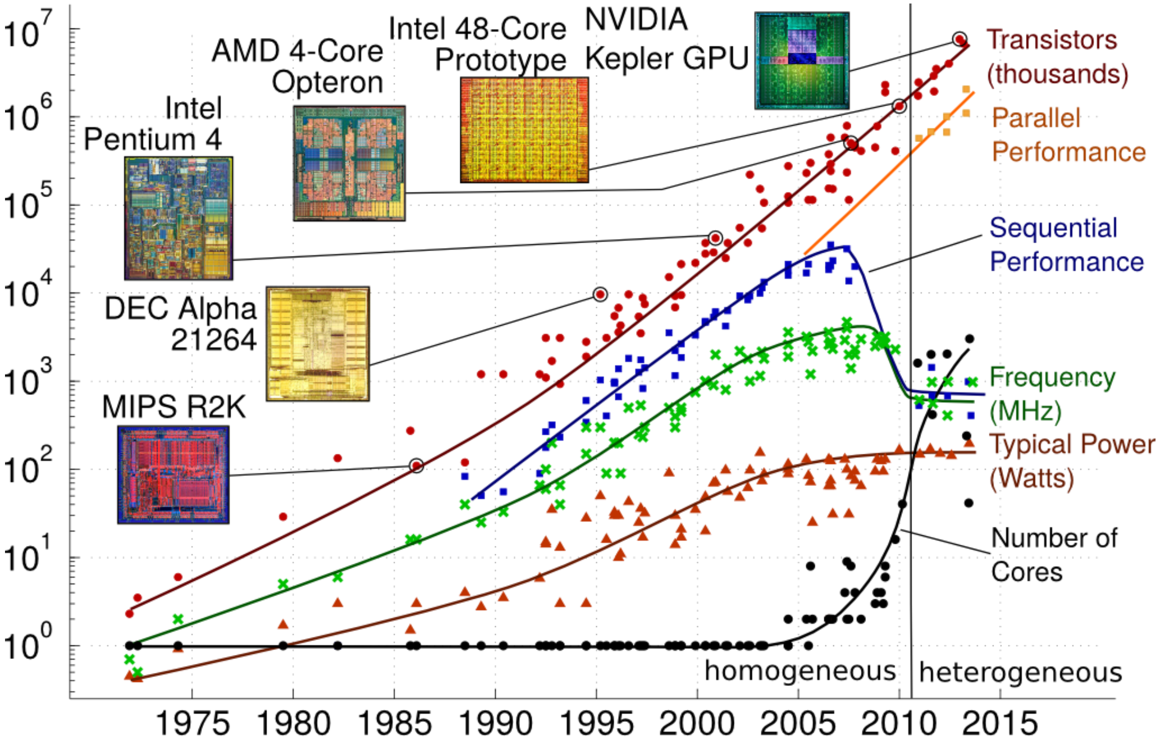
Polly-ACC: Transparent Compilation to Heterogeneous Hardware

Torsten Hoefler (with Tobias Grosser)

Evading various "ends" – the hardware view



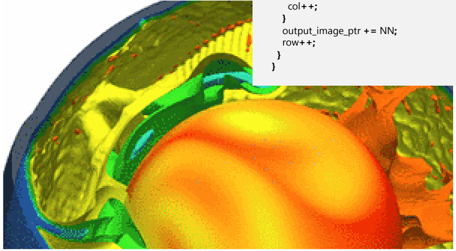
Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

