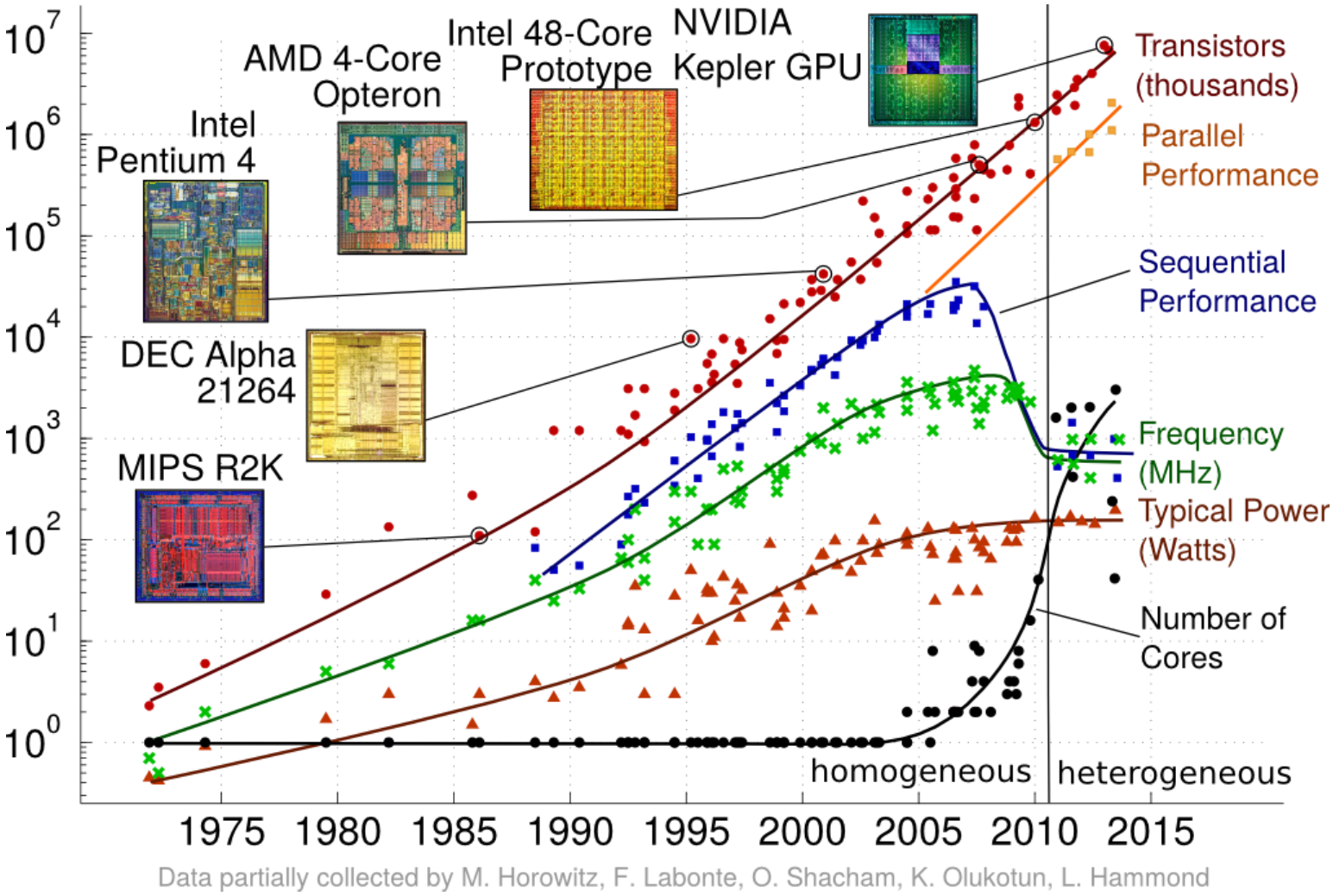
TORSTEN HOEFLER

Progress in automatic GPU compilation and why you want to run MPI on your GPU

with Tobias Grosser and Tobias Gysi @ SPCL
presented at High Performance Computing, Cetraro, Italy 2016

Evading various "ends" – the hardware view

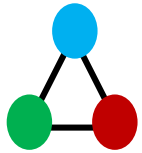


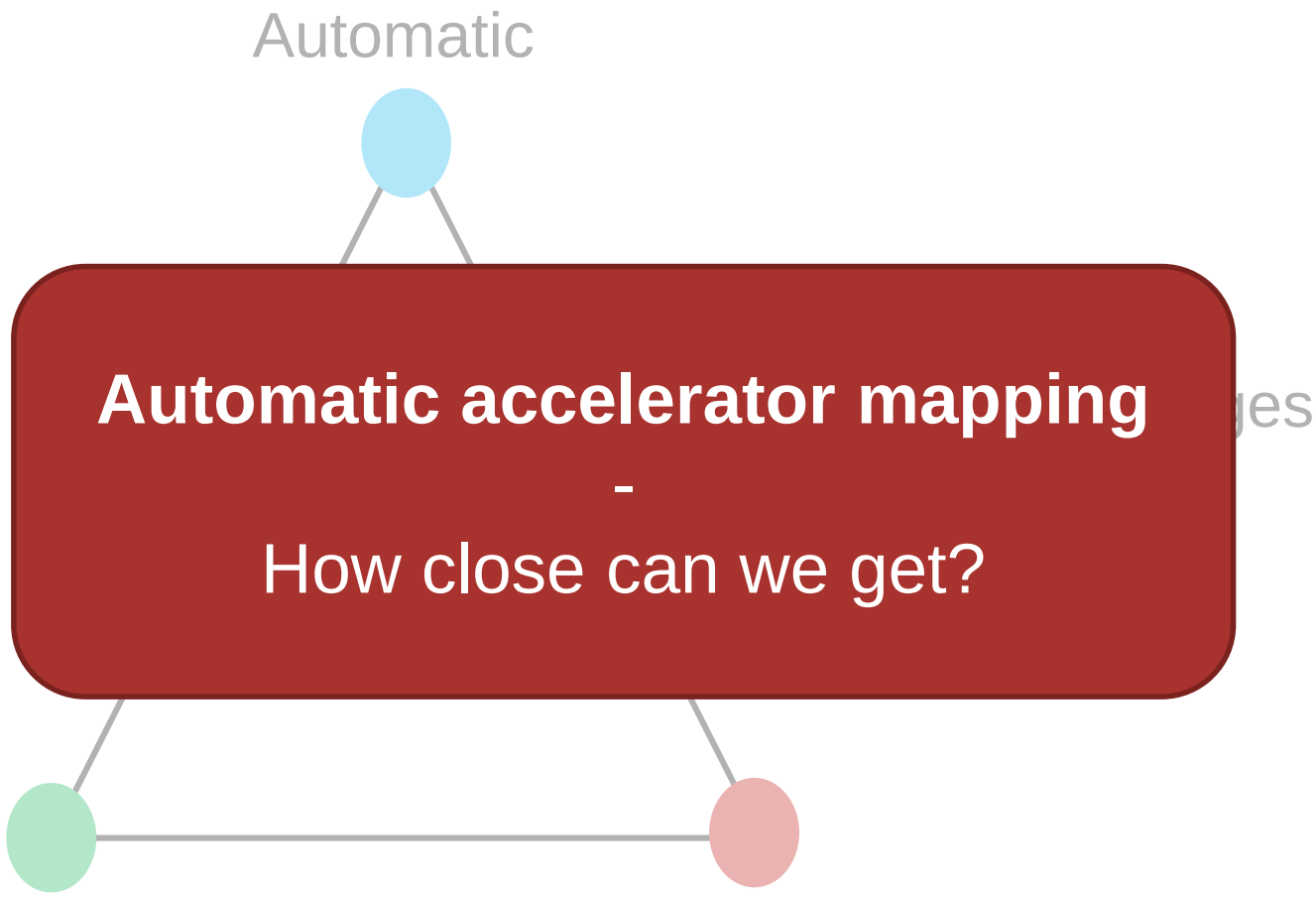
Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

Pete's system software view

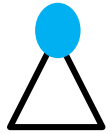
My software/programming model view

Holy grail – auto-~~parallelization~~ heterogenization





Tool: Polyhedral Modeling



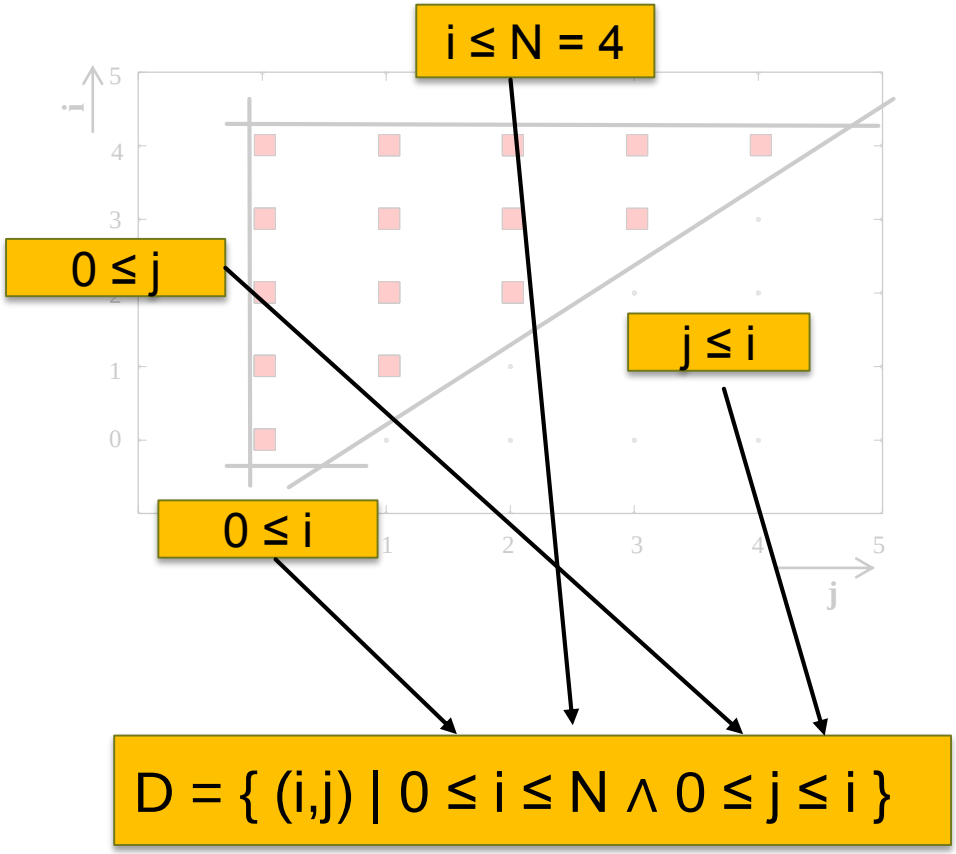
Program Code

```
for (i = 0; i <= N; i++)
  for (j = 0; j <= i; j++)
    S(i,j);
```

