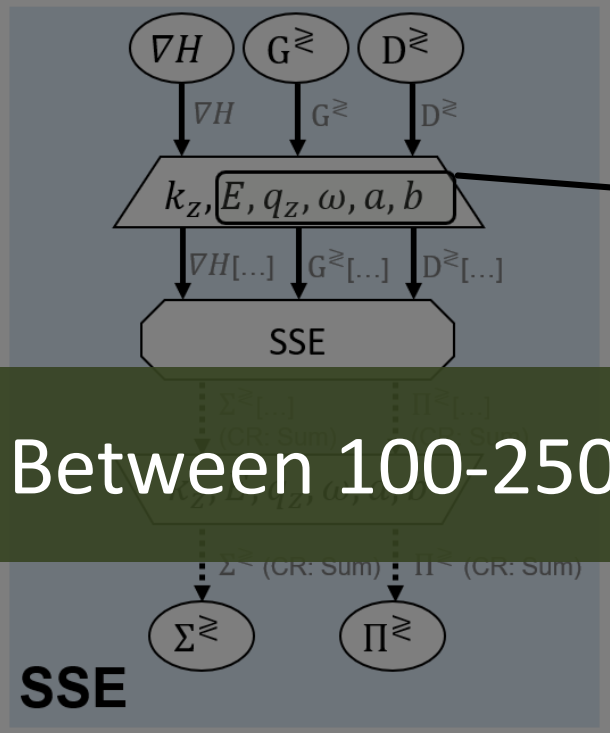
Energy-Momentum Pairs

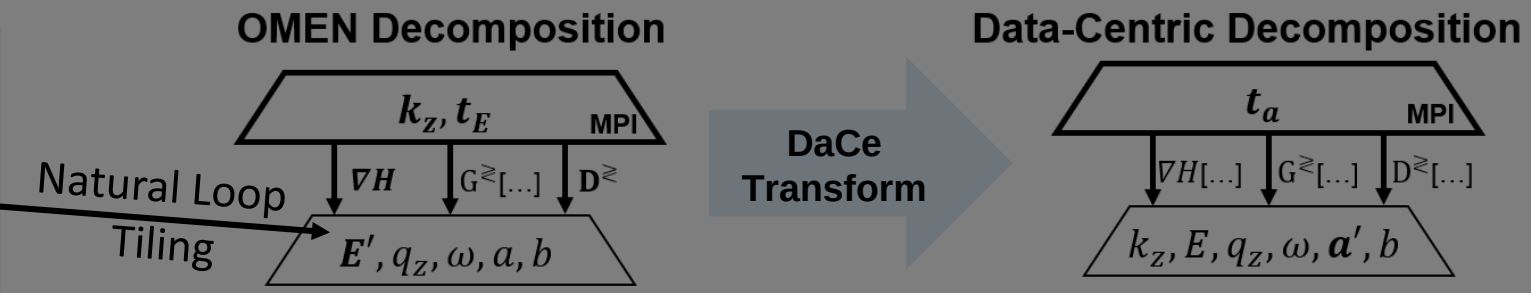


Nanodevice View

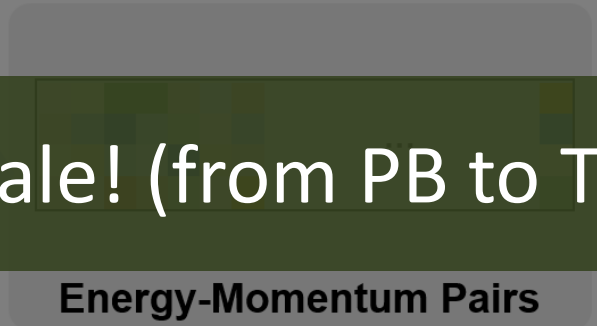
Method: Collective Alltoall
Volume: $\mathcal{O}(N_{k_z}(N_E + N_\omega)(N_a + N_b)\, N_{orb}^2)$
MPI Invocations: 4

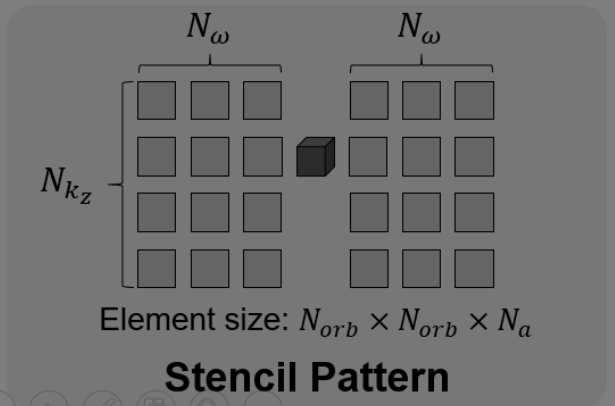
Zooming into SSE (large share of the runtime)

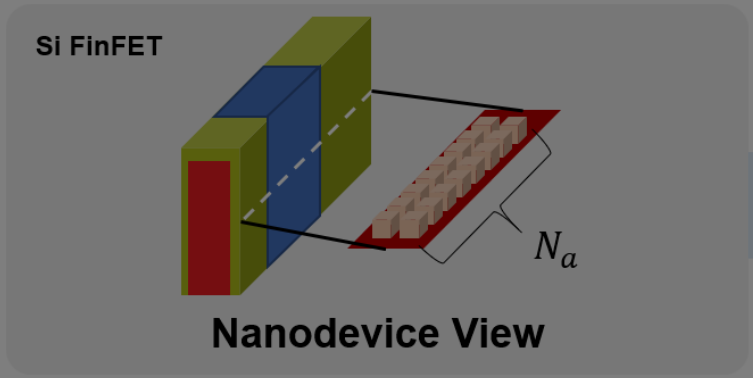


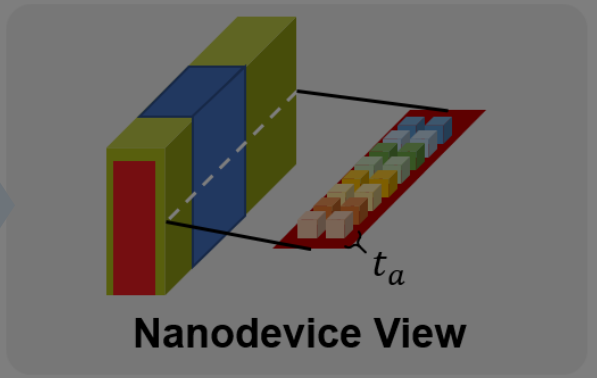


Between 100-250x less communication at scale! (from PB to TB)









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

Additional interesting performance insights

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

Piz Daint single node (P100)

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

Basic operation in SSE (many very small MMMs)

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

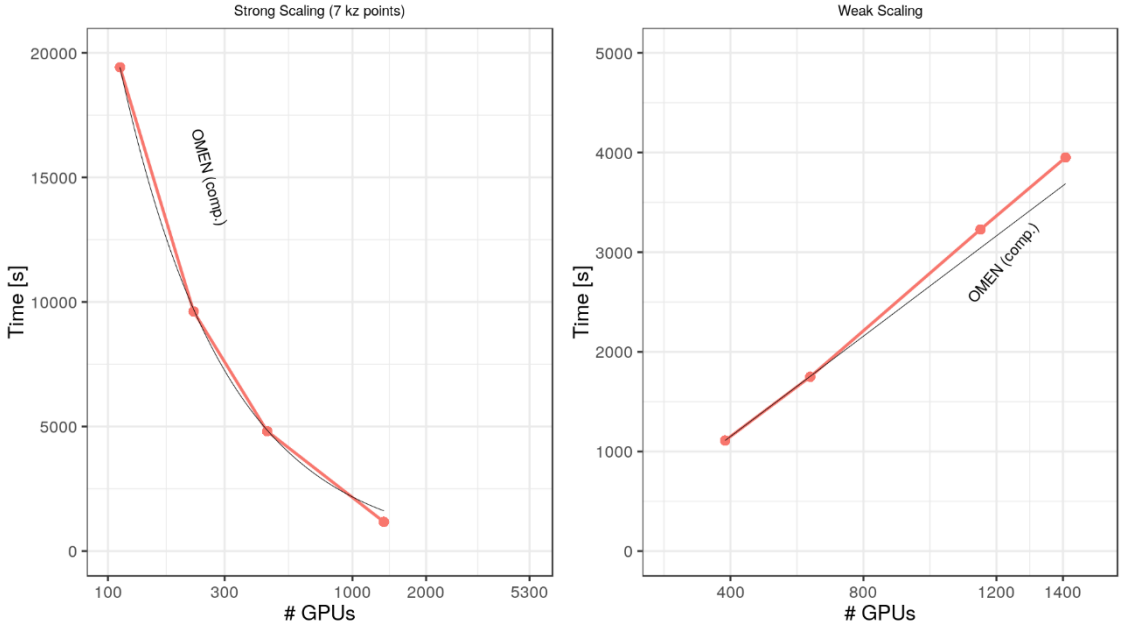
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

Basic operation in SSE (many very small MMMs)



5k atoms

Piz Daint

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

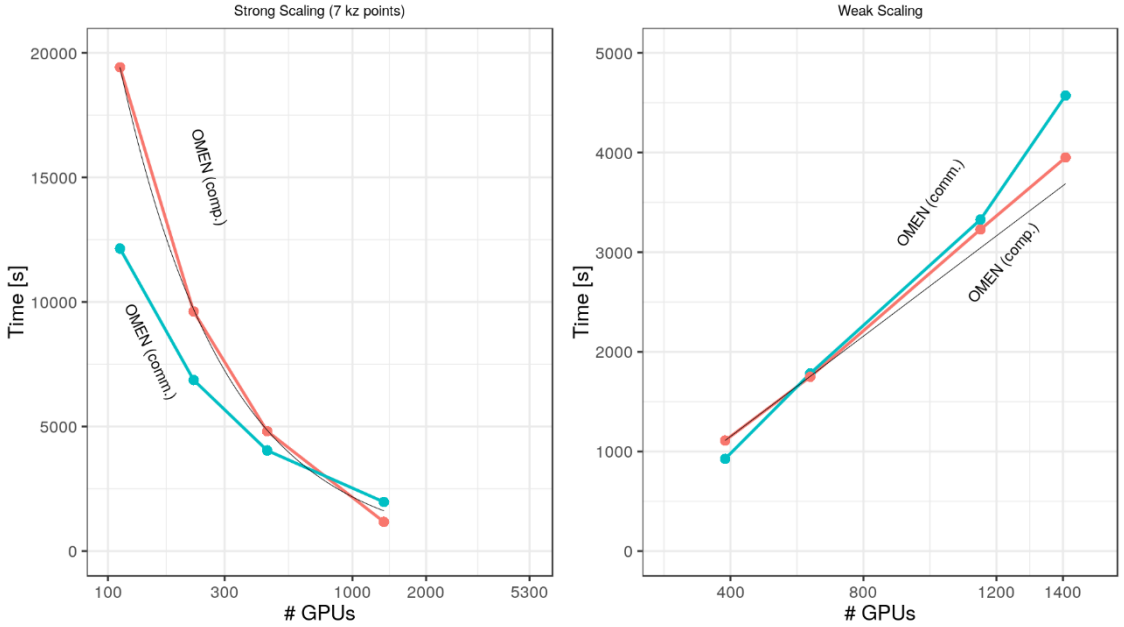
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