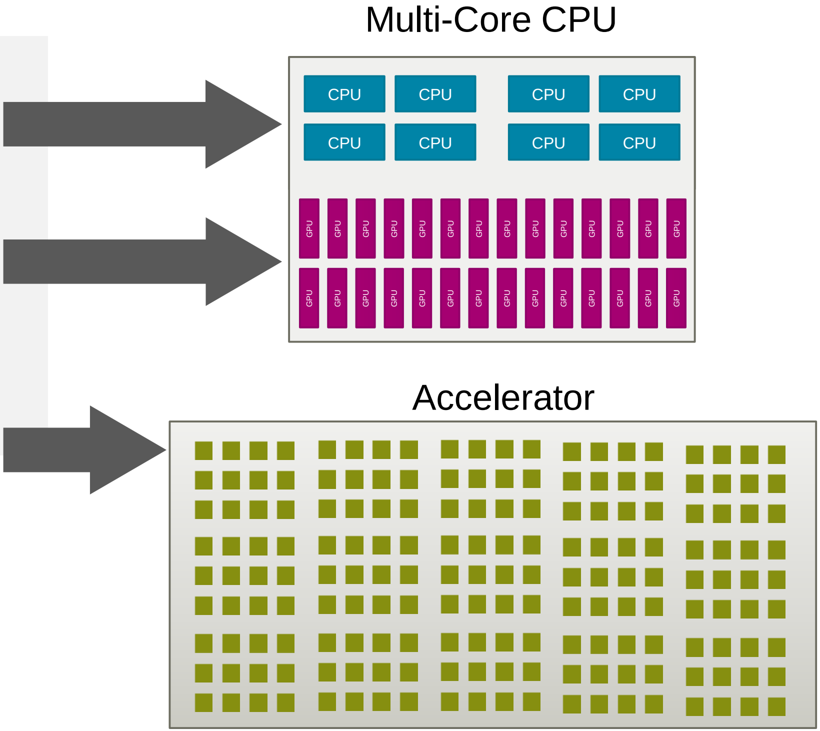
Sequential Software

Parallel Hardware

Fortran
C/C++

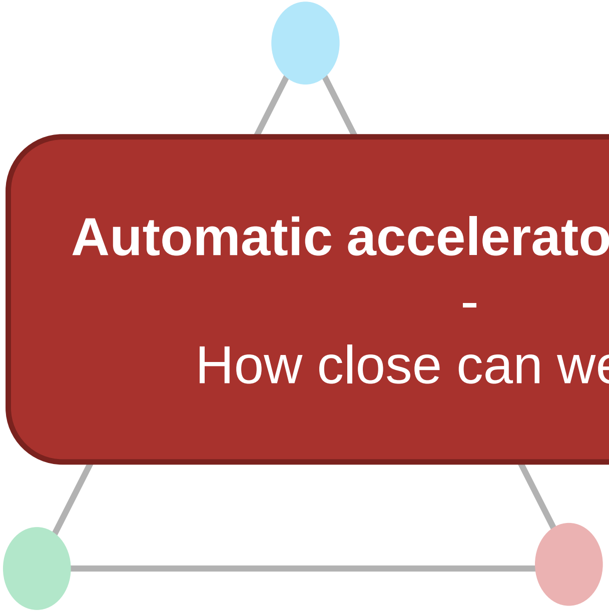
```
  row = 0;
  output_image_ptr = output_image;
  output_image_ptr += (NN * dead_rows);
  for (r = 0; r < NN - KK + 1; r++) {
   output_image_offset = output_image_ptr;
   output_image_offset += dead_cols;
   col = 0;
   for (c = 0; c < NN - KK + 1; c++) {
     input_image_ptr = input_image;
     input_image_ptr += (NN * row);
     kernel_ptr = kernel;
S0: *output_image_offset = 0;
     for (i = 0; i < KK; i++) {
       input_image_offset = input_image_ptr;
       input_image_offset += col;
       kernel_offset = kernel_ptr;
       for (j = 0; j < KK; j++) {
S1:      temp1 = *input_image_offset++;
S1:      temp2 = *kernel_offset++;
S1:      *output_image_offset += temp1 * temp2;
       }
       kernel_ptr += KK;
       input_image_ptr += NN;
     }
S2: *output_image_offset = ((*output_image_offset)/
normal_factor);
     output_image_offset++ ;
     col++;
   }
   output_image_ptr += NN;
   row++;
  }
}
```





Design Goals

Automatic



ges

Automatic accelerator mapping

-

How close can we get?

“Regression Free”

High Performance

Tool: Polyhedral Modeling

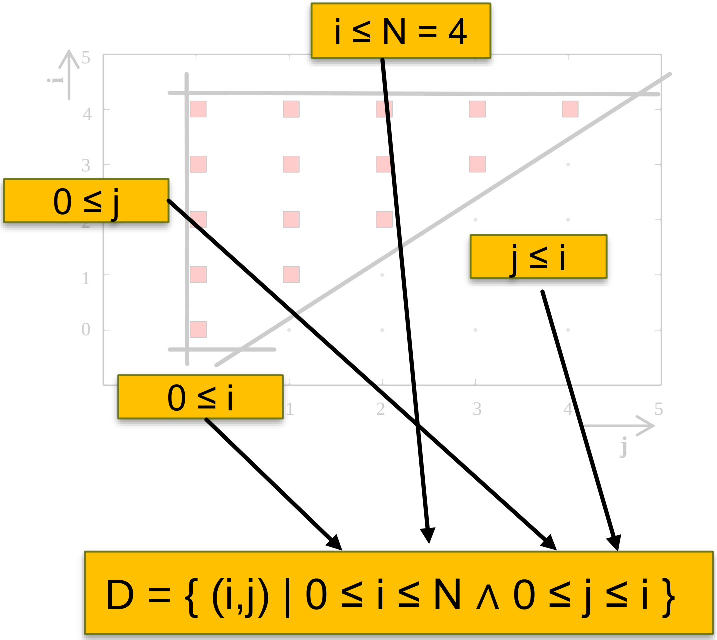
Program Code

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

N = 4

(i, j) = (4,4)