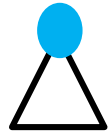
N = 4

(i, j) = (4,4)

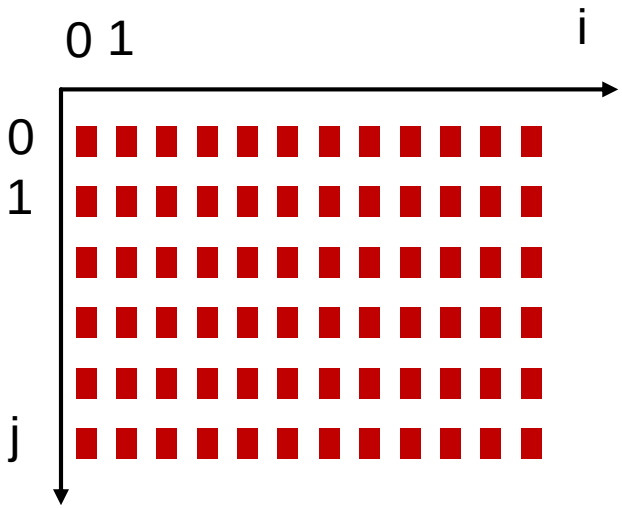
Iteration Space



Mapping Computation to Device



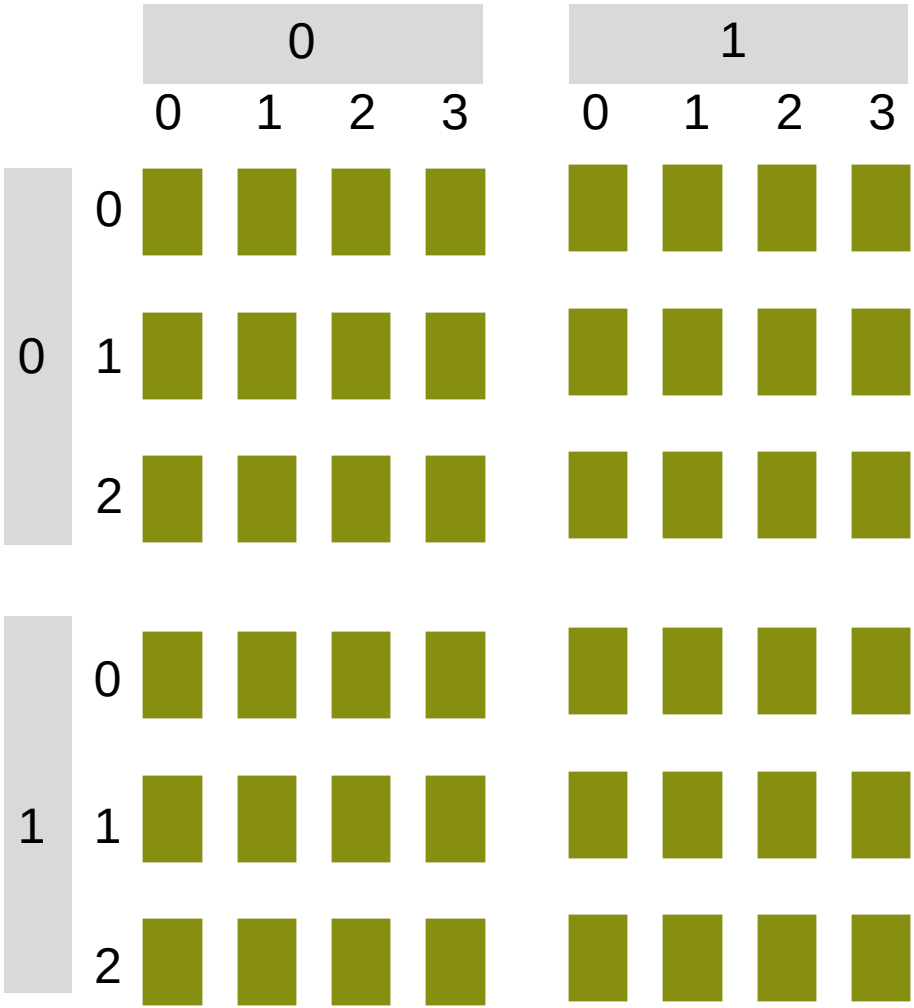
Iteration Space



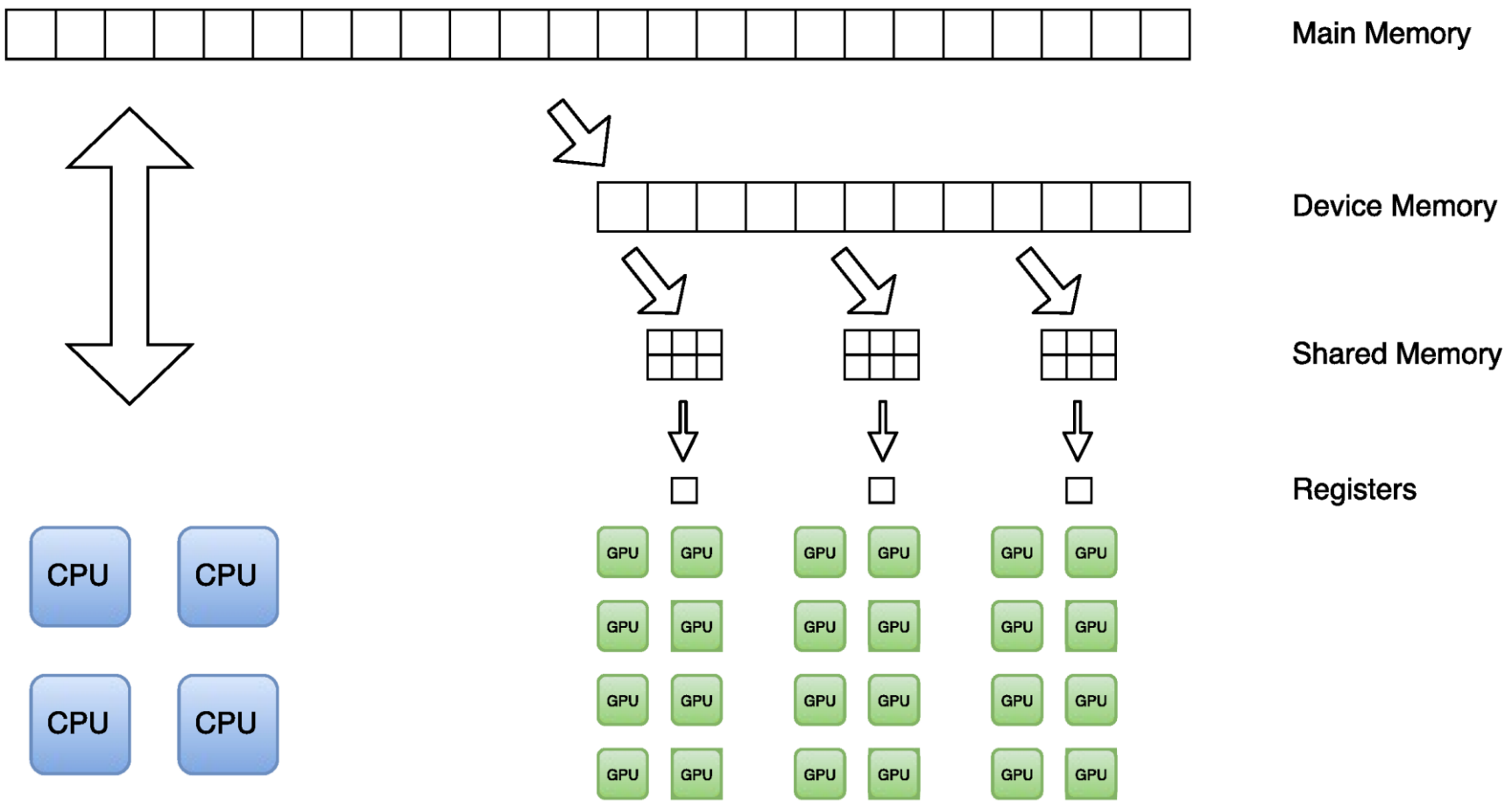
$$BID = \{(i,j) \rightarrow \left(\left\lfloor \frac{i}{4} \right\rfloor \% 2, \left\lfloor \frac{j}{3} \right\rfloor \% 2\right)\}$$