Basic operation in SSE (many very small MMMs)



5k atoms

Piz Daint

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

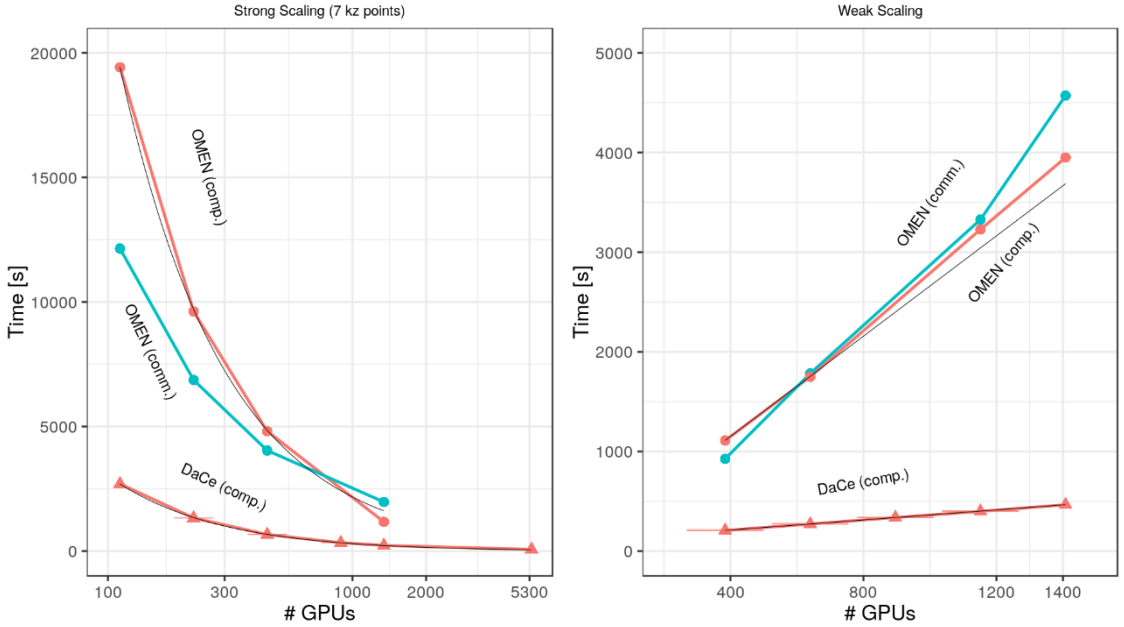
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

Basic operation in SSE (many very small MMMs)



5k atoms

Piz Daint

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

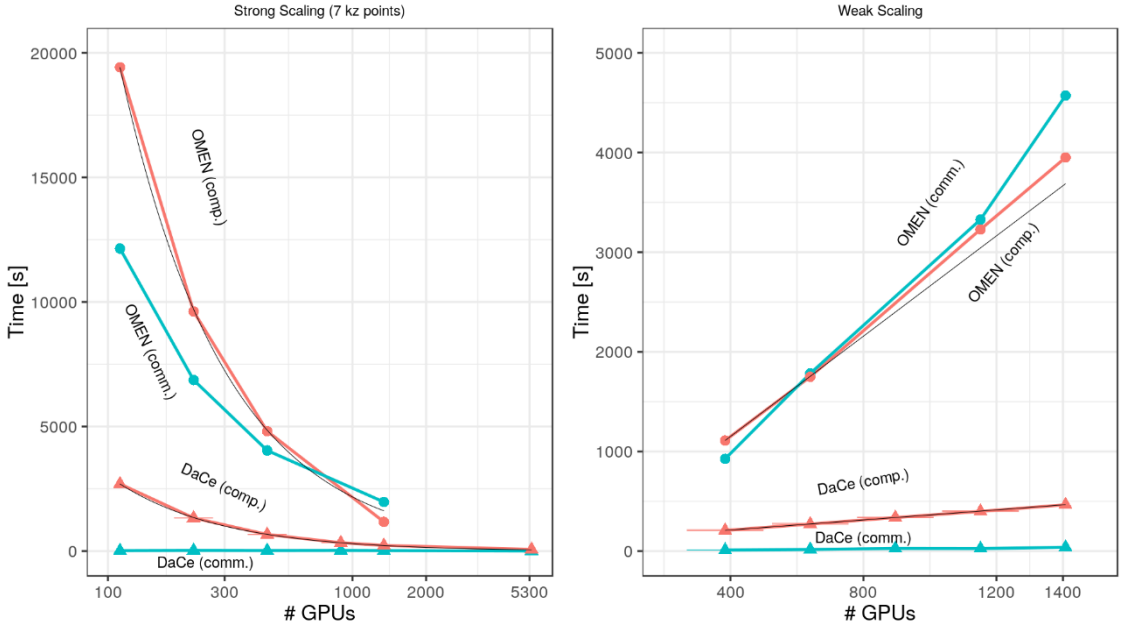
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

Basic operation in SSE (many very small MMMs)



5k atoms

Piz Daint

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

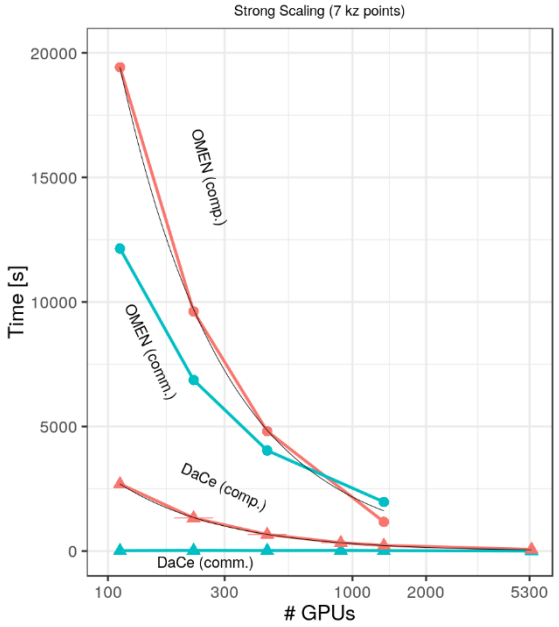
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

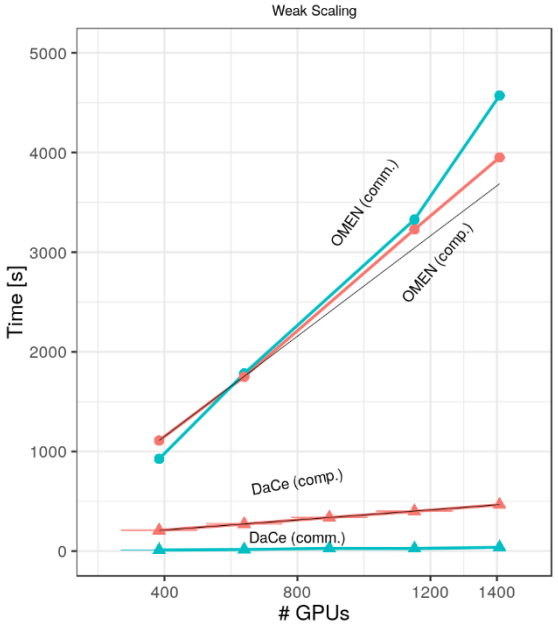
Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

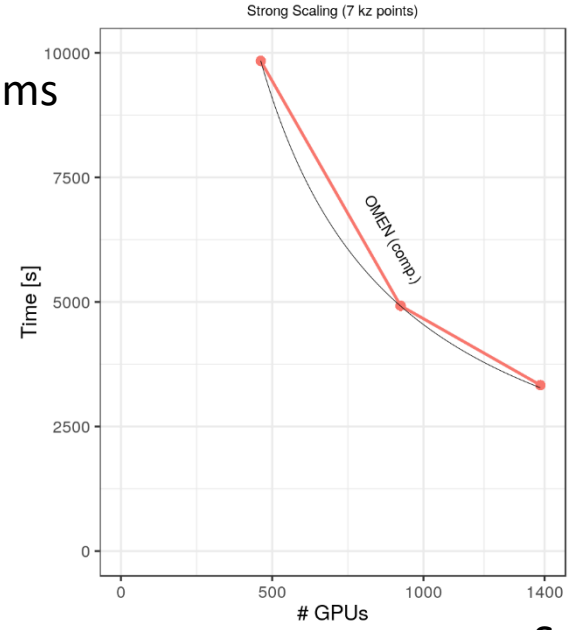
GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

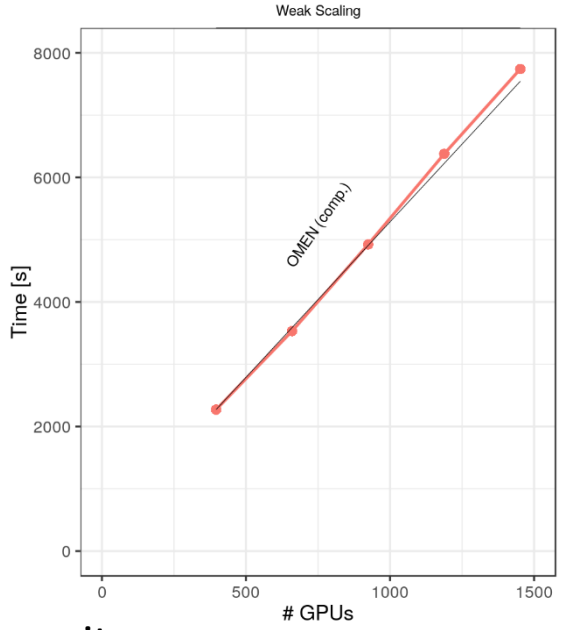
Basic operation in SSE (many very small MMMs)





5k atoms





Piz Daint

Summit

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

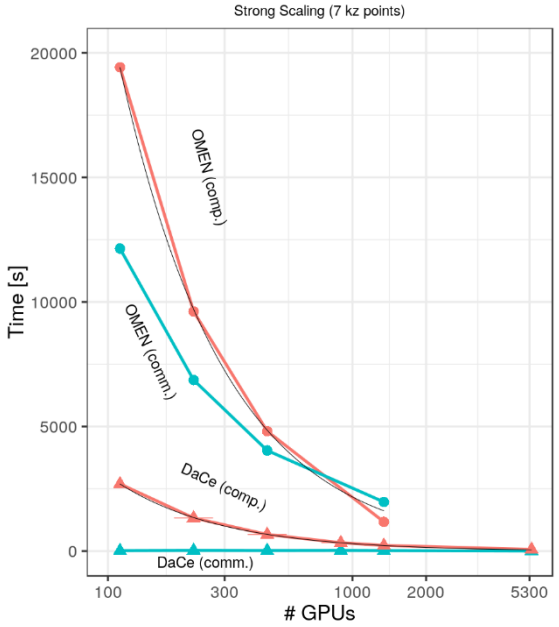
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