Iteration Space



i ≤ N = 4

0 ≤ j

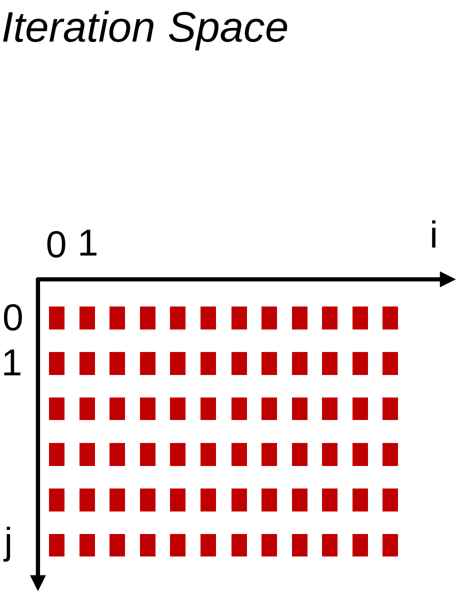
j ≤ i

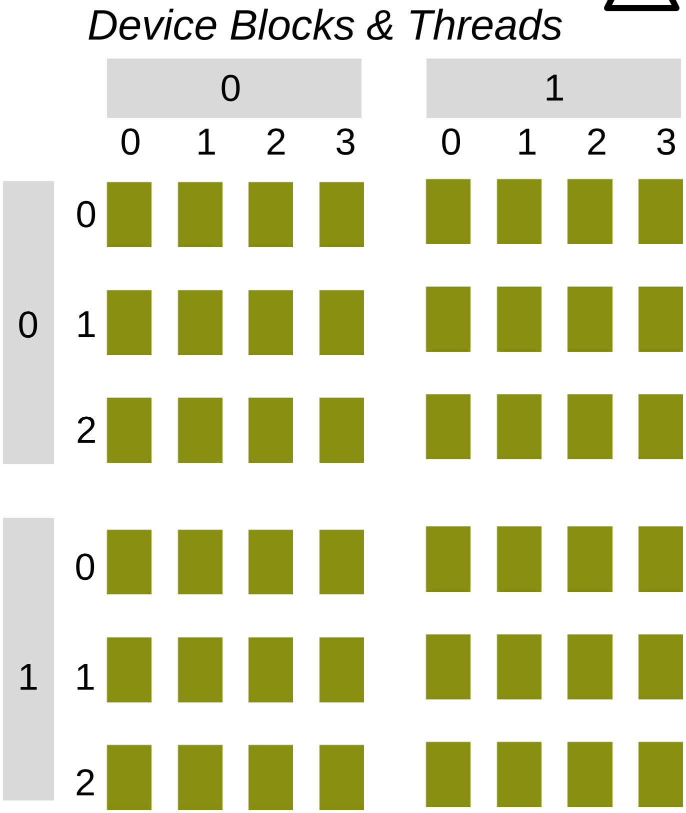
0 ≤ i

D = { (i,j) | 0 ≤ i ≤ N ∧ 0 ≤ j ≤ i }

Polly -- Performing Polyhedral Optimizations on a Low-Level Intermediate Representation
Tobias Grosser et al,
Parallel Processing Letter, 2012