$$TID = \{(i,j) \rightarrow (i \% 4, j \% 3)\}$$

Device Blocks & Threads

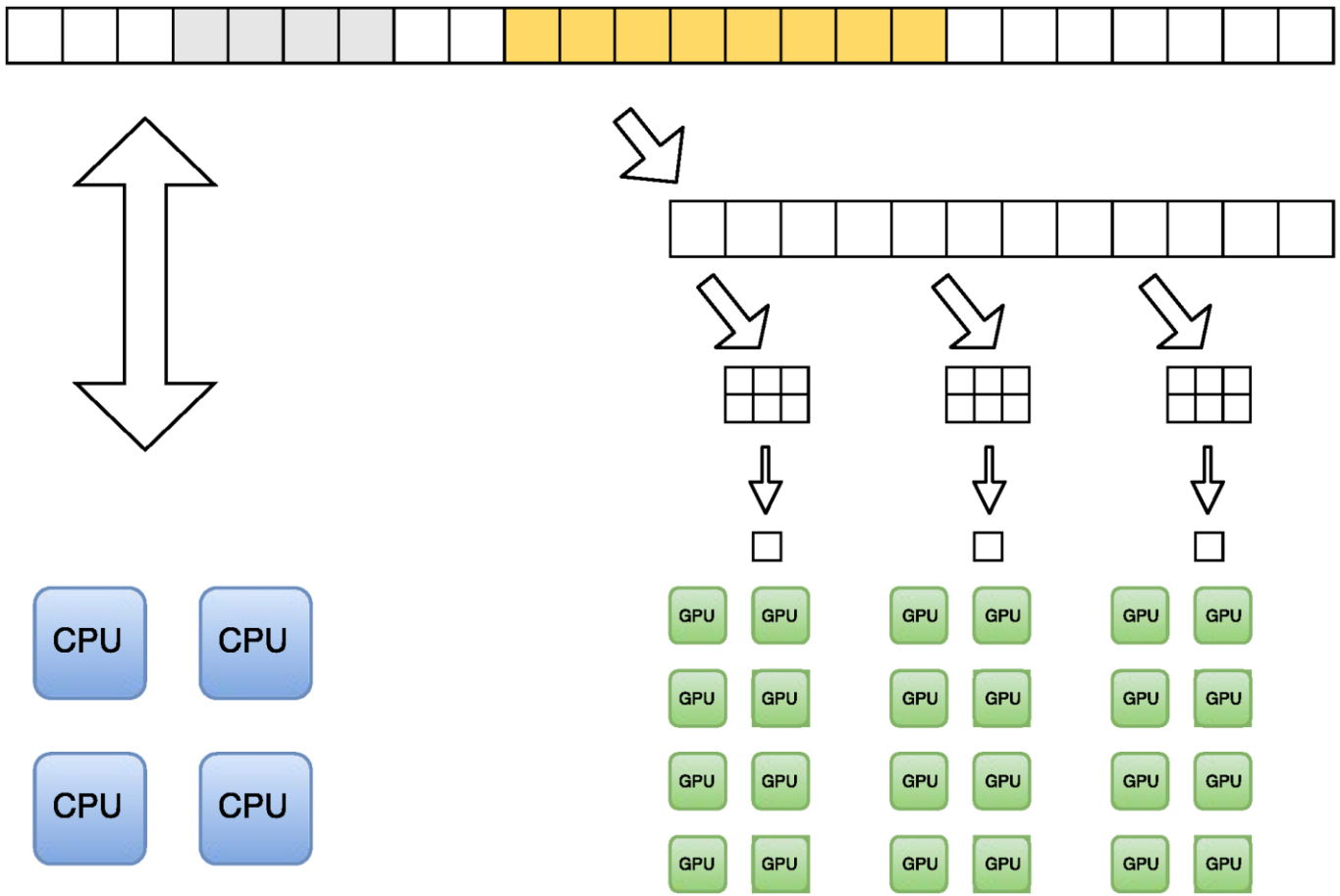


Memory Hierarchy of a Heterogeneous System



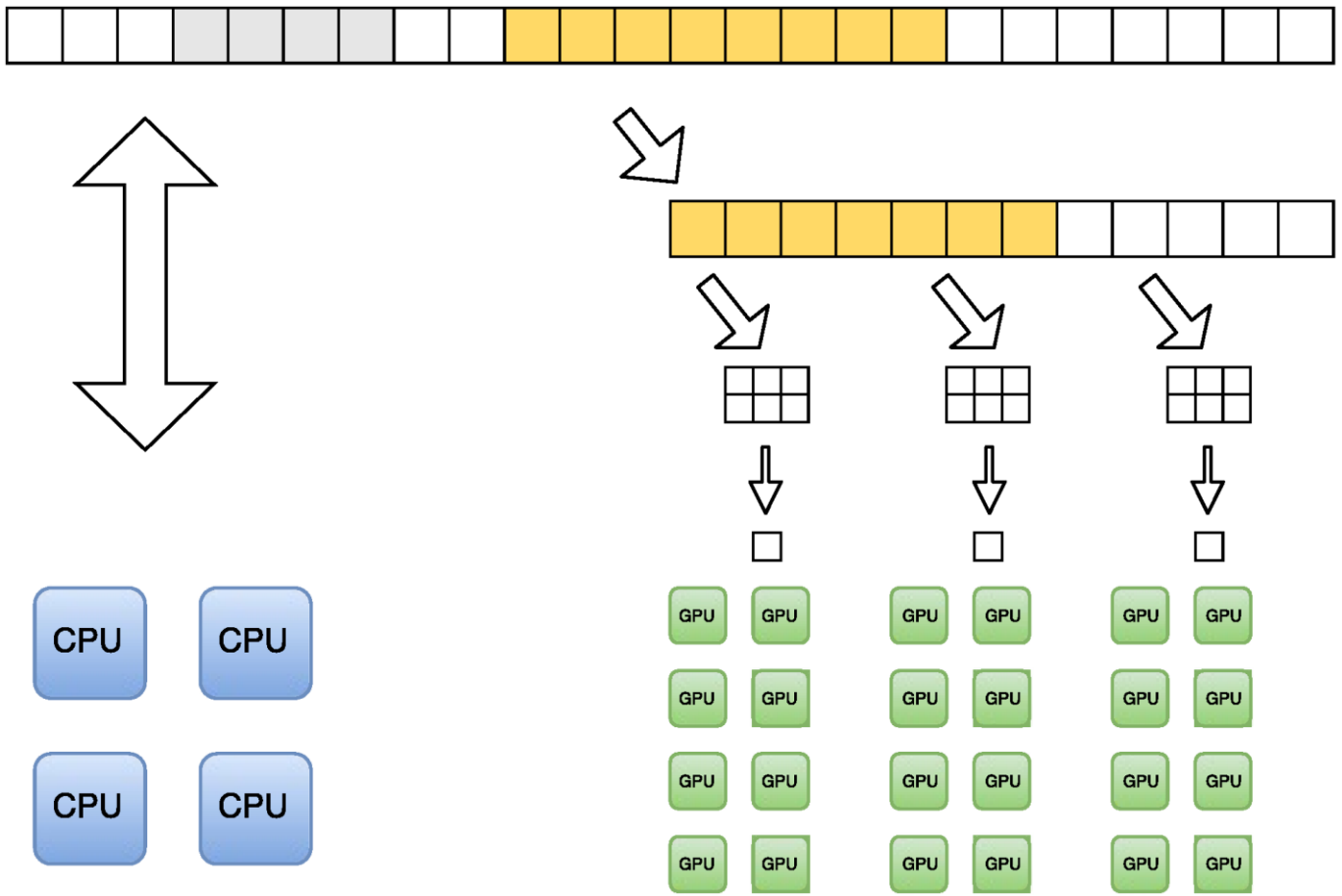
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

Host-device date transfers



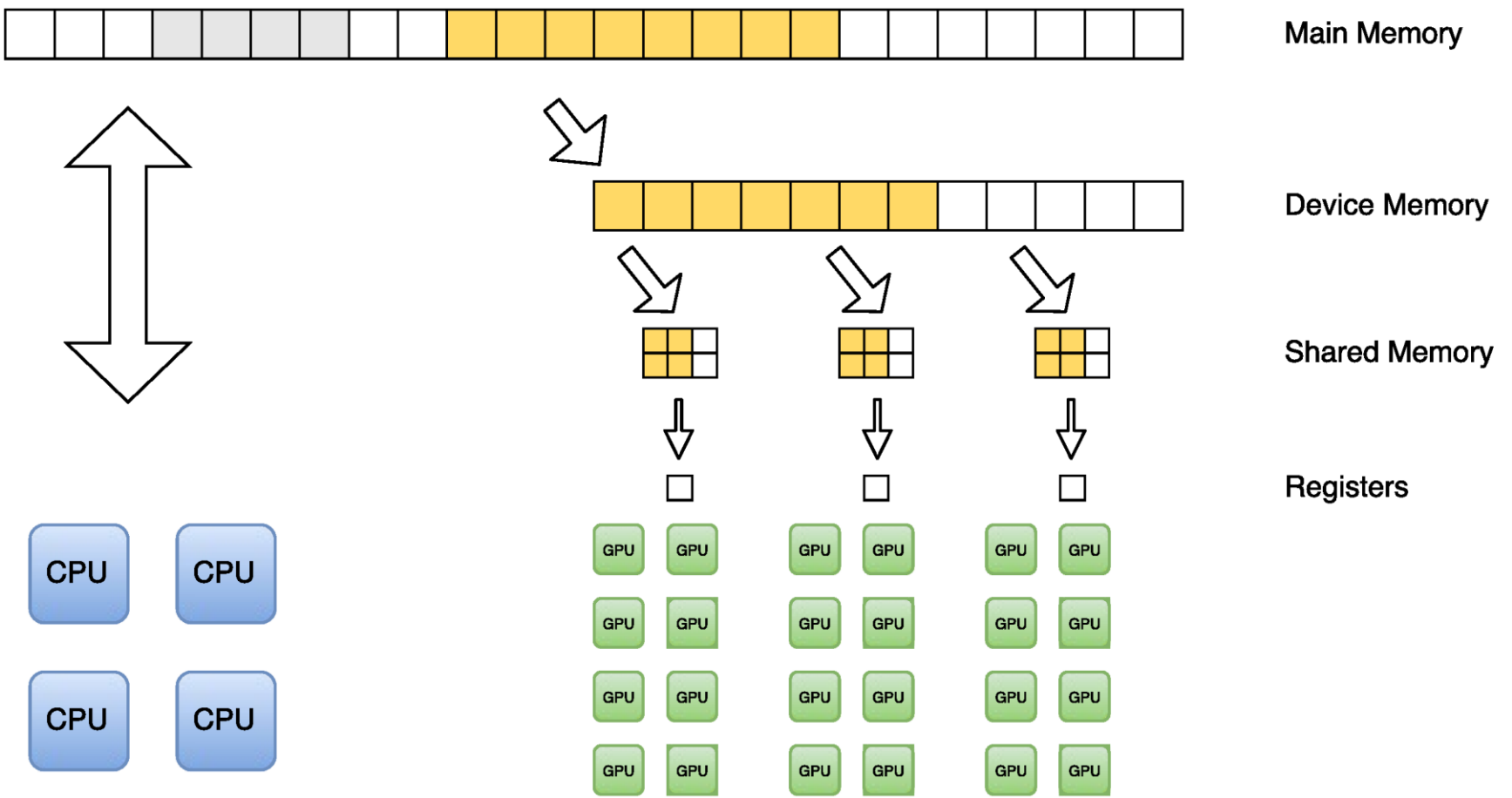
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

Host-device date transfers



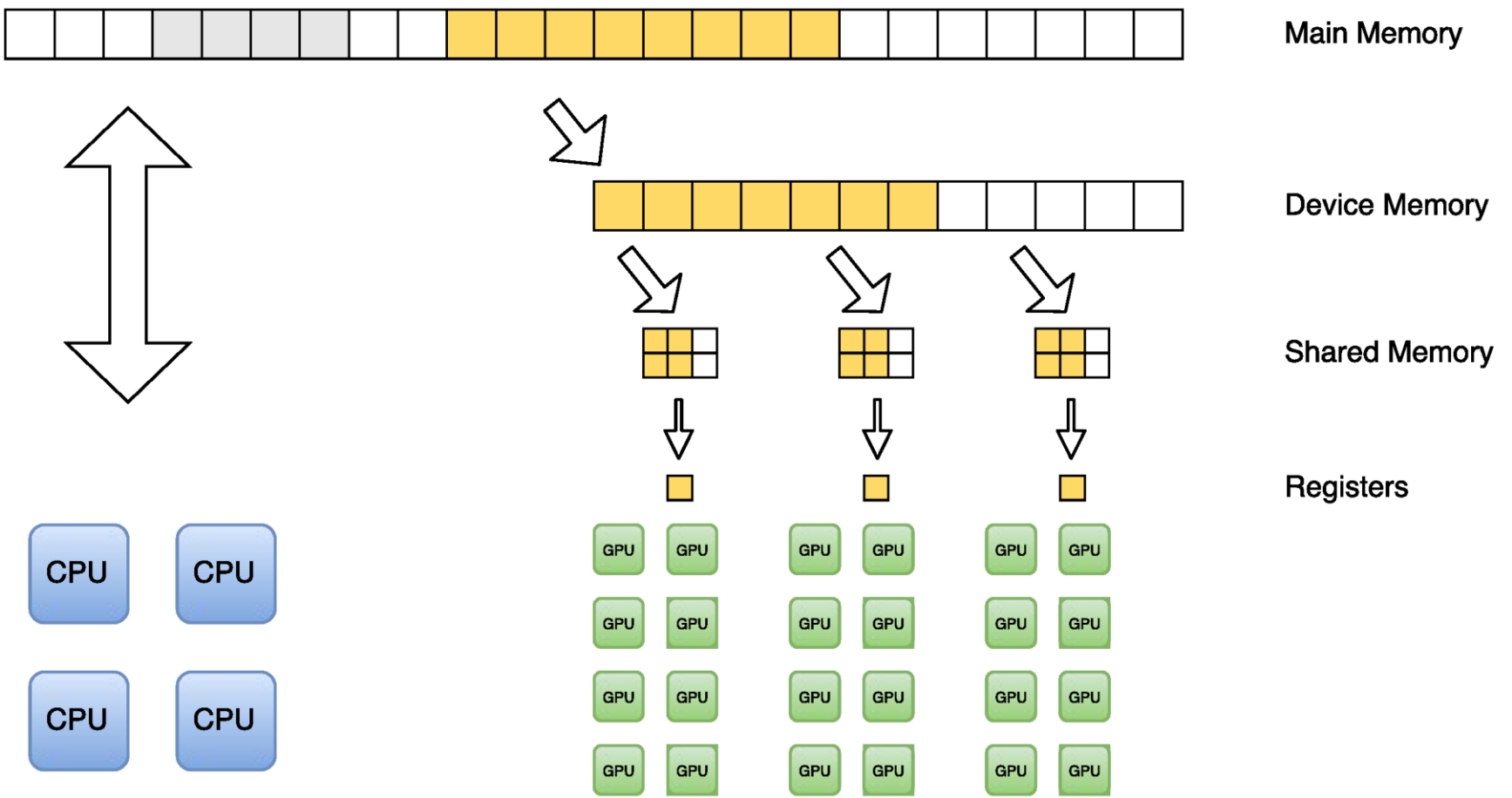
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