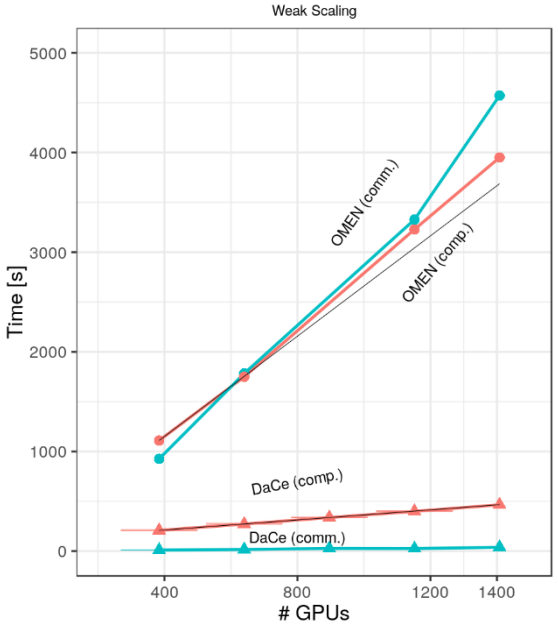
Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

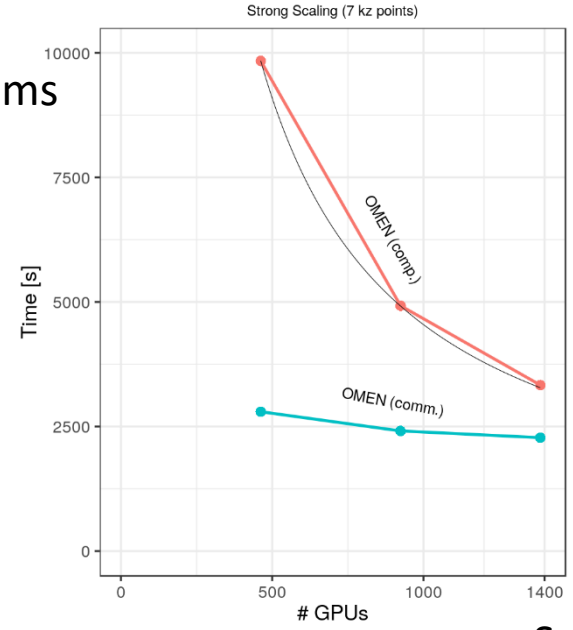
GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

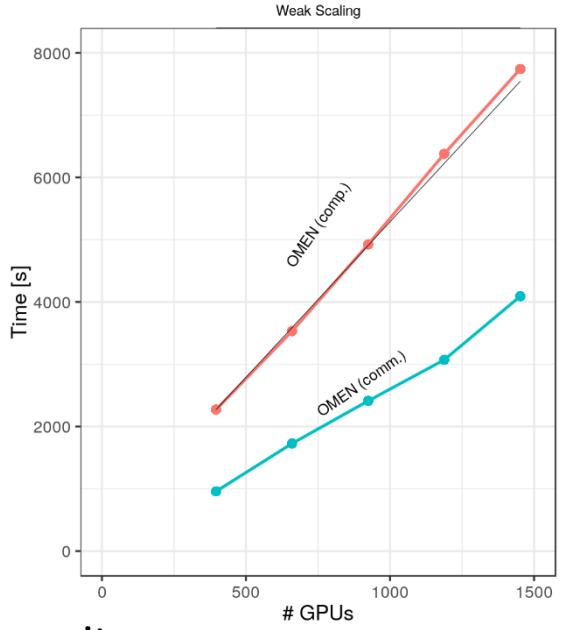
Basic operation in SSE (many very small MMMs)





5k atoms





Piz Daint

Summit

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

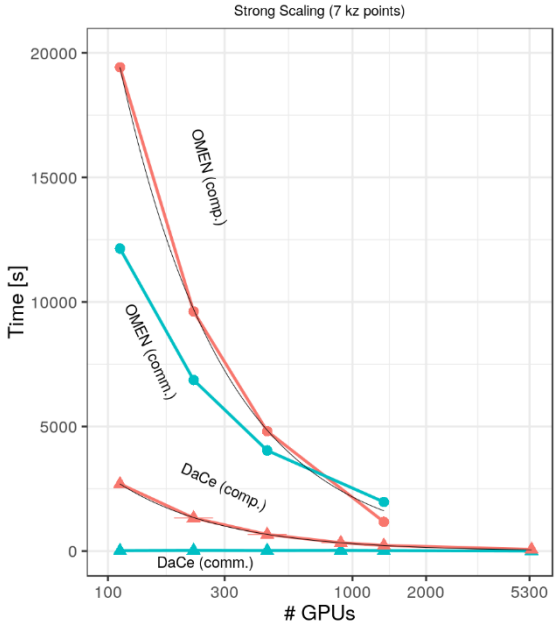
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

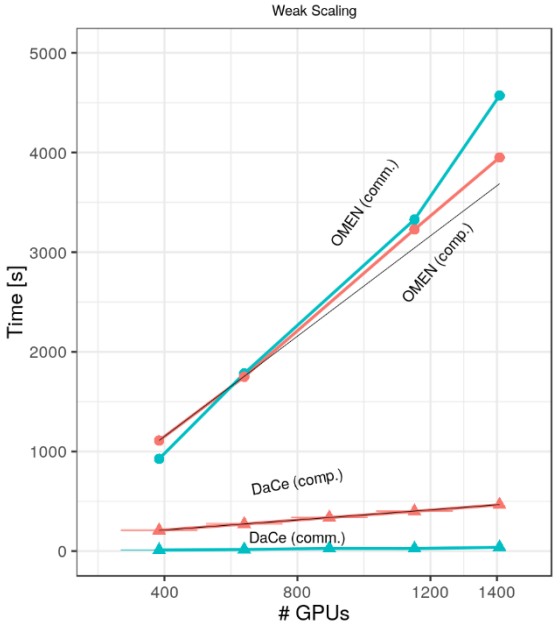
Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

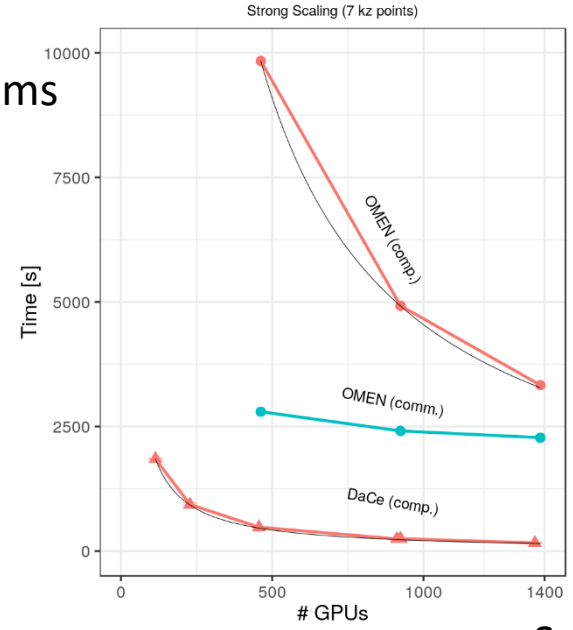
Basic operation in SSE (many very small MMMs)

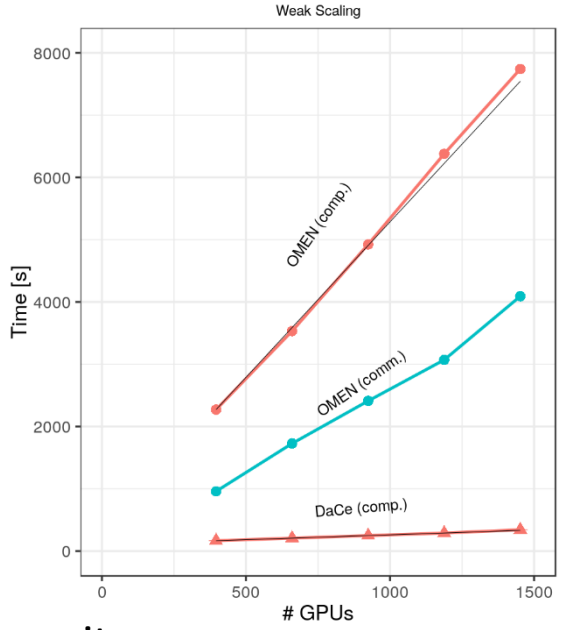




5k atoms

Piz Daint





Summit

Additional interesting performance insights

Python is slow! Ok, we knew that – but compiled can be fast!

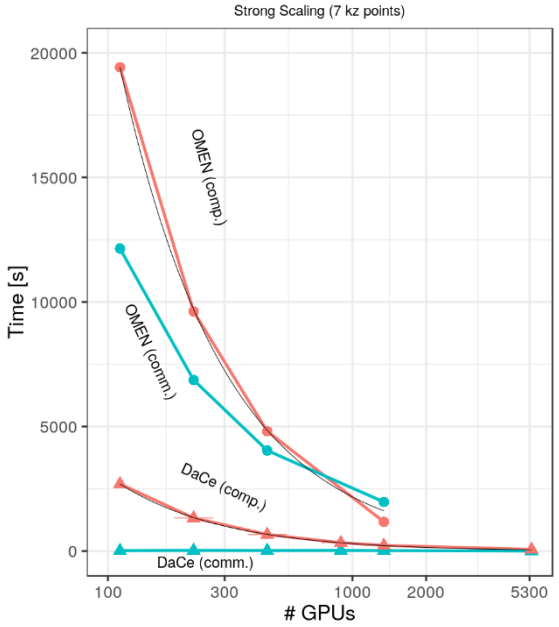
Variant	Phase					
	GF			SSE		
	Tflop	Time [s]	% Peak	Tflop	Time [s]	% Peak
OMEN	174.0	144.14	23.2%	63.6	965.45	1.3%
Python	174.0	1,342.77	2.5%	63.6	30,560.13	0.2%
DaCe	174.0	**111.25**	**30.1%**	31.8	**29.93**	**20.4%**