Mapping Computation to Device

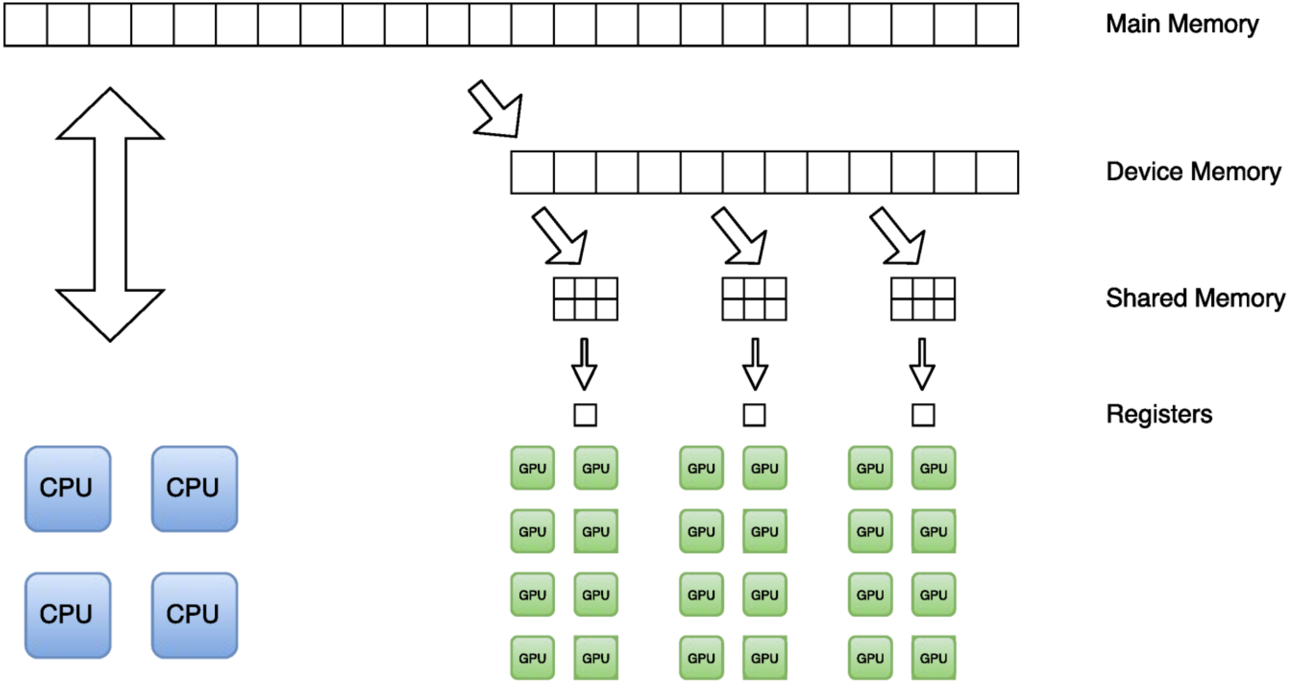




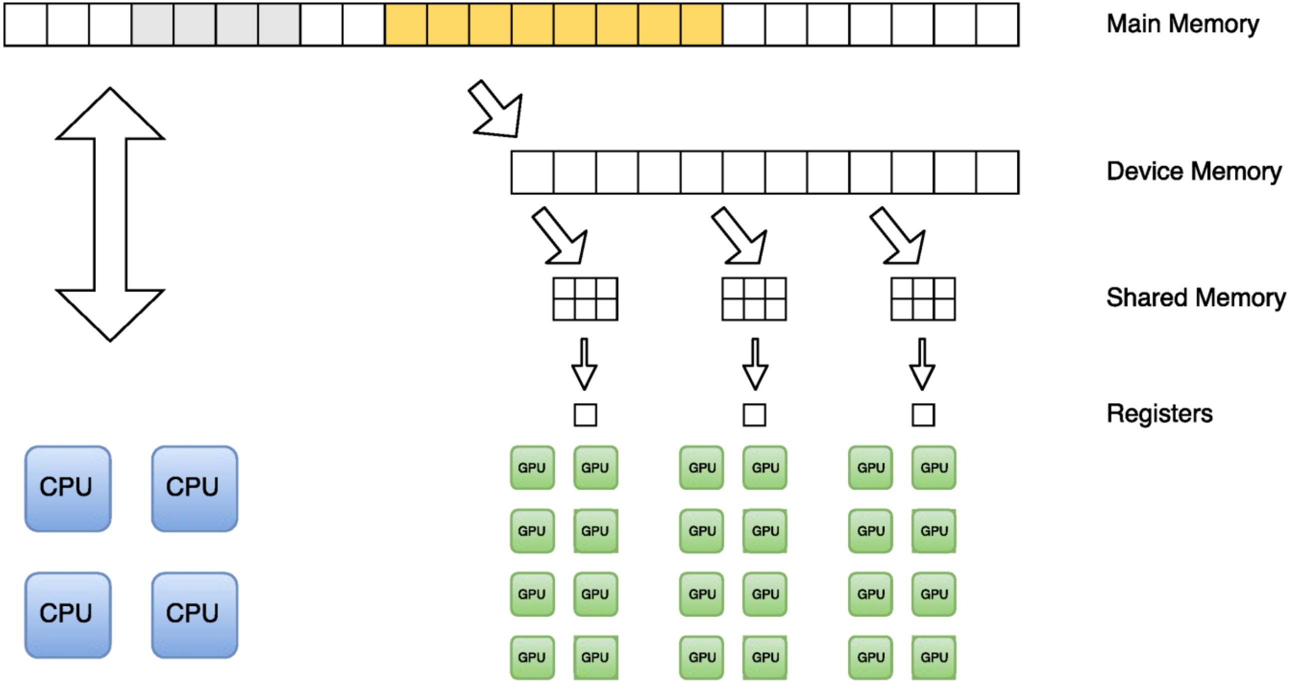
$$BID = \{(i,j) \to \left(\left\lfloor \frac{i}{4} \right\rfloor \% \, 2, \left\lfloor \frac{j}{3} \right\rfloor \% \, 2\right)\}$$

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

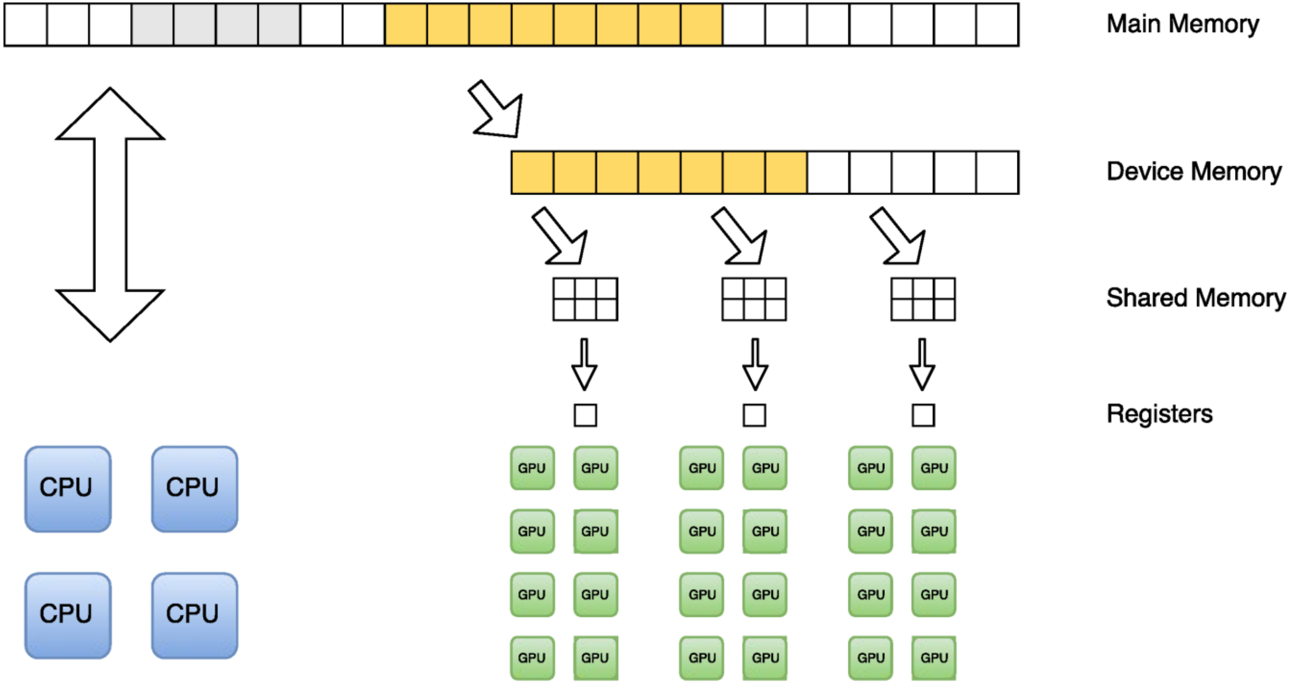
Memory Hierarchy of a Heterogeneous System