Mapping onto fast memory



T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

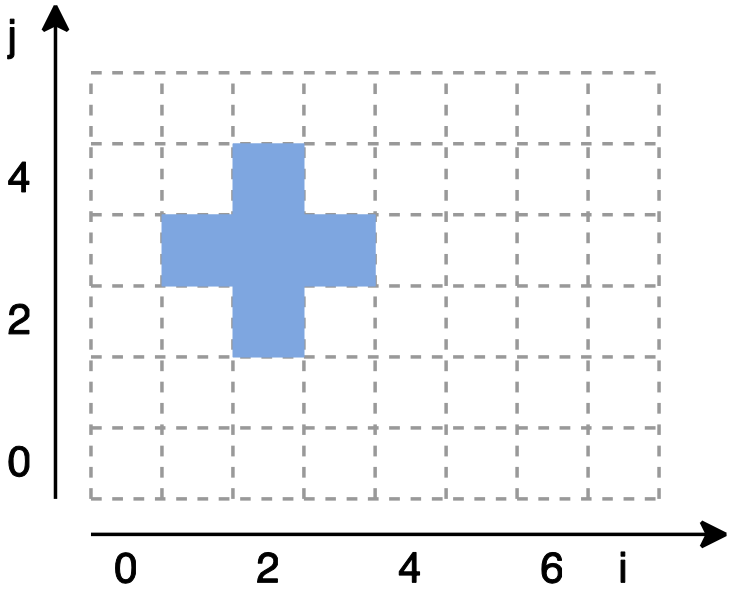
Mapping onto fast memory



T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

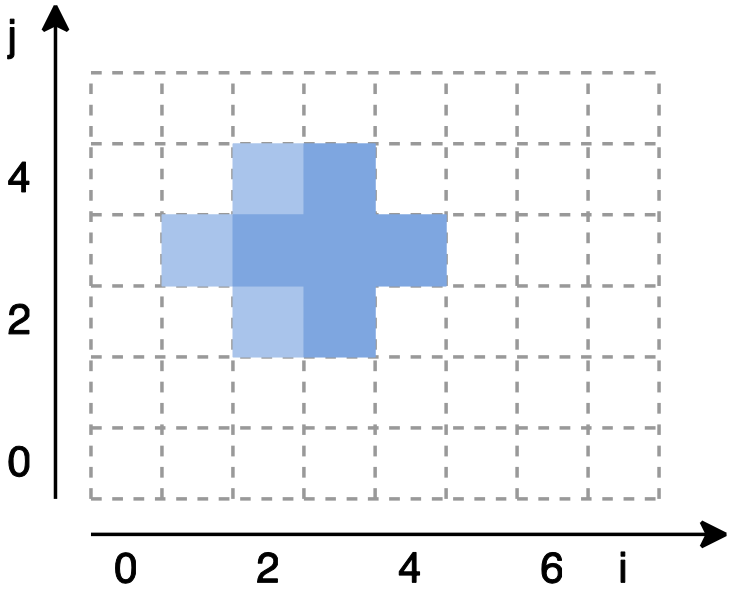
Accessed Data (for a 2x2 thread block)

```
for (i = 1; i <= 6; i++)
  for (j = 1; j <= 4; j++)
    … = A[i+1][j] + A[i-1][j] + A[i][j] + A[i][j-1] + A[i][j+1];
```



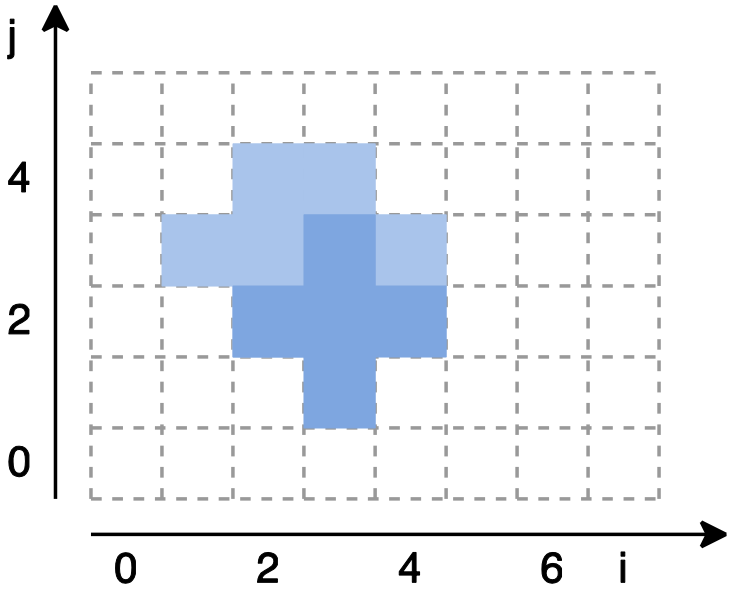
Accessed Data (for a 2x2 thread block)

```
for (i = 1; i <= 6; i++)
  for (j = 1; j <= 4; j++)
    … = A[i+1][j] + A[i-1][j] + A[i][j] + A[i][j-1] + A[i][j+1];
```



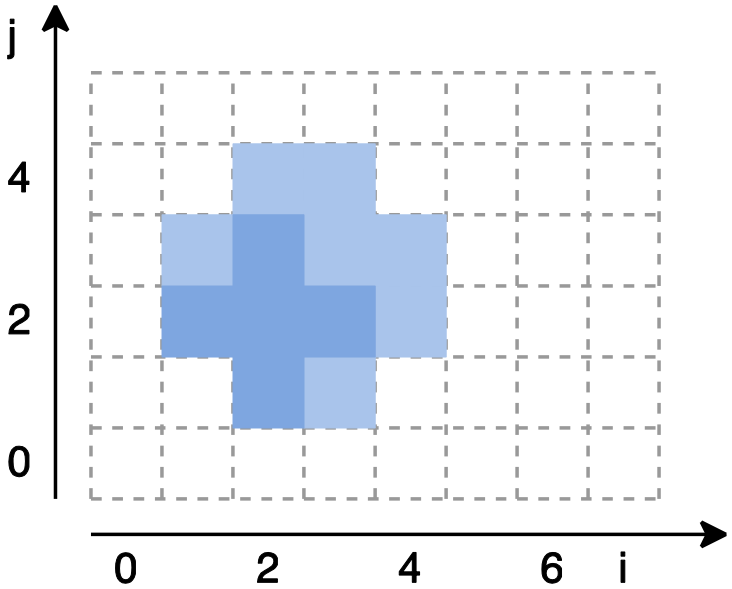
Accessed Data (for a 2x2 thread block)


```
for (i = 1; i <= 6; i++)
  for (j = 1; j <= 4; j++)
    … = A[i+1][j] + A[i-1][j] + A[i][j] + A[i][j-1] + A[i][j+1];
```



Accessed Data (for a 2x2 thread block)

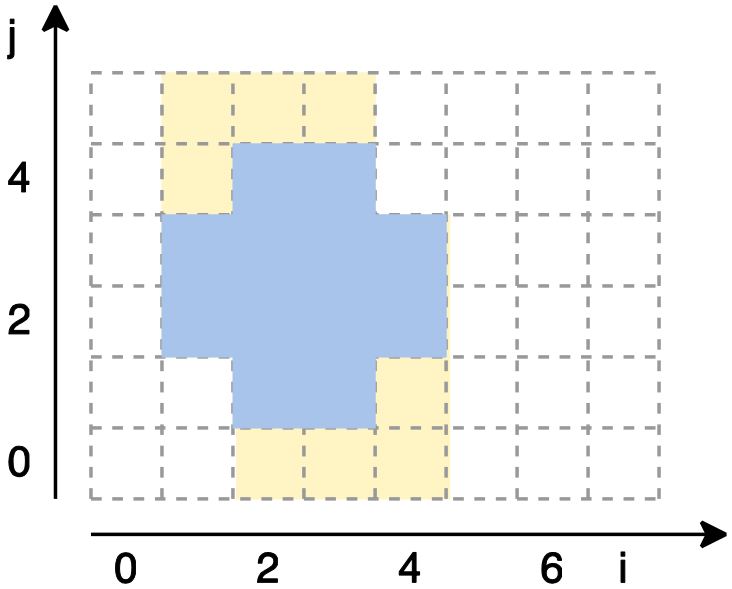
```
for (i = 1; i <= 6; i++)
  for (j = 1; j <= 4; j++)
    … = A[i+1][j] + A[i-1][j] + A[i][j] + A[i][j-1] + A[i][j+1];
```



- Data needed on device
- 12 elements
- Minimal data, but complex transfer

Accessed Data (for a 2x2 thread block)

```
for (i = 1; i <= 6; i++)
  for (j = 1; j <= 4; j++)
    … = A[i+1][j] + A[i-1][j] + A[i][j] + A[i][j-1] + A[i][j+1];
```