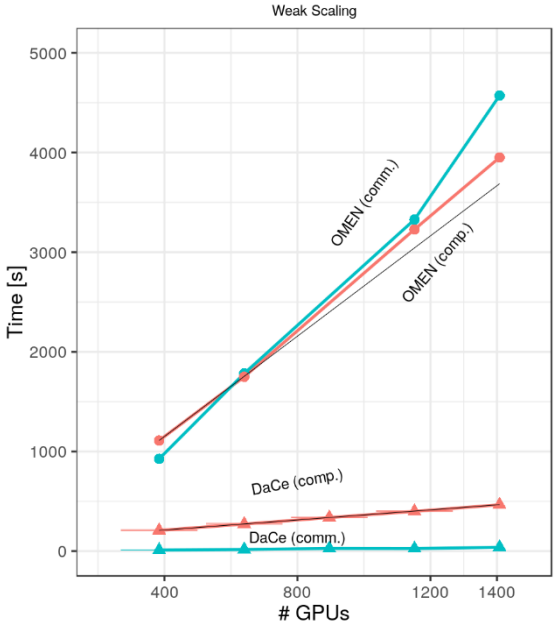
Piz Daint single node (P100)

cuBLAS can be very inefficient (well, unless you floptimize)

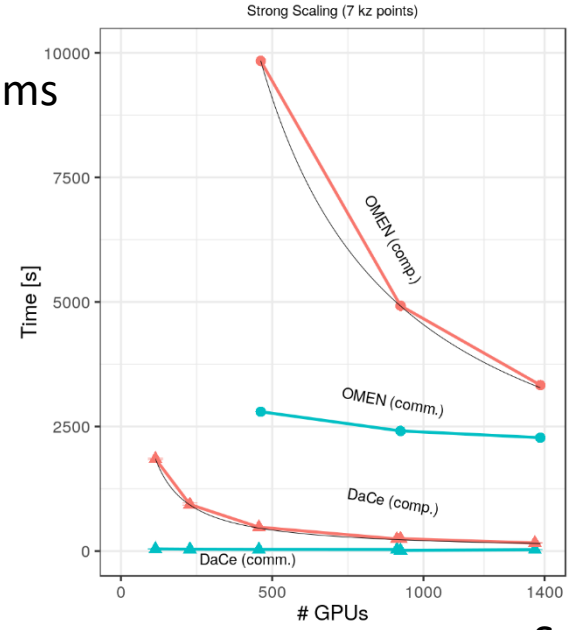
GPU	cuBLAS			DaCe (SBSMM)		
	Gflop	Time	% Peak (Useful)	Gflop	Time	% Peak
P100	27.42	6.73 ms	86.6% (6.1%)	1.92	4.03 ms	10.1%
V100	27.42	4.62 ms	84.8% (5.9%)	1.92	0.97 ms	28.3%

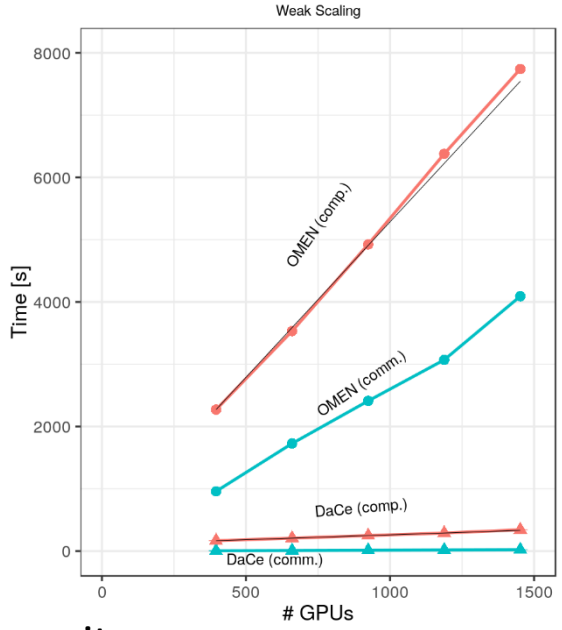
Basic operation in SSE (many very small MMMs)





5k atoms



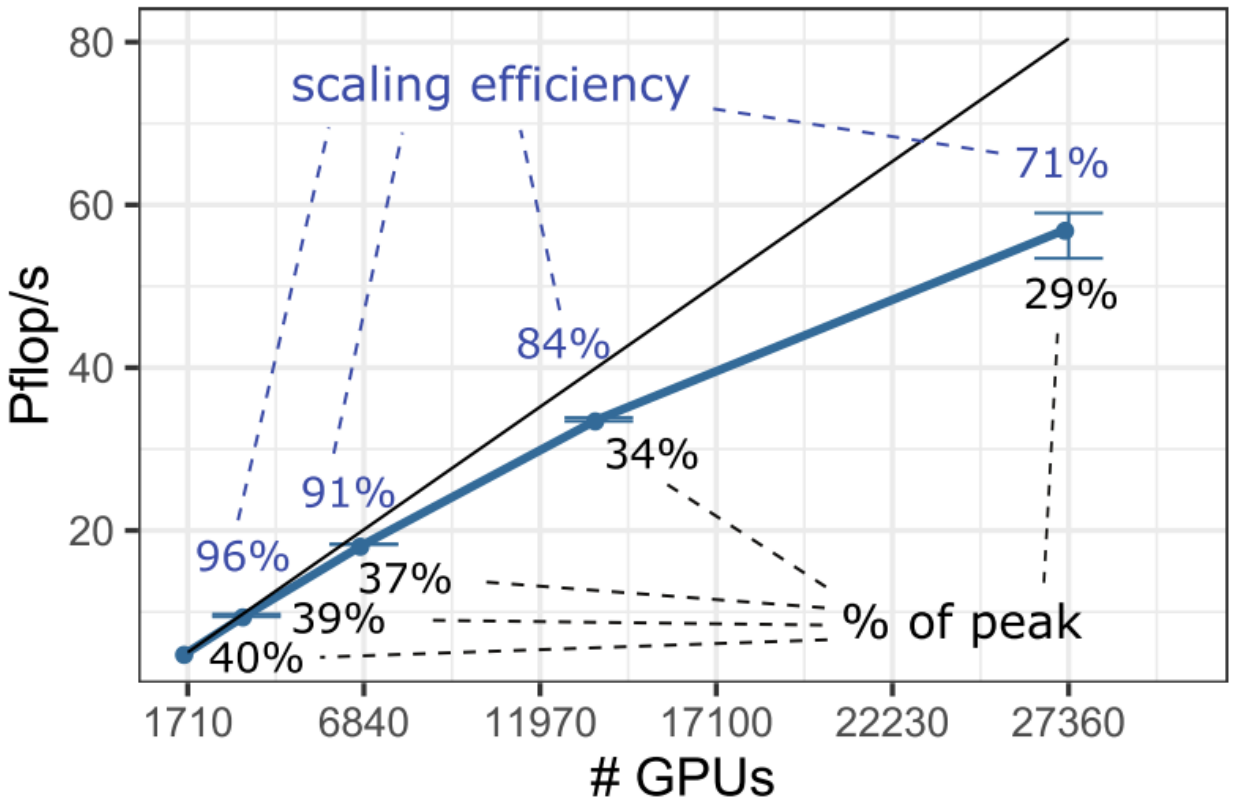


Piz Daint

Summit

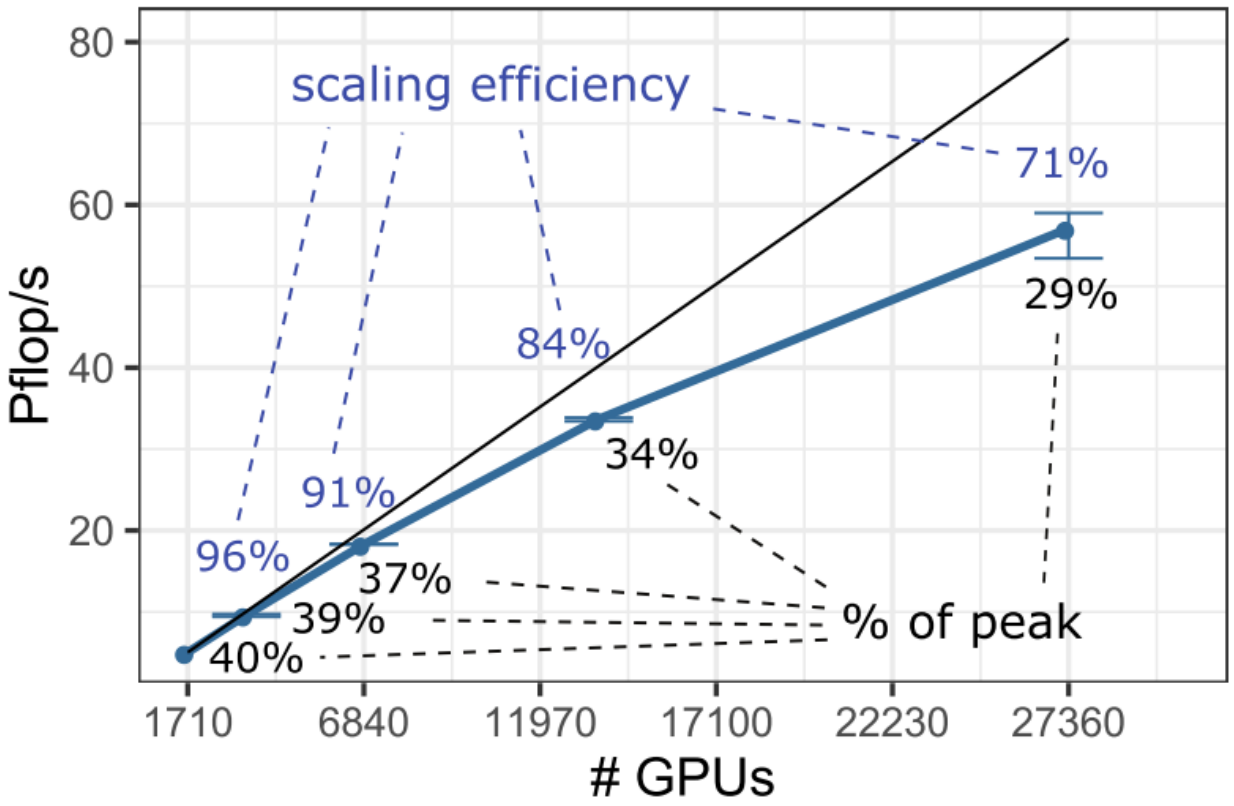
10,240 atoms on 27,360 V100 GPUs (full-scale Summit)

- 56 Pflop/s with I/O (28% peak)



10,240 atoms on 27,360 V100 GPUs (full-scale Summit)

- 56 Pflop/s with I/O (28% peak)