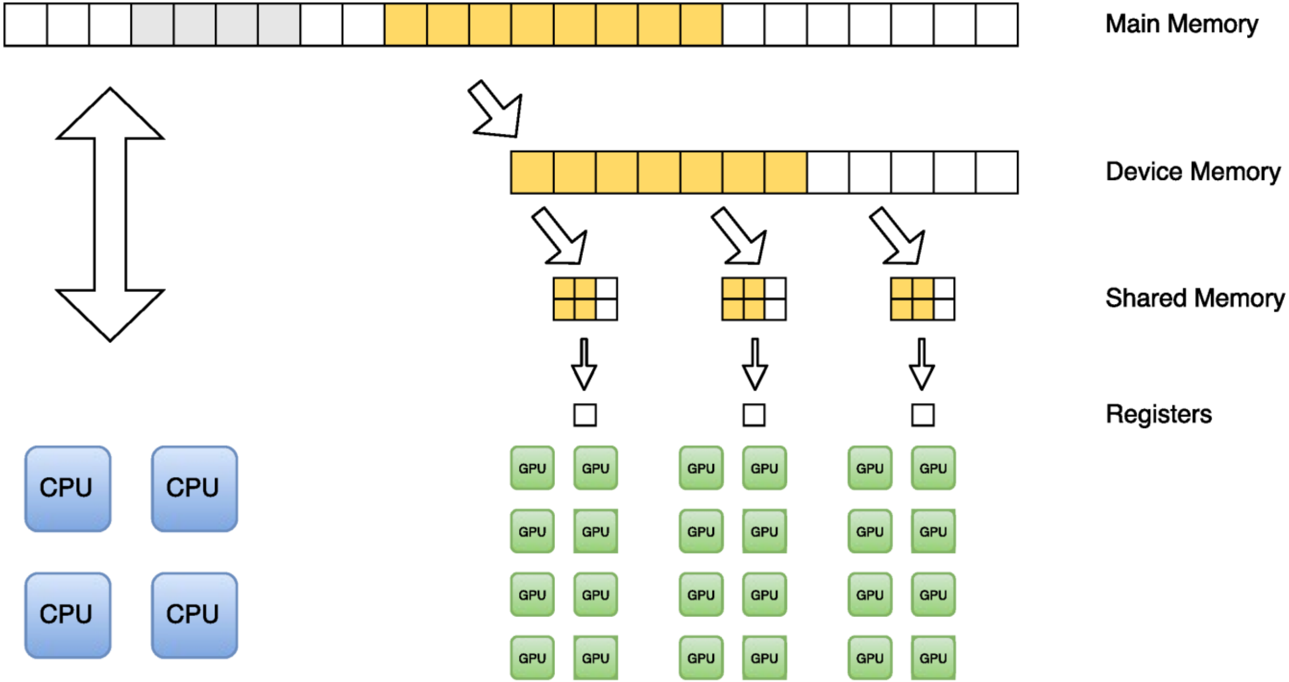
Host-device date transfers



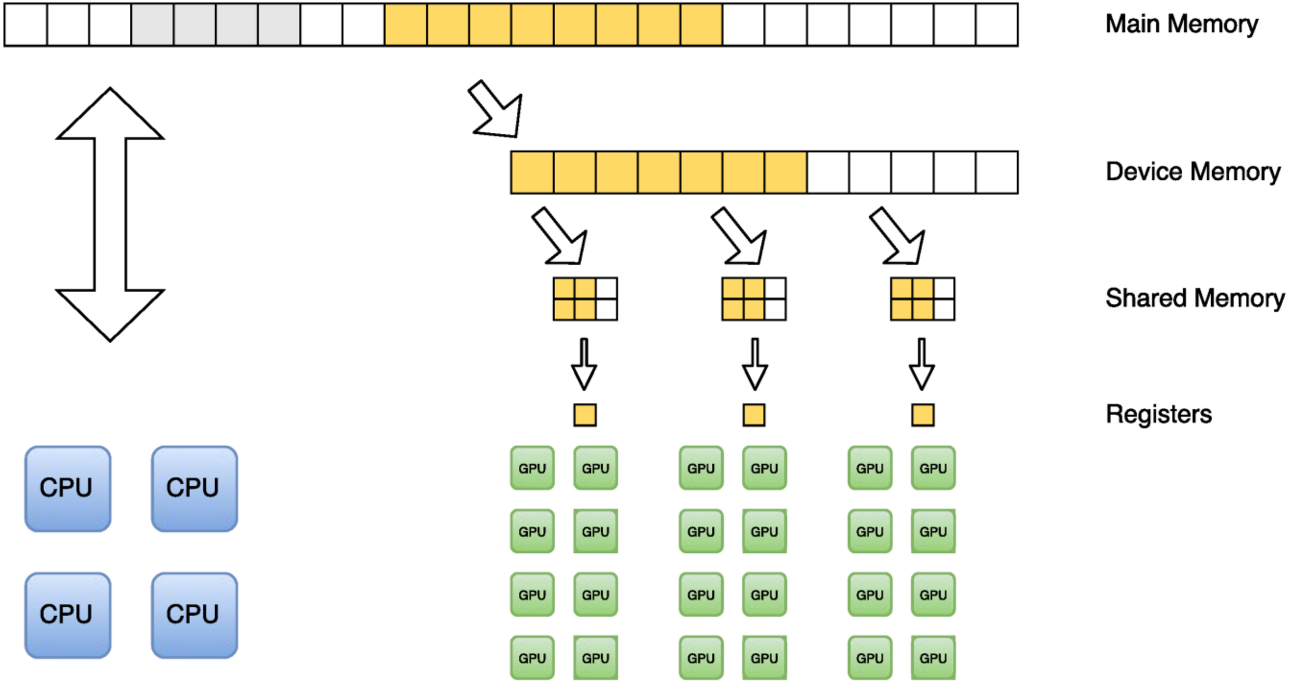
Host-device date transfers



Mapping onto fast memory