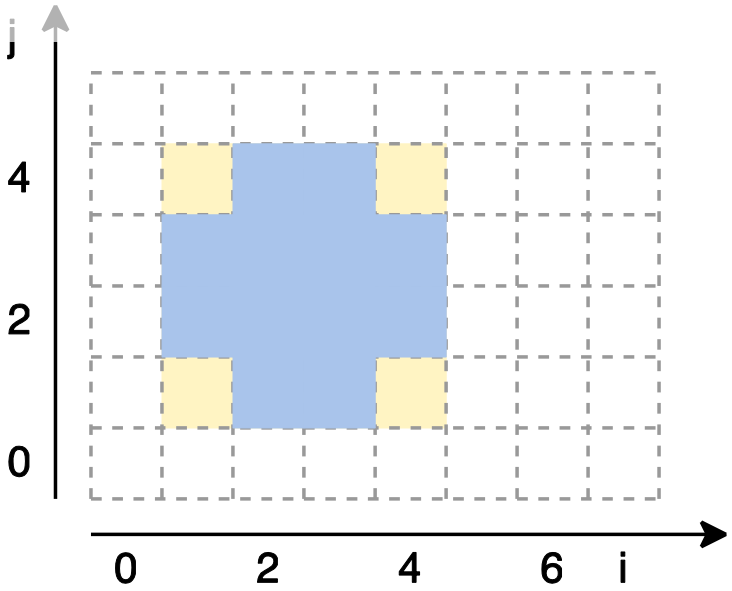


- One-dimensional hull
- 20 elements
- Simple transfer, but redundant data

Accessed Data (for a 2x2 thread block)

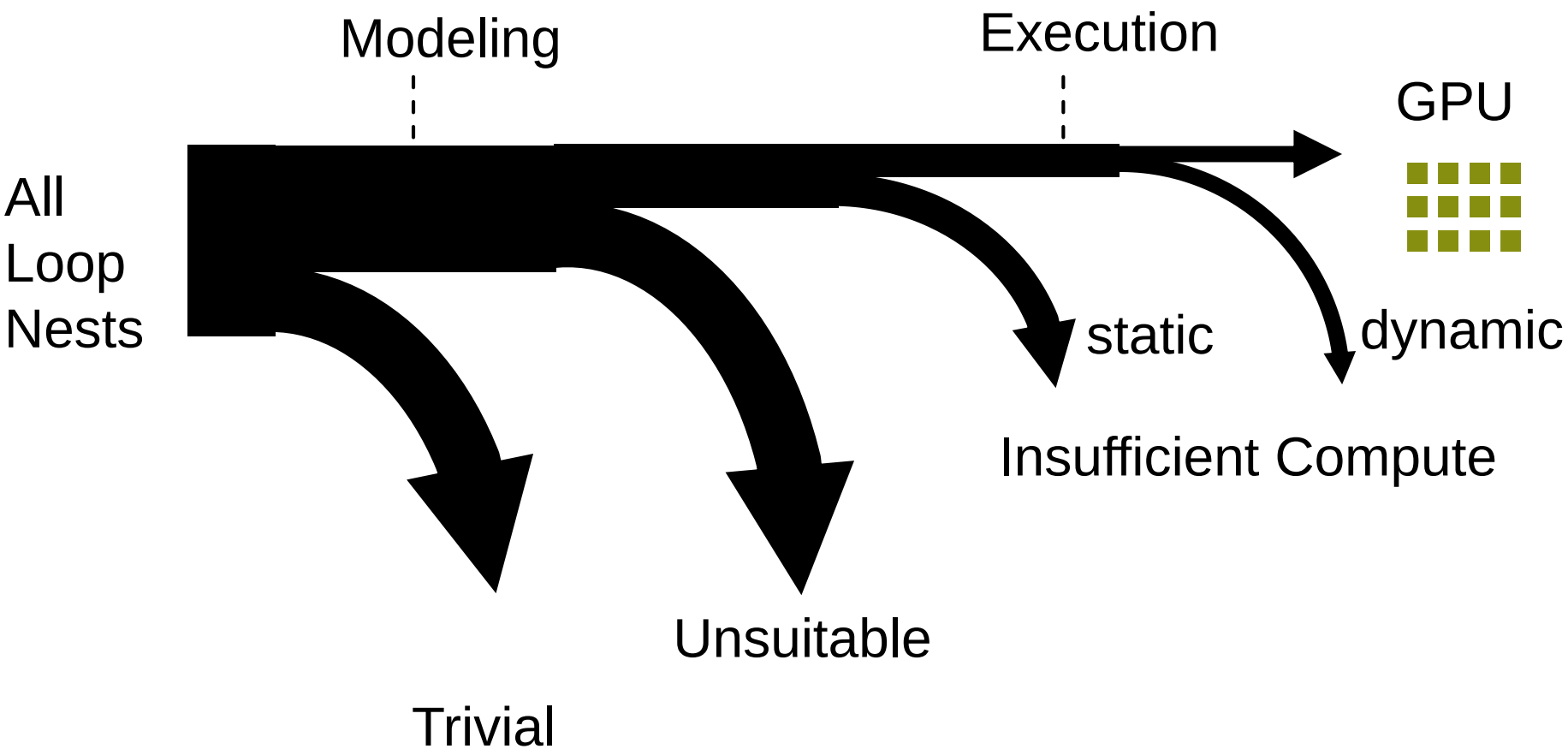
```
for (i = 1; i <= 6; i++)
  for (
    ...
```

Modeling multi-dimensional access behavior is important



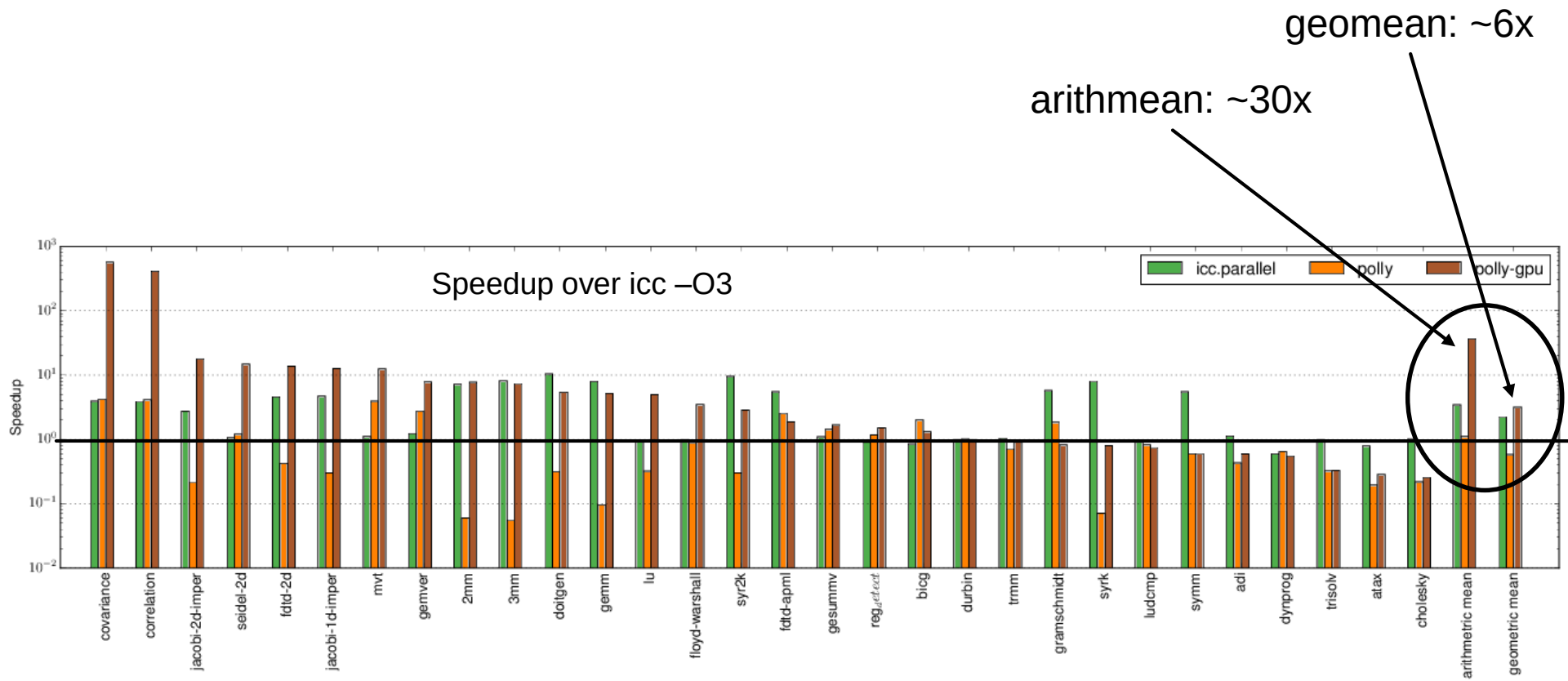
- Two-dimensional hull
- 16 elements
- Simple transfer, less redundant data

Profitability Heuristic



T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

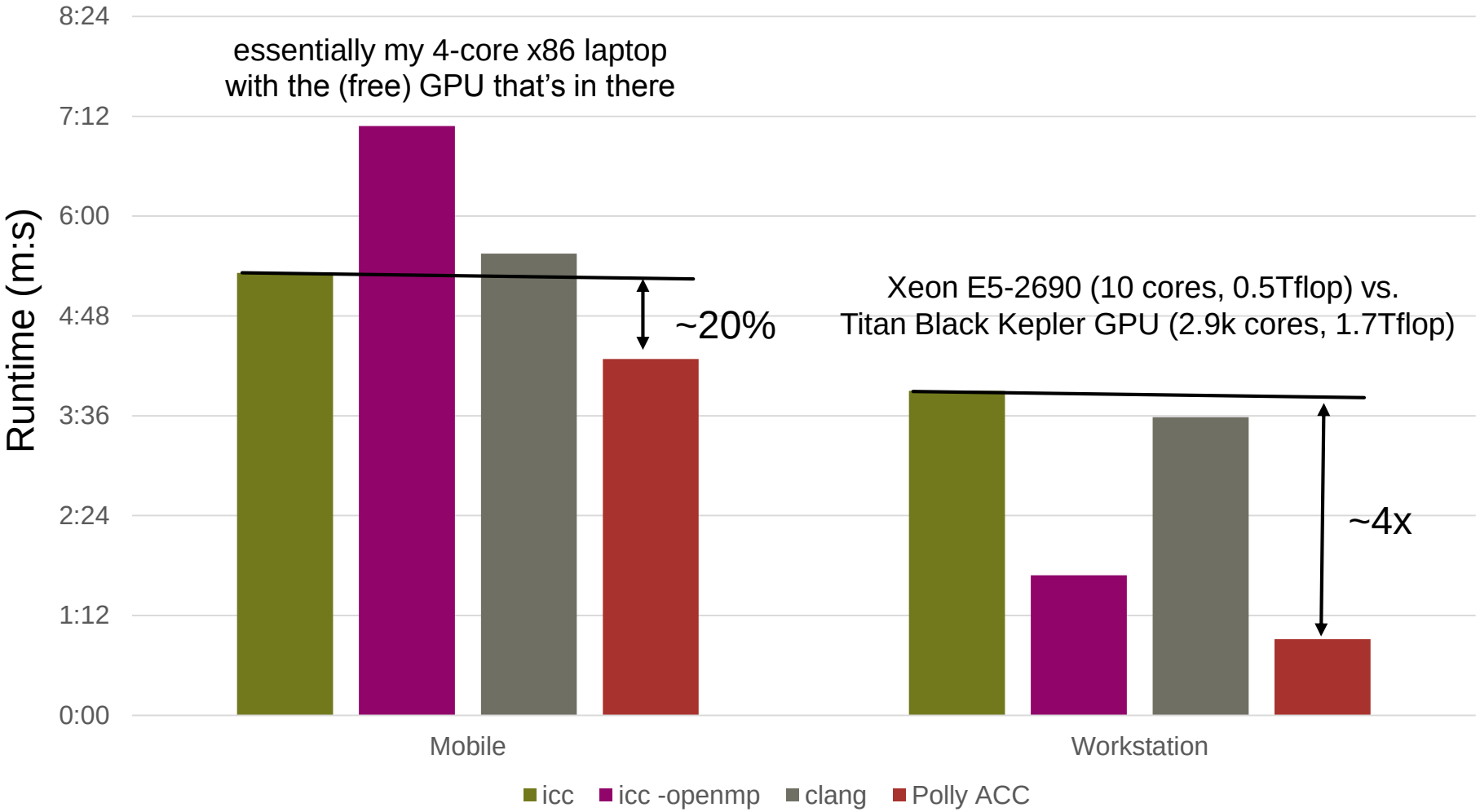
Some results: Polybench 3.2



Xeon E5-2690 (10 cores, 0.5Tflop) vs. Titan Black Kepler GPU (2.9k cores, 1.7Tflop)