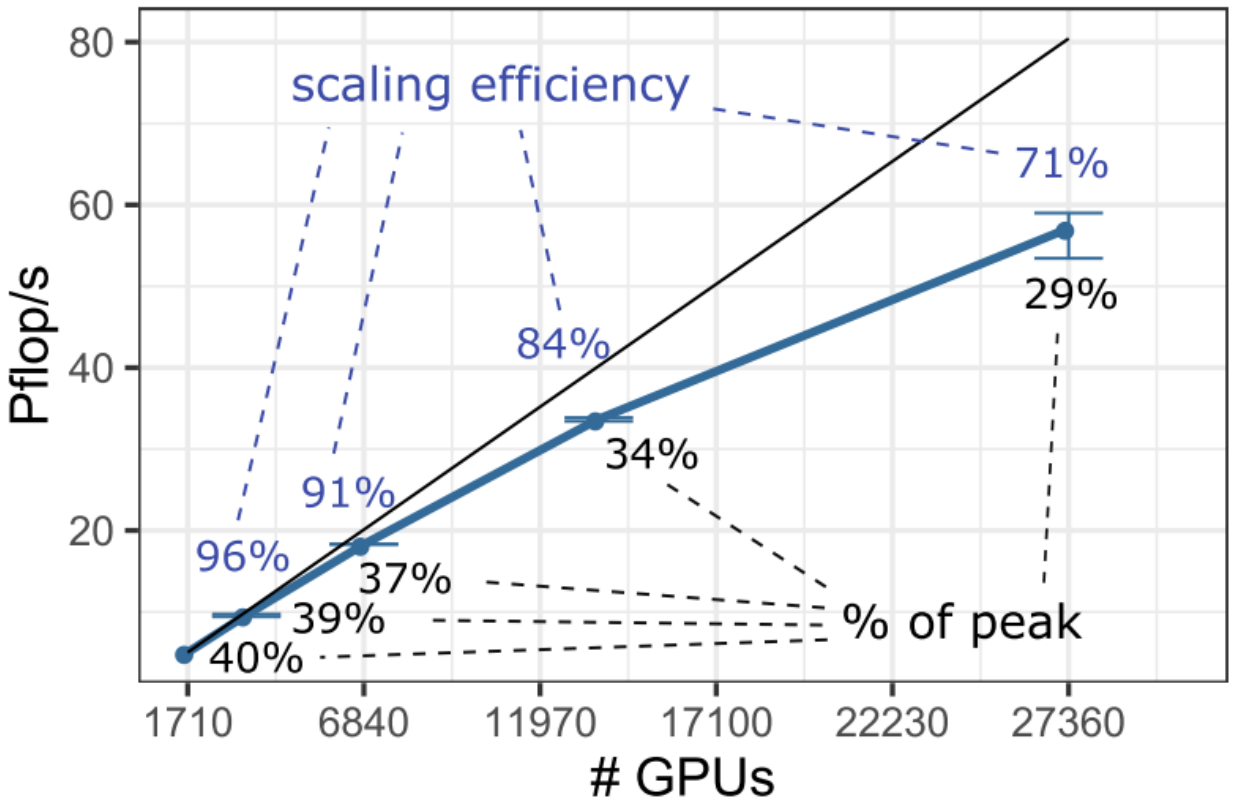


Variant	N_a	Time [s]	Time/Atom [s]	Speedup
OMEN	1,064	4695.70	4.413	1.0x
DaCe	10,240	489.83	0.048	92.3x

$P = 6,840$, $N_b = 34$, $N_{orb} = 12$, $N_E = 1,220$, $N_\omega = 70$.

10,240 atoms on 27,360 V100 GPUs (full-scale Summit)

- 56 Pflop/s with I/O (28% peak)



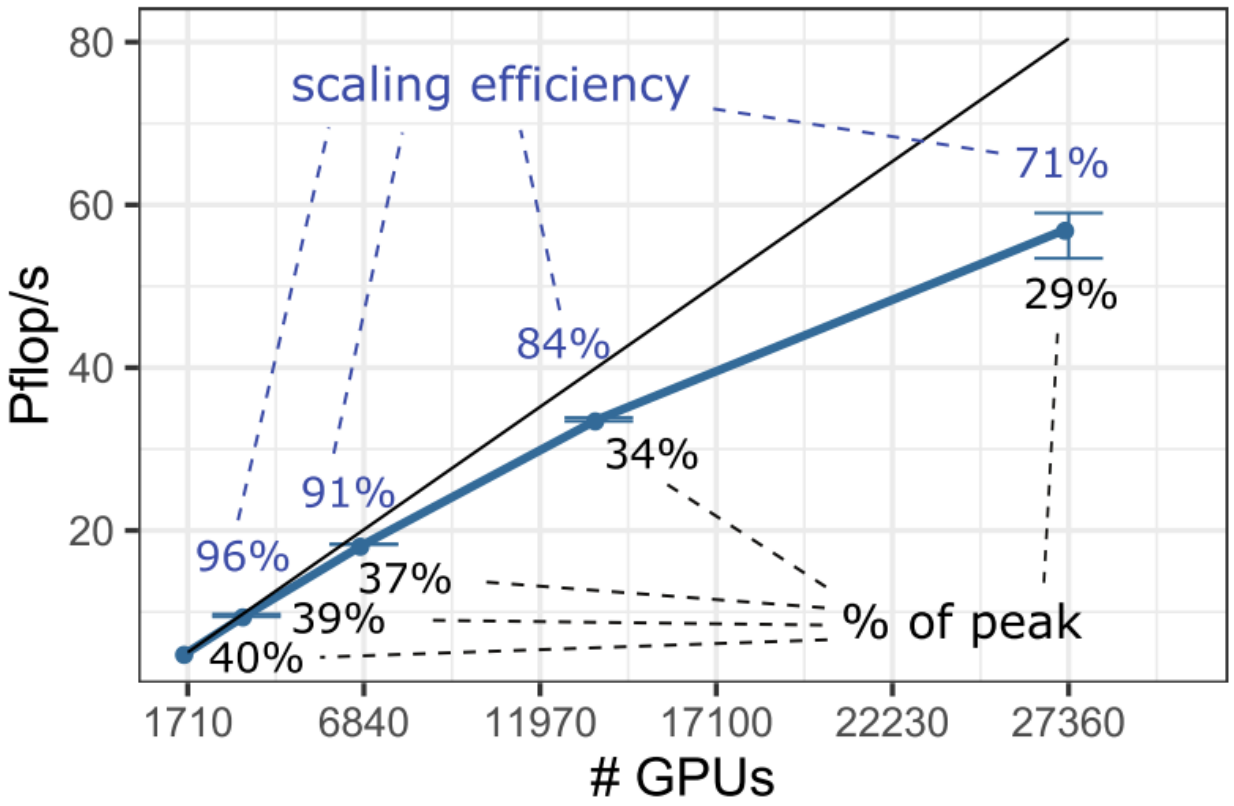
Variant	N_a	Time [s]	Time/Atom [s]	Speedup
OMEN	1,064	4695.70	4.413	1.0x
DaCe	10,240	489.83	0.048	92.3x

$P = 6,840$, $N_b = 34$, $N_{orb} = 12$, $N_E = 1,220$, $N_\omega = 70$.

Already ~100x speedup on 25% of Summit – the original OMEN does not scale further!

10,240 atoms on 27,360 V100 GPUs (full-scale Summit)

- 56 Pflop/s with I/O (28% peak)



Variant	N_a	Time [s]	Time/Atom [s]	Speedup
OMEN	1,064	4695.70	4.413	1.0x
DaCe	10,240	489.83	0.048	92.3x

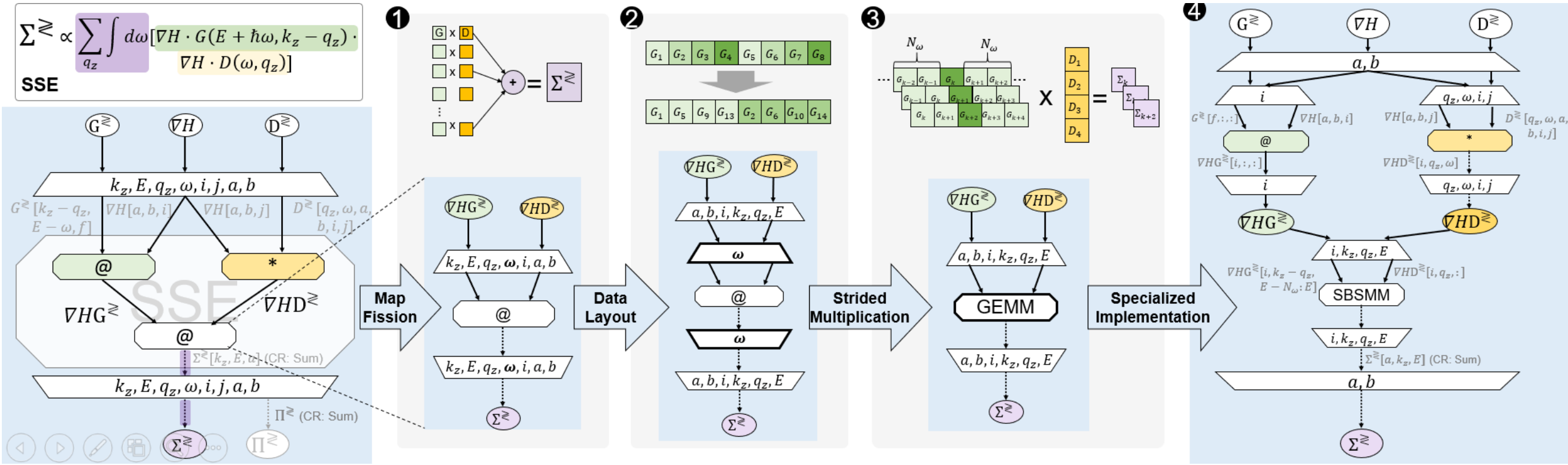
$P = 6,840$, $N_b = 34$, $N_{orb} = 12$, $N_E = 1,220$, $N_\omega = 70$.

Already ~100x speedup on 25% of Summit – the original OMEN does not scale further!

Communication time reduced by 417x on Piz Daint!

Volume on full-scale Summit from 12 PB/iter → 87 TB/iter

An example of fine-grained data-centric optimization (i.e., how to vectorize)

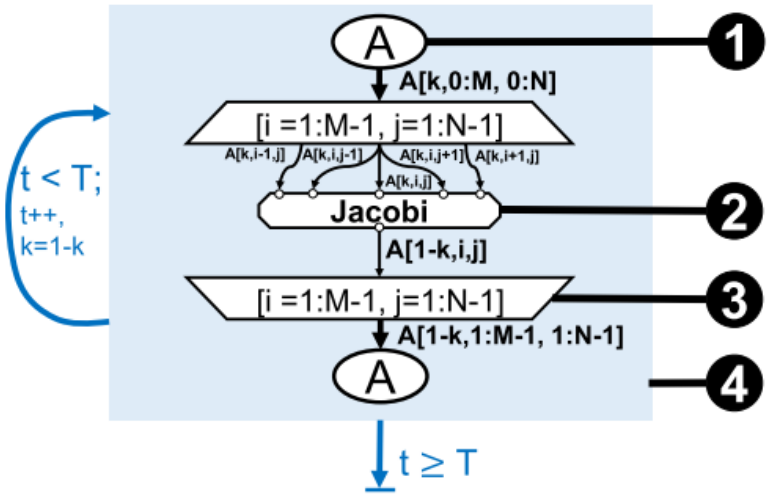


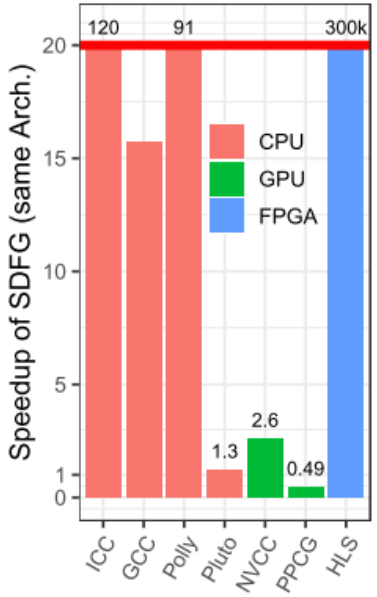
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

A closing note on FPGAs ☺ - kudos to Hal!