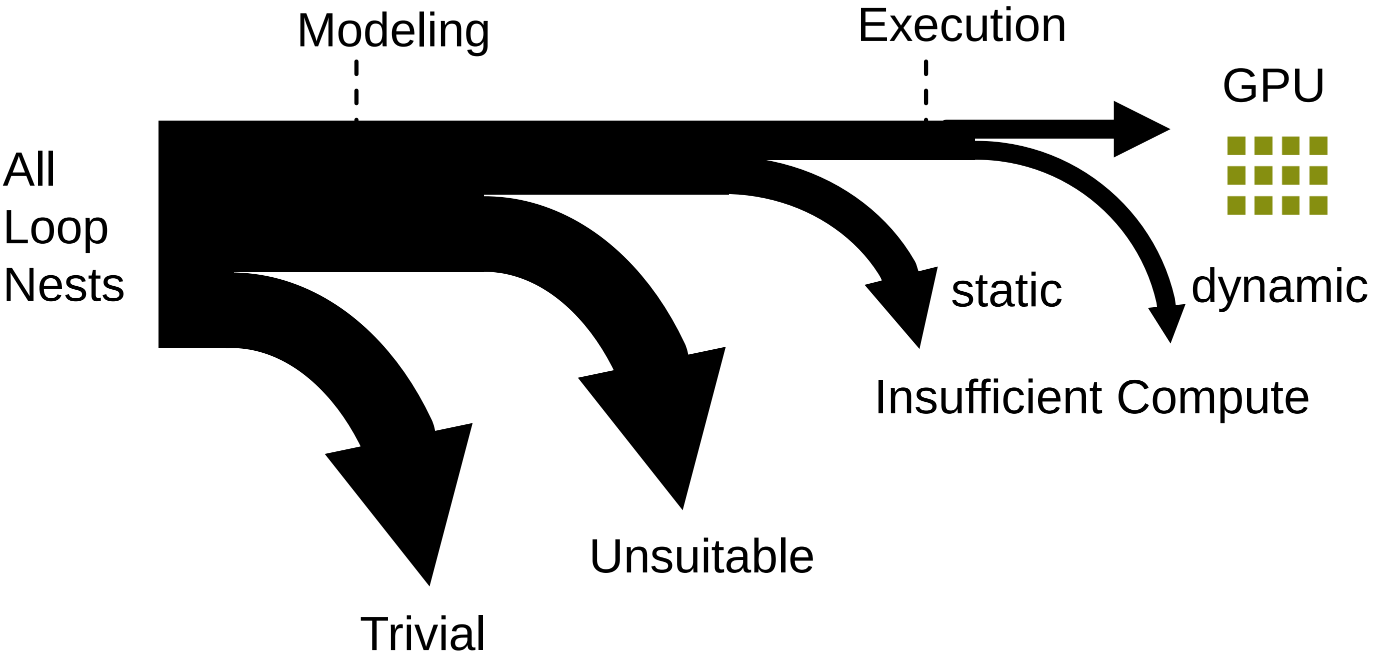


Mapping onto fast memory



Polyhedral parallel code generation for CUDA, Verdoolaege, Sven et. al, ACM Transactions on Architecture and Code Optimization, 2013