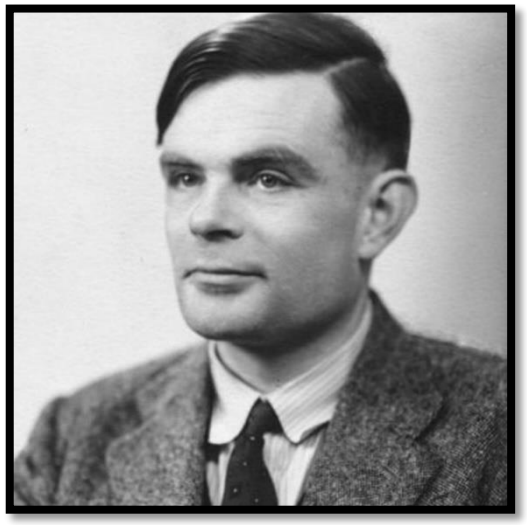
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

Compiles all of SPEC CPU 2006 – Example: LBM



T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

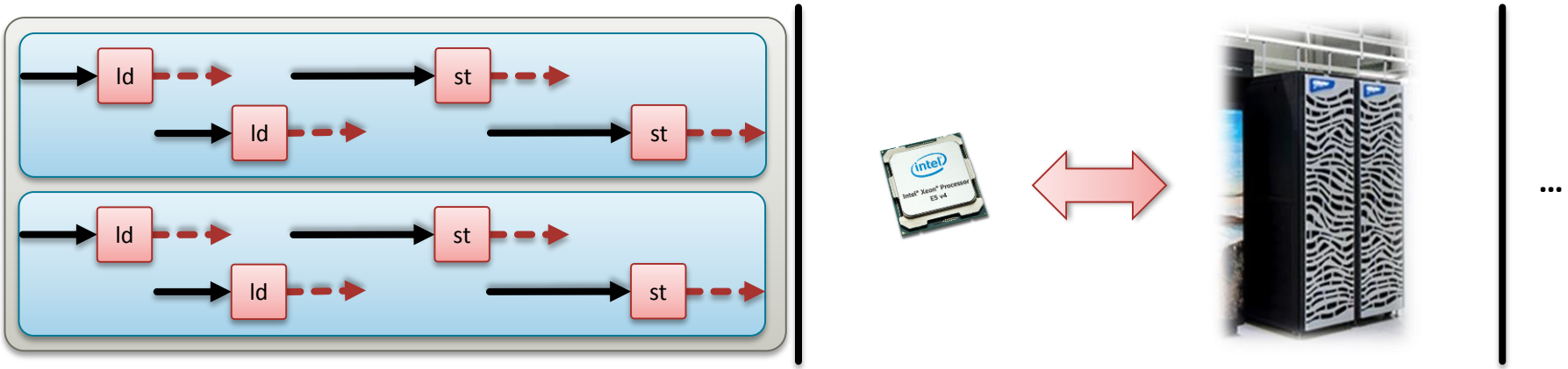
Brave new compiler world!?



- **Unfortunately not …**
  - Limited to affine code regions
  - Maybe generalizes to control-restricted programs
  - No distributed anything!!
- **Good news:**
  - Much of traditional HPC fits that model
  - Infrastructure is coming along

- **Bad news:**
  - Modern data-driven HPC and Big Data fits less well
  - Need a programming model for **distributed** heterogeneous machines!

How do we program GPUs today?

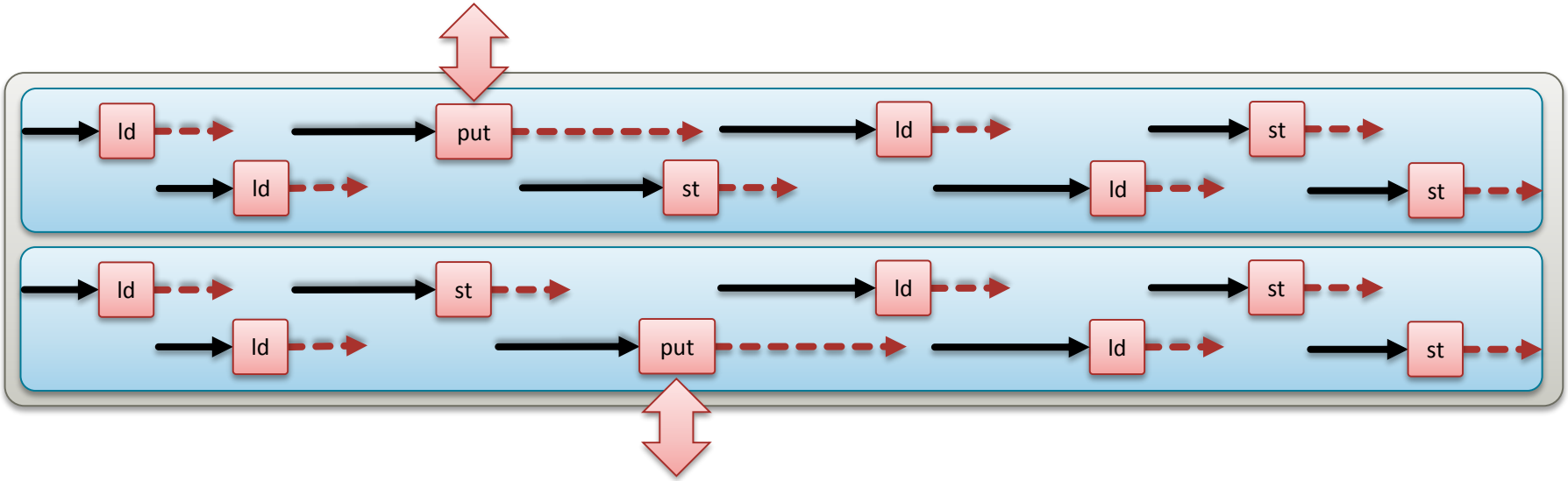


CUDA
- over-subscribe hardware
- use spare parallel slack for latency hiding

MPI
- host controlled
- full device synchronization

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

Latency hiding at the cluster level?



dCUDA (distributed CUDA)

- unified programming model for GPU clusters
- avoid unnecessary device synchronization to enable system wide latency hiding

device | compute core | active thread | instruction latency

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

dCUDA extends CUDA with MPI-3 RMA and notifications

```
for (int i = 0; i < steps; ++i) {
  for (int idx = from; idx < to; idx += jstride)
    out[idx] = -4.0 * in[idx] +
      in[idx + 1] + in[idx - 1] +
      in[idx + jstride] + in[idx - jstride];

  if (lsend)
    dcuda_put_notify(ctx, wout, rank - 1,
      len + jstride, jstride, &out[jstride], tag);
  if (rsend)
    dcuda_put_notify(ctx, wout, rank + 1,
      0, jstride, &out[len], tag);

  dcuda_wait_notifications(ctx, wout,
    DCUDA_ANY_SOURCE, tag, lsend + rsend);

  swap(in, out); swap(win, wout);
}
```

computation

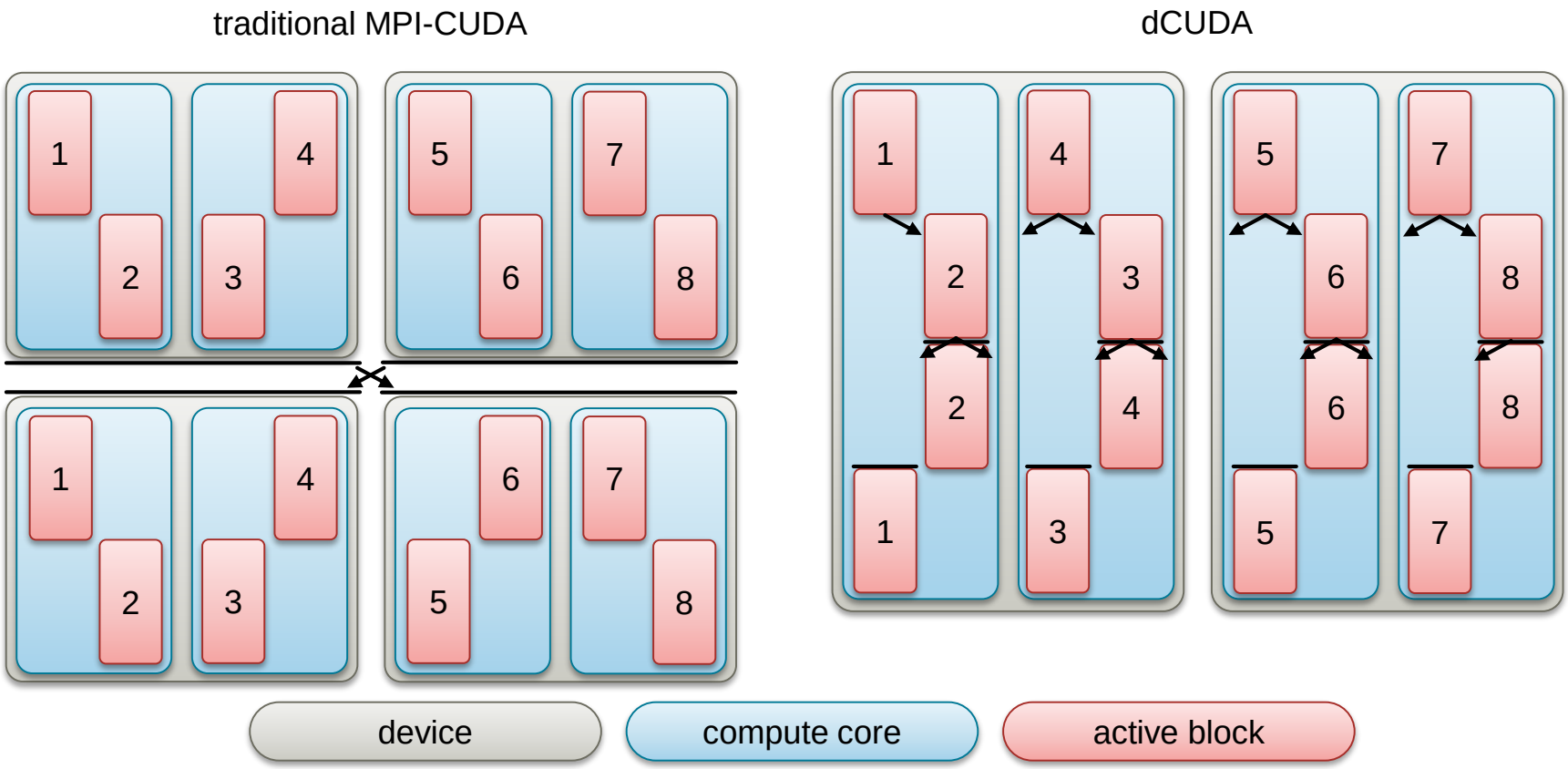
communication

- iterative stencil kernel
- thread specific idx

- map ranks to blocks
- device-side put/get operations
- notifications for synchronization
- shared and distributed memory