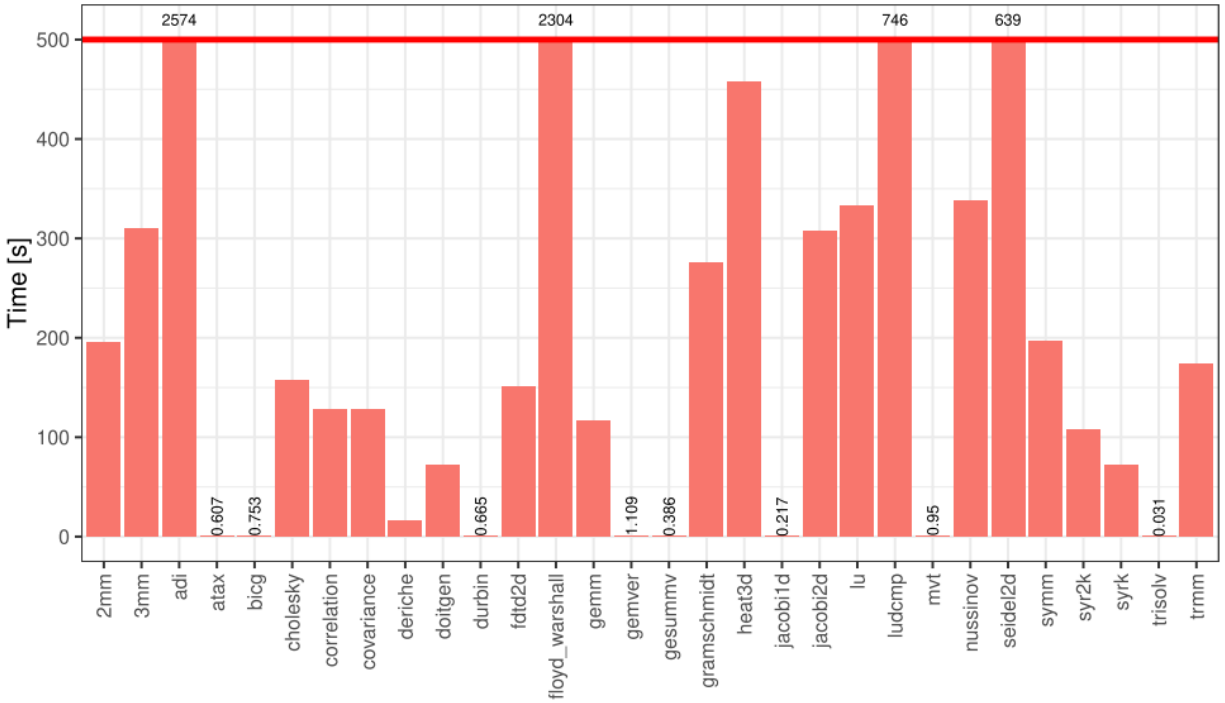
- **We heard today "OpenCL is not a great abstraction"**
- **Data-centric may be the solution – lay out wide pipelines (if data-independent) or streams (if data-dependent)**
  - Encouraging initial results

simple Jacobi stencil





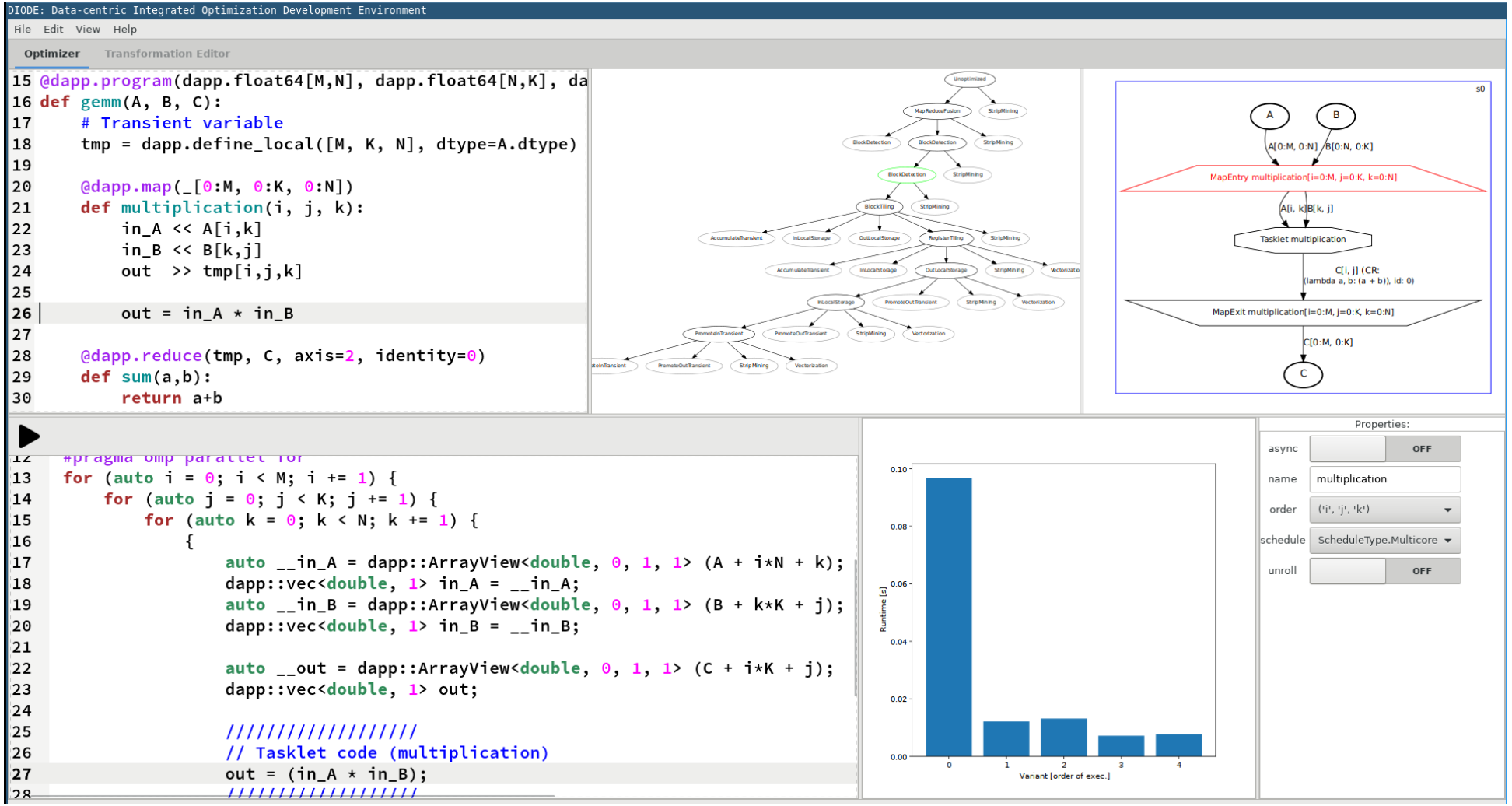
300k times speedup over Vivado HLS (OpenCL) ☺



First to run **all** of Polybench on FPGA (not simulated)!

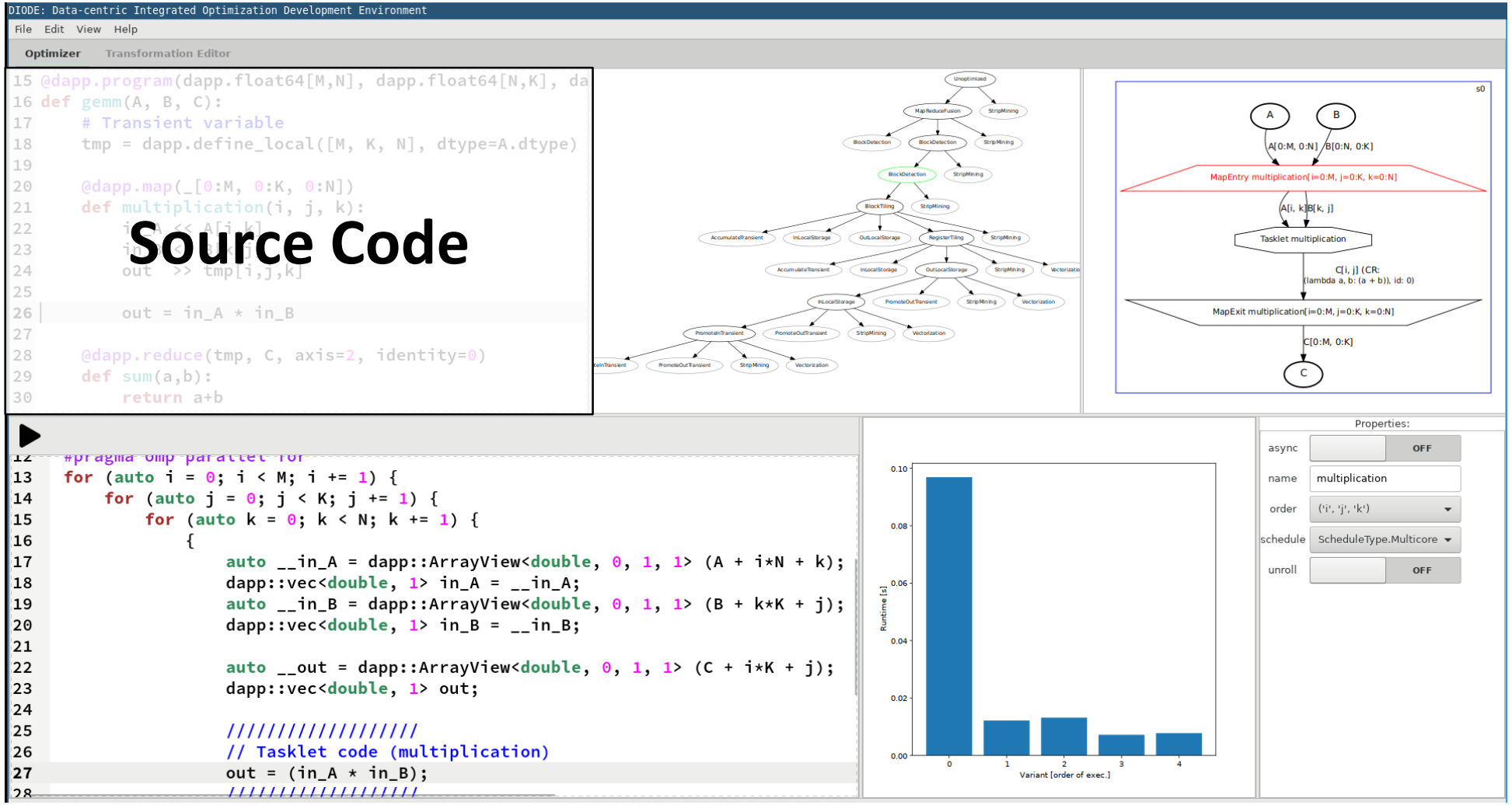
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