Profitability Heuristic



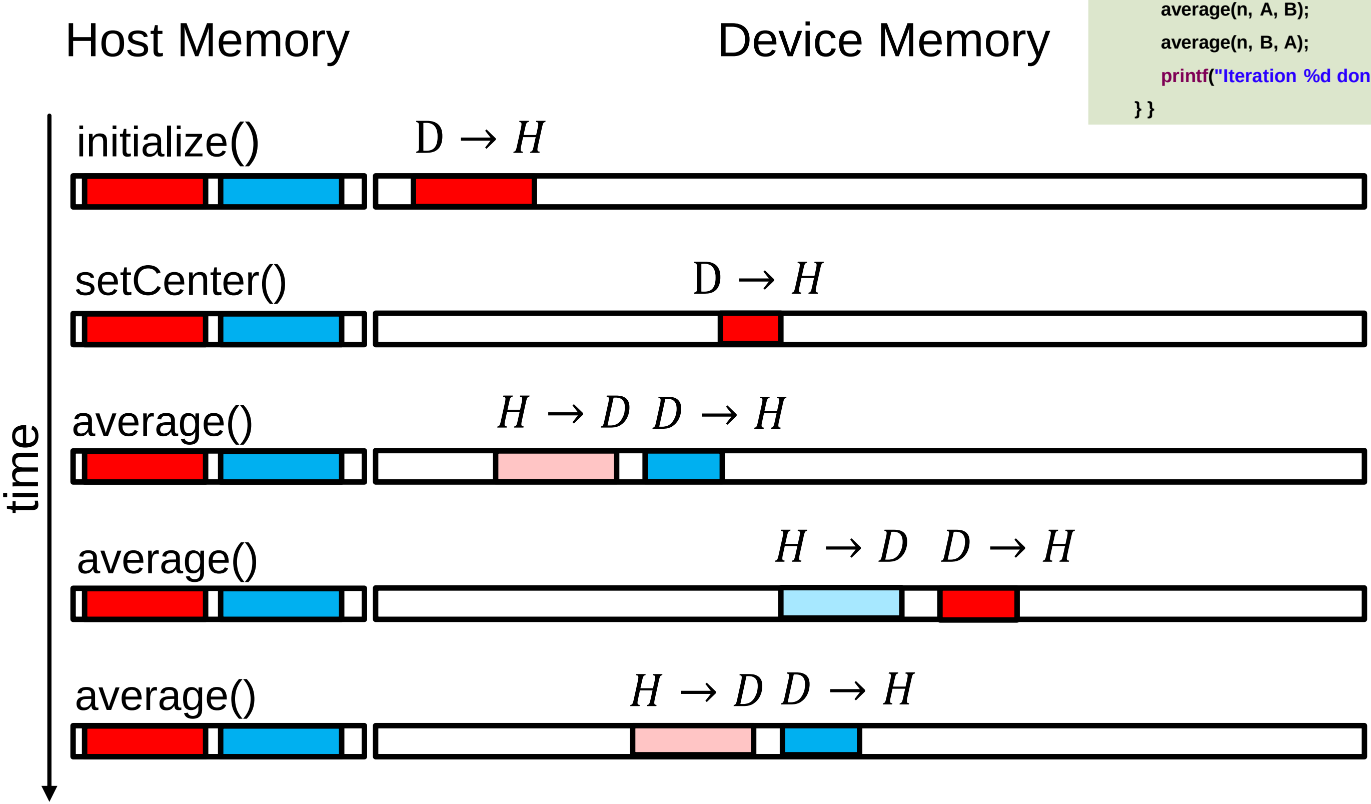
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

From kernels to program – data transfers

```
void heat(int n, float A[n], float hot, float cold) {

    float B[n] = {0};

    initialize(n, A, cold);
    setCenter(n, A, hot, n/4);

    for (int t = 0; t < T; t++) {
        average(n, A, B);
        average(n, B, A);
        printf("Iteration %d done", t);
    }
}
```

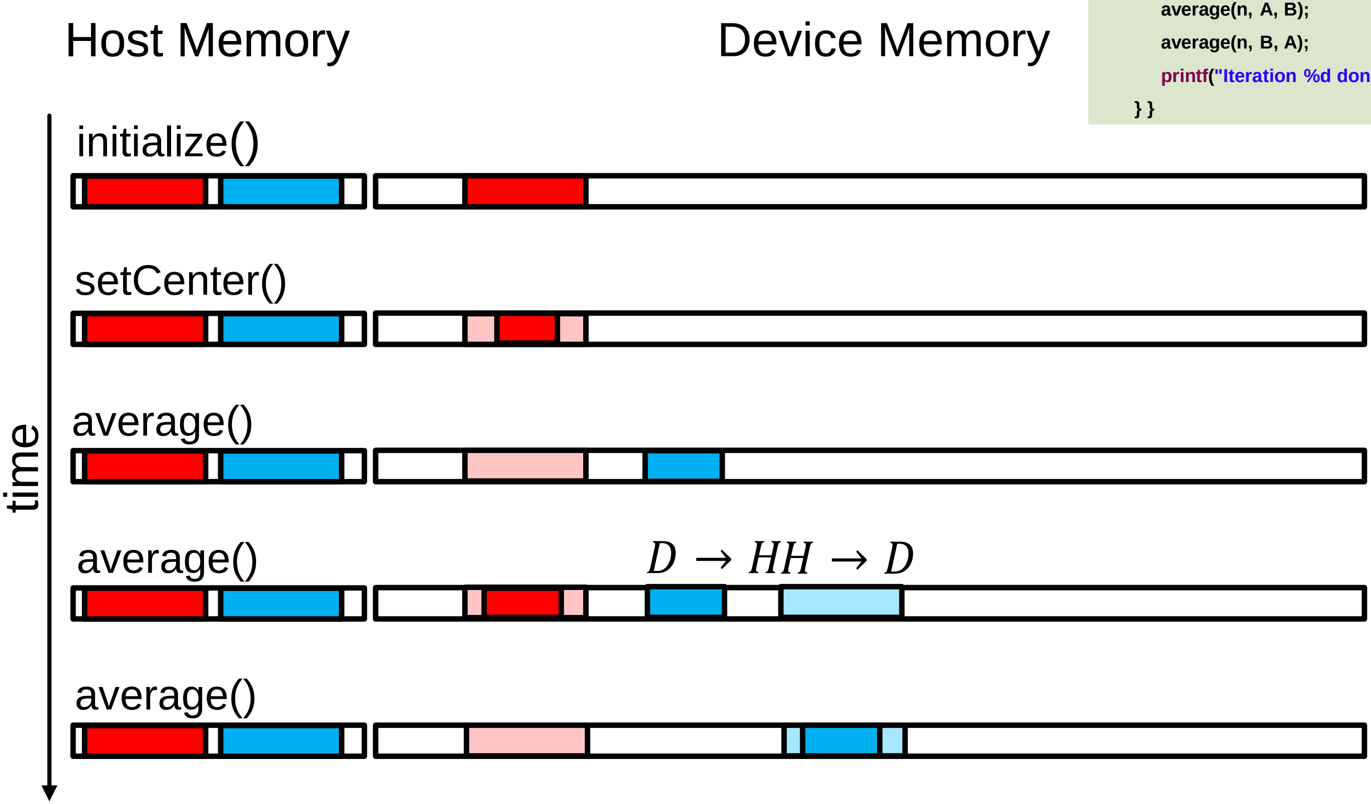
Data Transfer – Per Kernel

```
void heat(int n, float A[n], ...) {
   initialize(n, A, cold);
   setCenter(n, A, hot, n/4);
   for (int t = 0; t < T; t++) {
      average(n, A, B);
      average(n, B, A);
      printf("Iteration %d done", t);
 } }
```