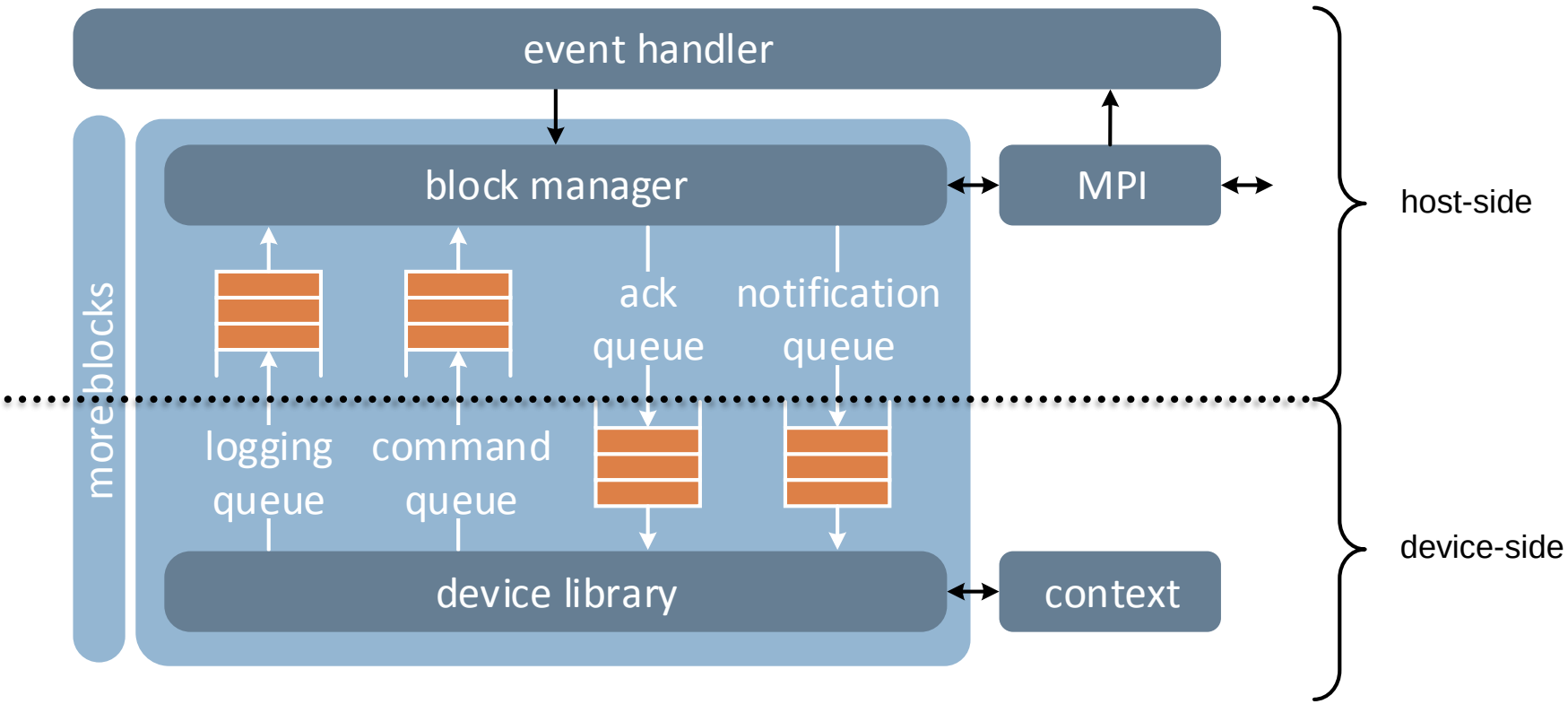
T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

Hardware supported communication overlap



T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

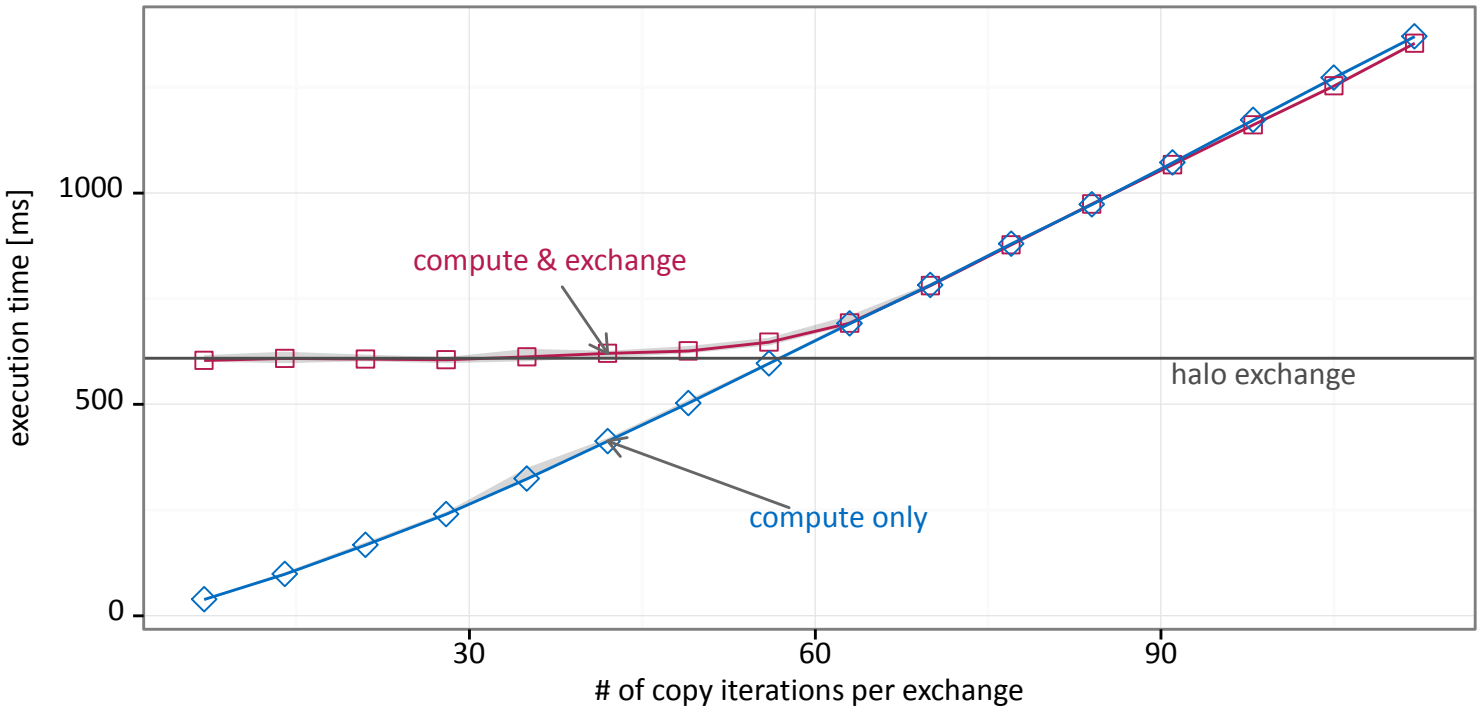
Implementation of the dCUDA runtime system



T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

Overlap of a copy kernel with halo exchange communication

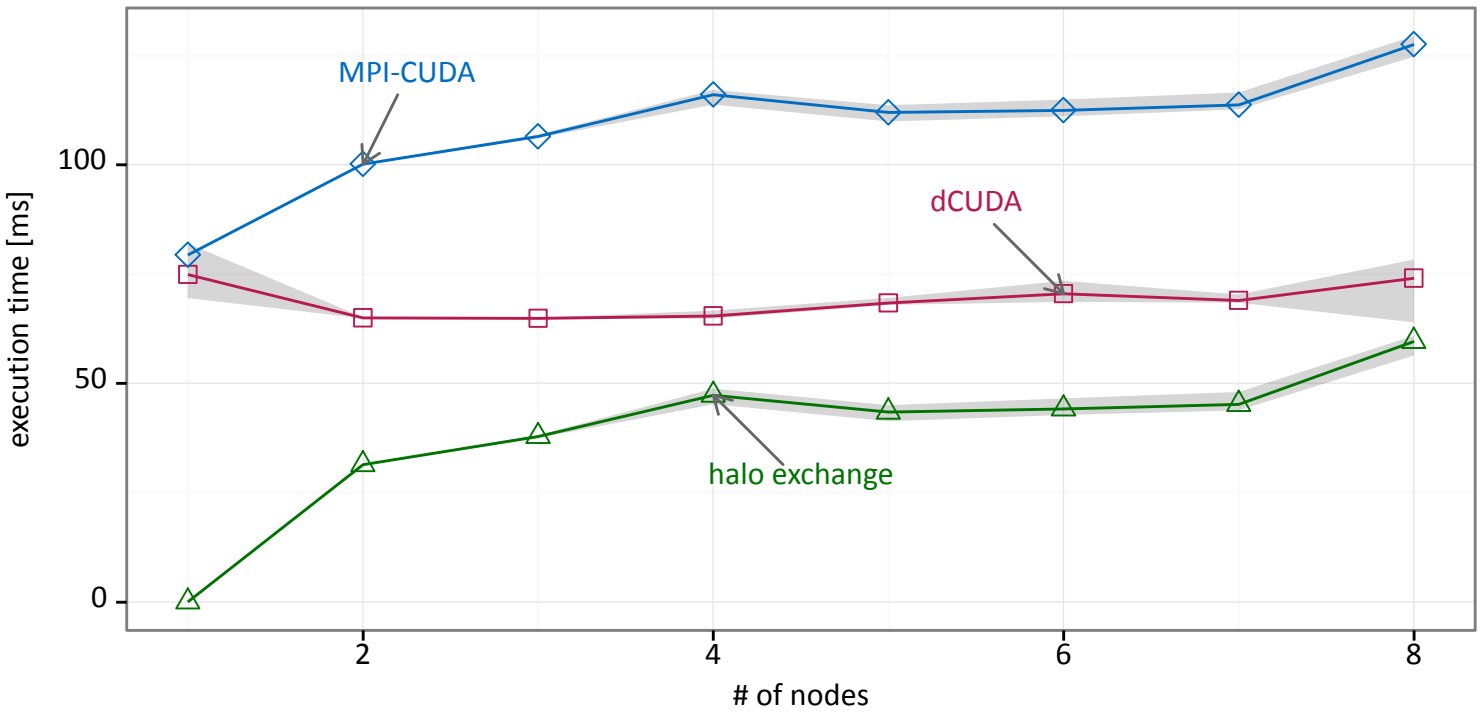
- **Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)**