DIODE User Interface



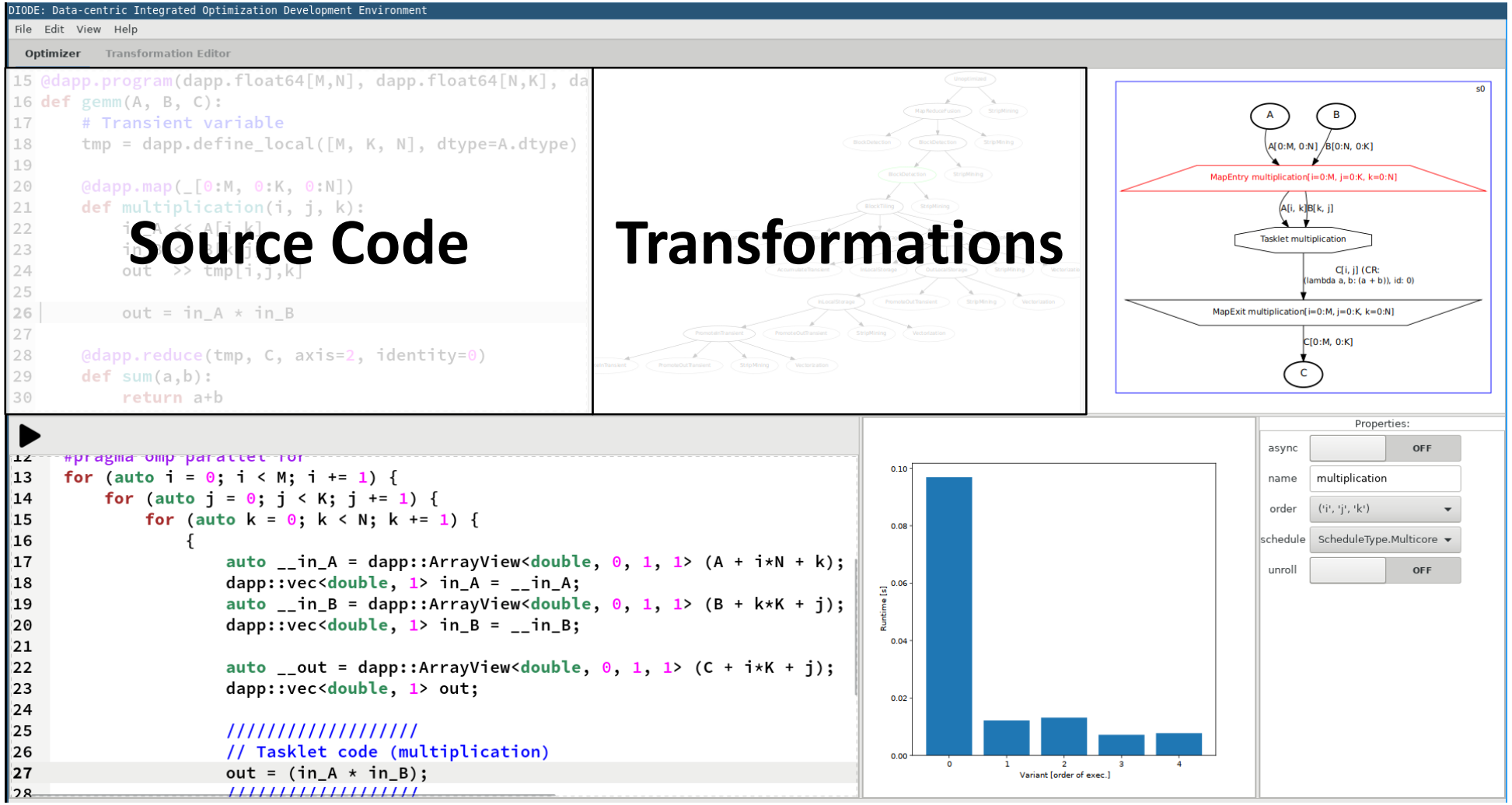
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

DIODE User Interface



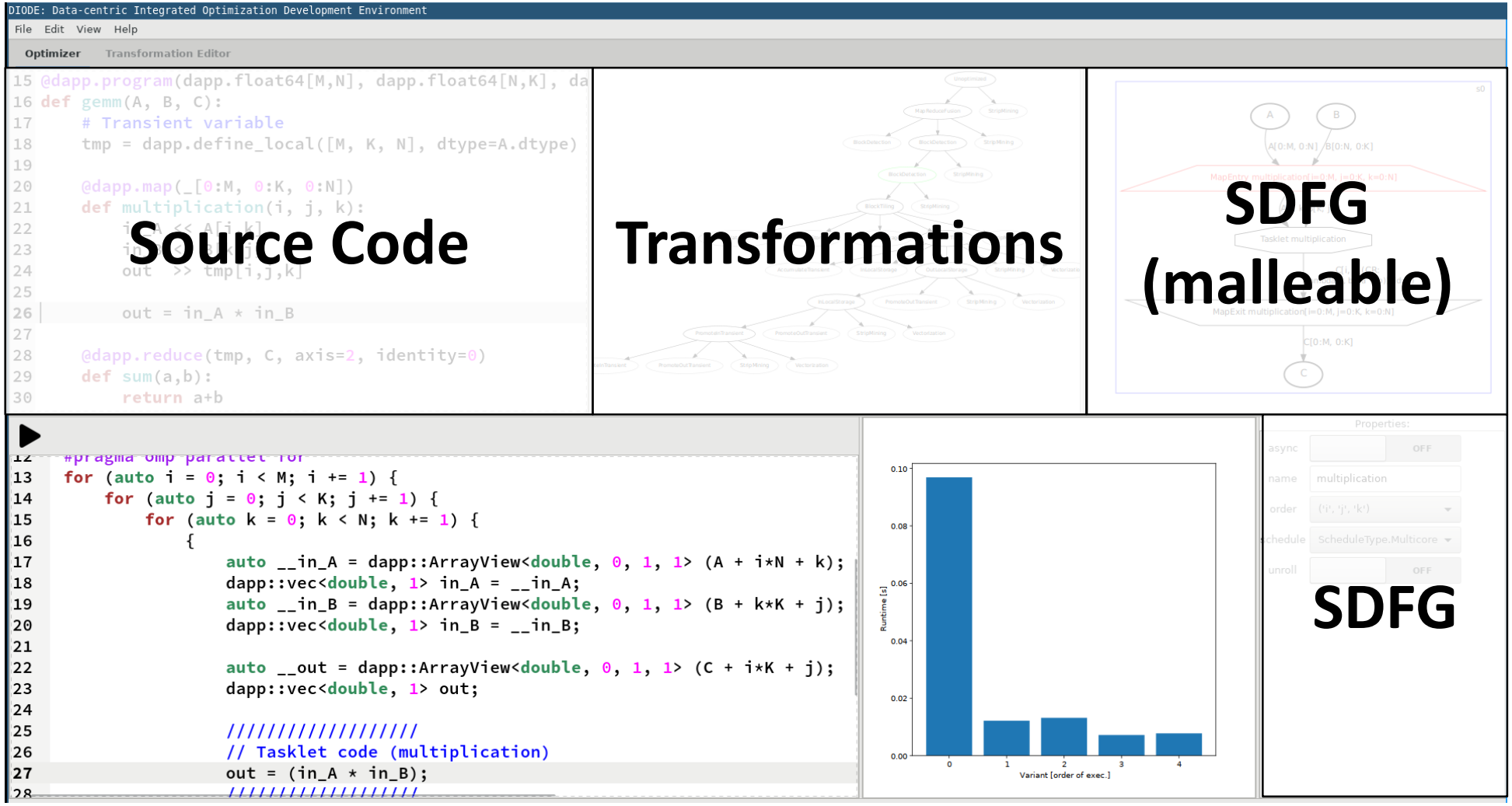
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

DIODE User Interface

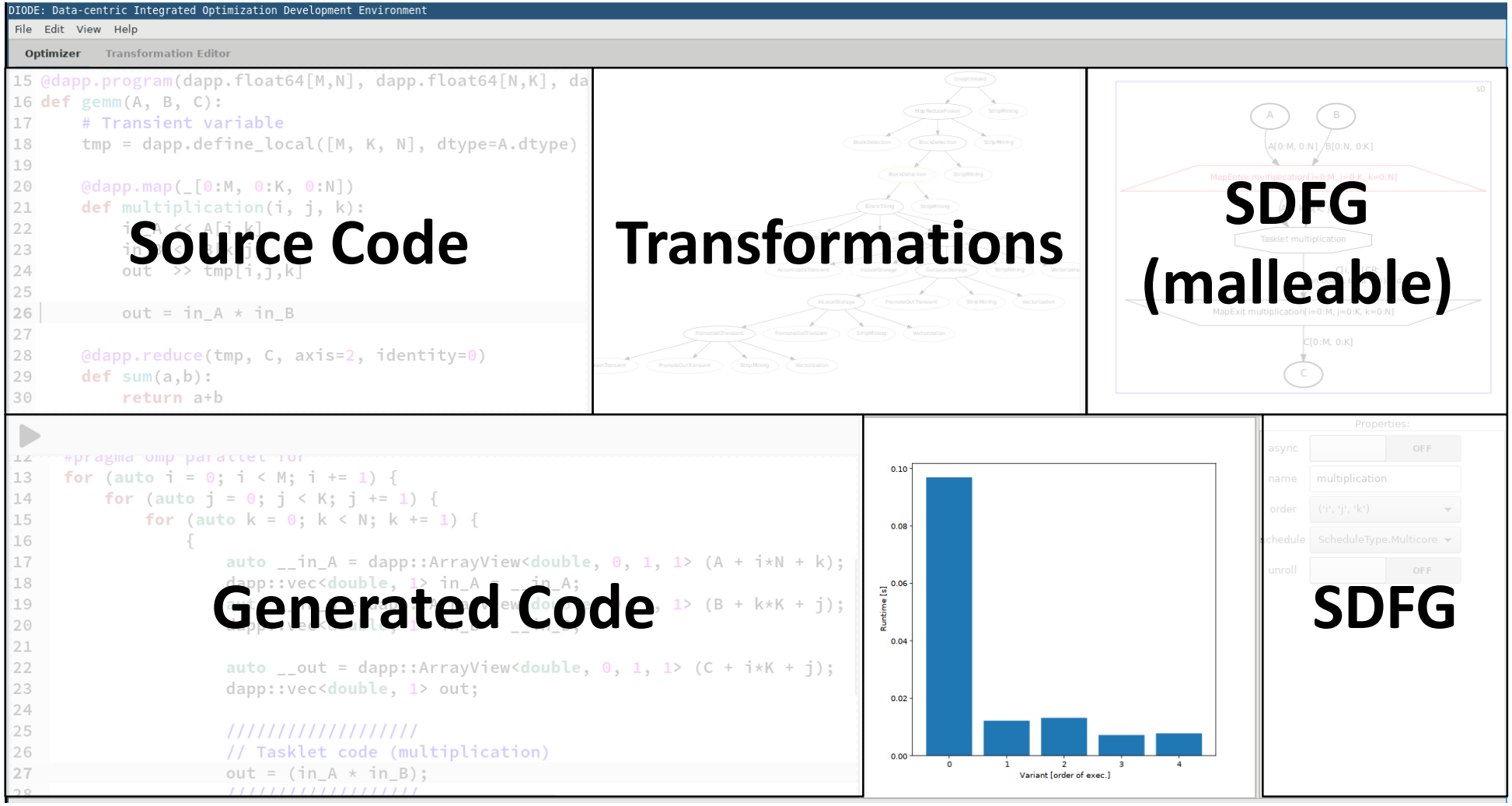


Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

DIODE User Interface

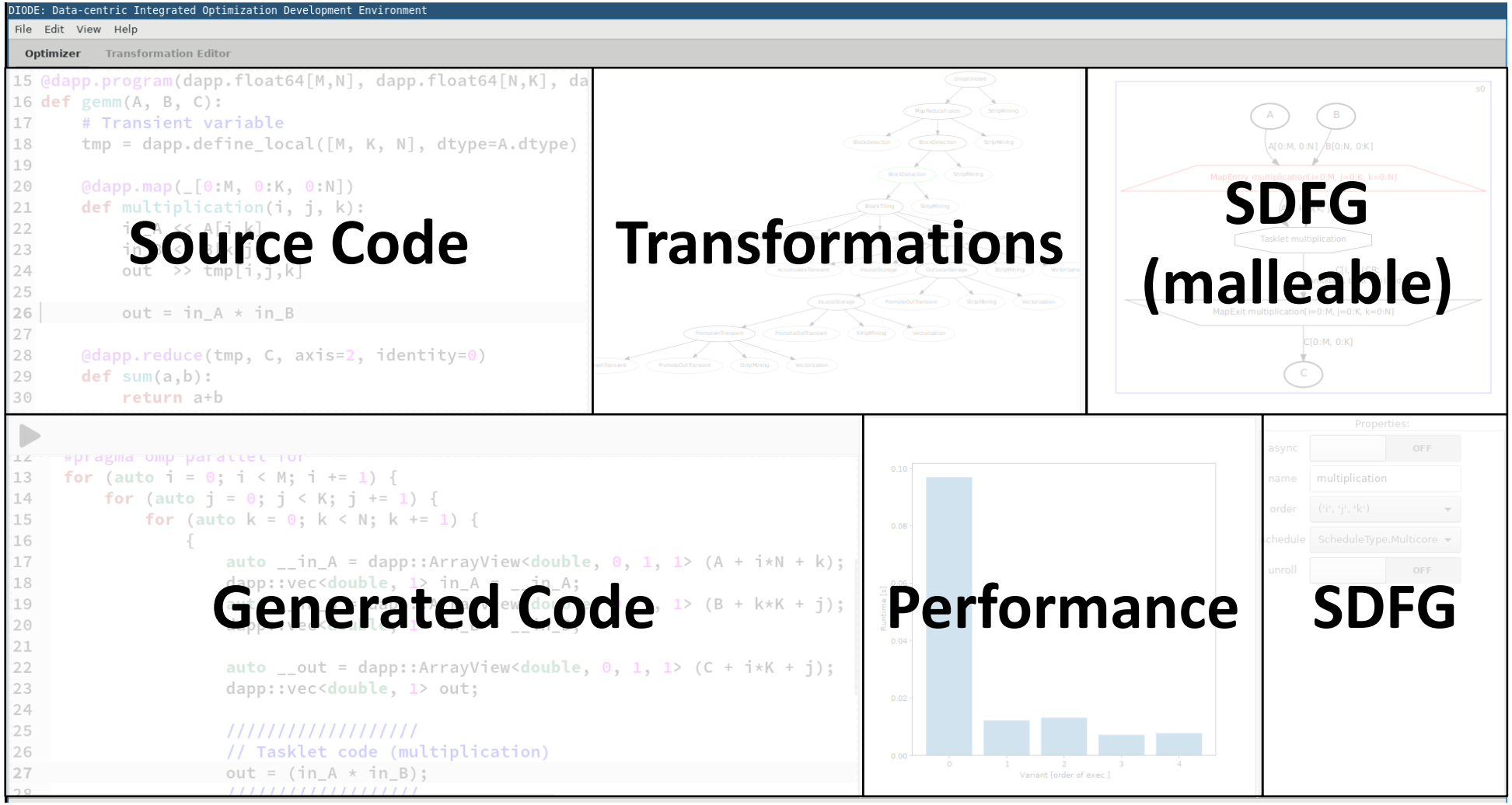


DIODE User Interface



Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

DIODE User Interface



Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Backup

Stateful dataflow graphs – one-minute introduction

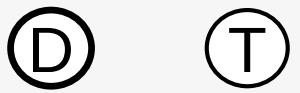
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Stateful dataflow graphs – one-minute introduction

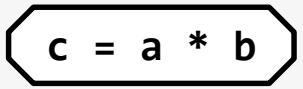
Ⓓ Ⓣ **Data**: Array containers

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Stateful dataflow graphs – one-minute introduction



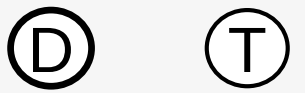
Data: Array containers



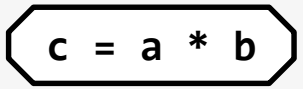
Tasklet: Fine-grained computations

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

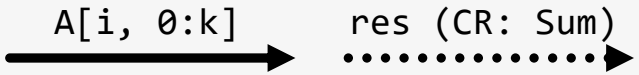
Stateful dataflow graphs – one-minute introduction



Data: Array containers



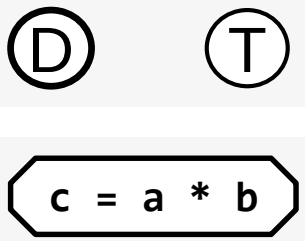
Tasklet: Fine-grained computations



Memlet: Data movement units

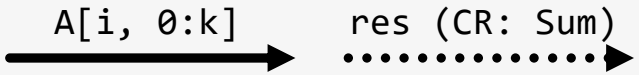
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Stateful dataflow graphs – one-minute introduction

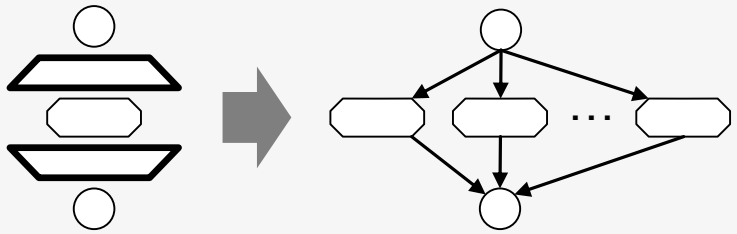


Data: Array containers

Tasklet: Fine-grained computations



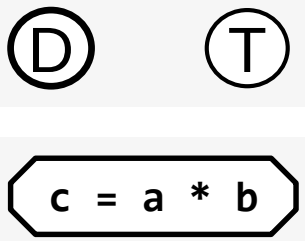
Memlet: Data movement units



Map: Parametric parallelism scopes

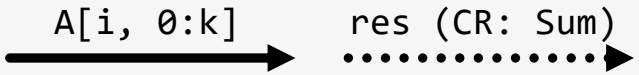
Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Stateful dataflow graphs – one-minute introduction

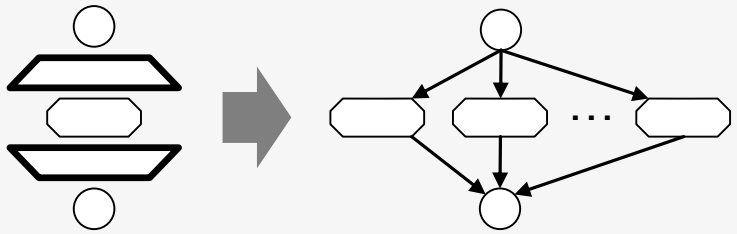


Data: Array containers

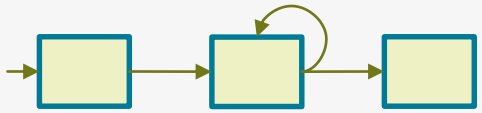
Tasklet: Fine-grained computations



Memlet: Data movement units



Map: Parametric parallelism scopes



State: Control dependencies

Preprint (arXiv): Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs

Optimization Techniques

Optimization Techniques

- **Multi-core CPU**
  - Tiling for complex cache hierarchies
  - Register optimizations
  - Vectorization

Optimization Techniques

- **Multi-core CPU**
  - Tiling for complex cache hierarchies
  - Register optimizations
  - Vectorization
- **Many-core GPU**
  - Coalesced memory access
  - Warp divergence minimization, register tiling
  - Task fusion

Optimization Techniques

- **Multi-core CPU**
  - Tiling for complex cache hierarchies
  - Register optimizations
  - Vectorization
- **Many-core GPU**
  - Coalesced memory access
  - Warp divergence minimization, register tiling
  - Task fusion
- **FPGA**
  - Maximize resource utilization (logic units, DSPs)
  - Streaming optimizations, pipelining
  - Explicit buffering (FIFO) and wiring

Optimization Techniques

- **Multi-core CPU**
  - Tiling for complex cache hierarchies
  - Register optimizations
  - Vectorization
- **Many-core GPU**
  - [illegible]
  - Warp divergence minimization, register tiling
  - Task fusion
- **FPGA**
  - Maximize resource utilization (logic units, DSPs)
  - Streaming optimizations, pipelining
  - Explicit buffering (FIFO) and wiring

High-performance optimization = data movement reduction