T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

Weak scaling of MPI-CUDA and dCUDA for a stencil program

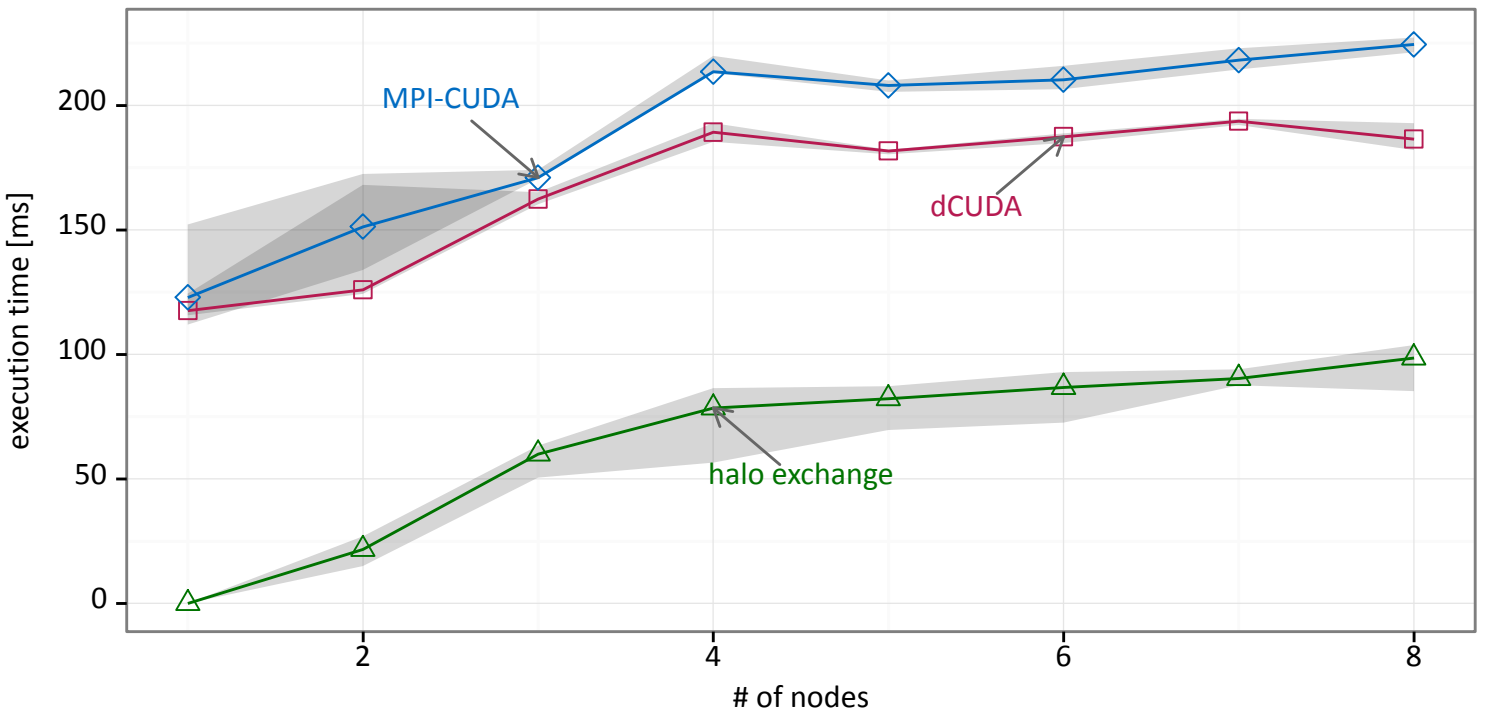
- **Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)**



T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

Weak scaling of MPI-CUDA and dCUDA for a particle simulation

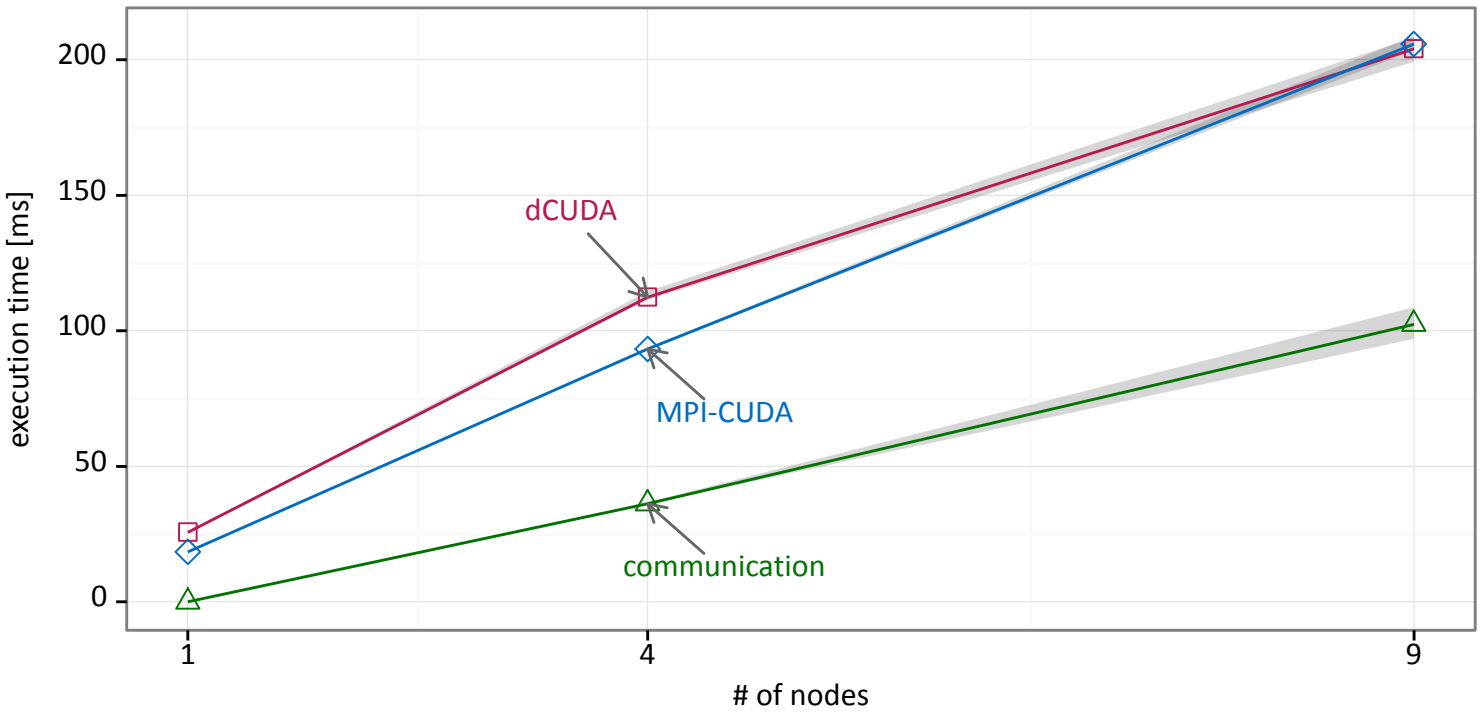
- **Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)**



T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

Weak scaling of MPI-CUDA and dCUDA for sparse-matrix vector multiplication

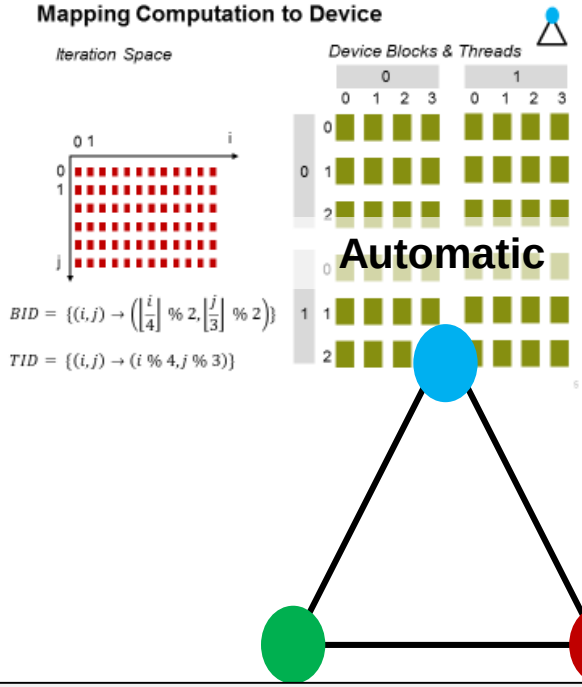
- **Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)**

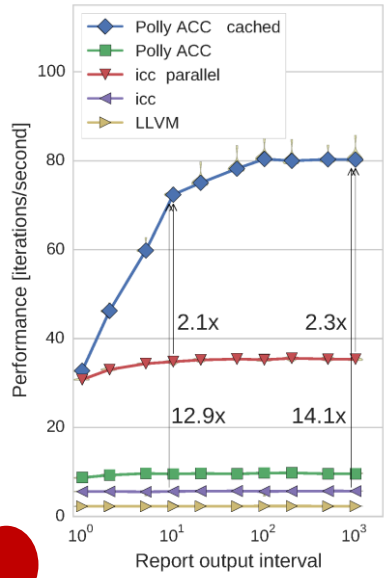


T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)

http://spcl.inf.ethz.ch/Polly-ACC

Platform for Advanced Scientific Computing





dCUDA – distributed memory

```
for (int i = 0; i < steps; ++i) {
  for (int idx = from; idx < to; idx += jstride)
    out[idx] = -4.0 * in[idx] +
      in[idx + 1] + in[idx - 1] +
      in[idx + jstride] + in[idx - jstride];

  if (lsend)
    dcuda_put_notify(ctx, wout, rank - 1,
      len + jstride, jstride, &out[jstride], tag);
  if (rsend)
    dcuda_put_notify(ctx, wout, rank + 1,
      0, jstride, &out[len], tag);

  dcuda_wait_notifications(ctx, wout,
    DCUDA_ANY_SOURCE, tag, lsend + rsend);

  swap(in, out); swap(win, wout);
}
```

Automatic