Host Memory
Device Memory
time
initialize()
$\mathrm{D} \rightarrow H$
setCenter()
$\mathrm{D} \rightarrow H$
average()
$H \rightarrow D$ $D \rightarrow H$
average()
$H \rightarrow D$ $D \rightarrow H$
average()
$H \rightarrow D$ $D \rightarrow H$

Data Transfer – Inter Kernel Caching

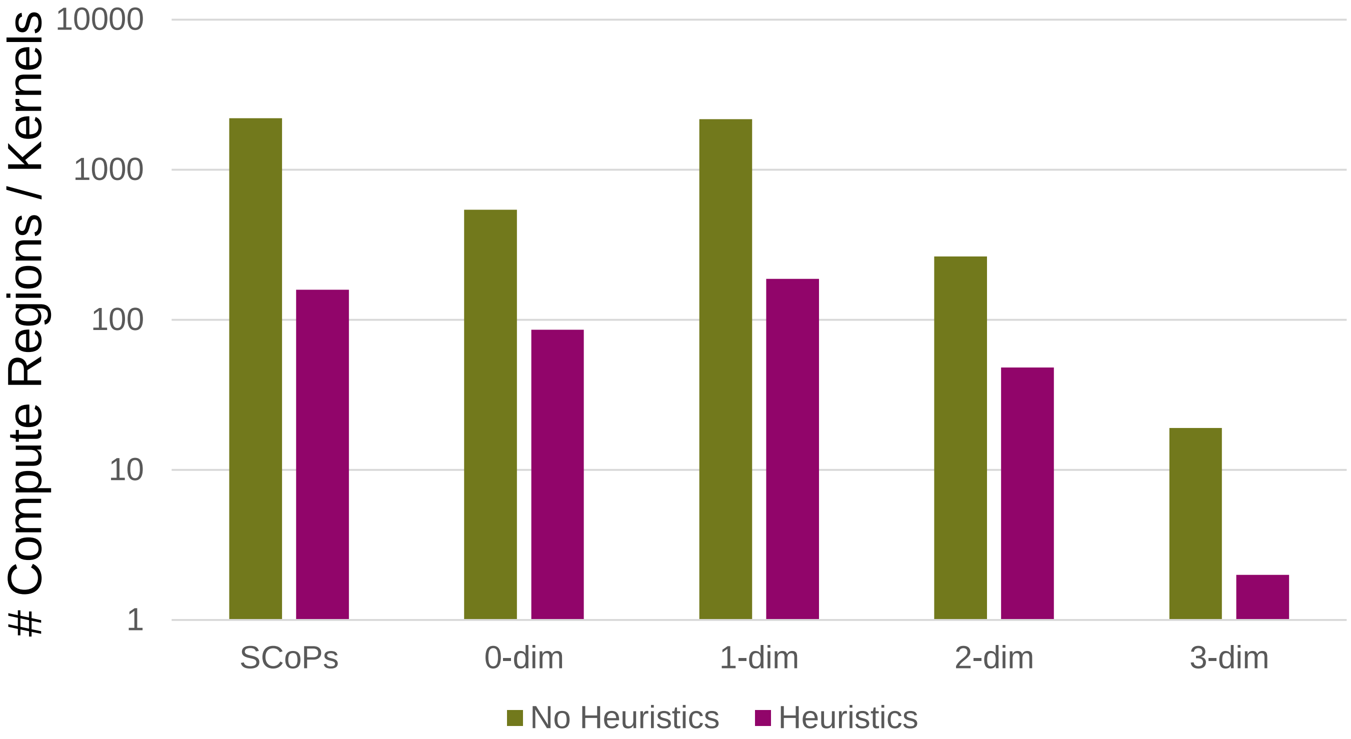
```
void heat(int n, float A[n], ...) {
   initialize(n, A, cold);
   setCenter(n, A, hot, n/4);
   for (int t = 0; t < T; t++) {
      average(n, A, B);
      average(n, B, A);
      printf("Iteration %d done", t);
  } }
```



Evaluation

Workstation:	10 core SandyBridge	NVIDIA Titan Black (Kepler)
Mobile:	4 core Haswell	NVIDIA GT730M (Kepler)

LLVM Nightly Test Suite

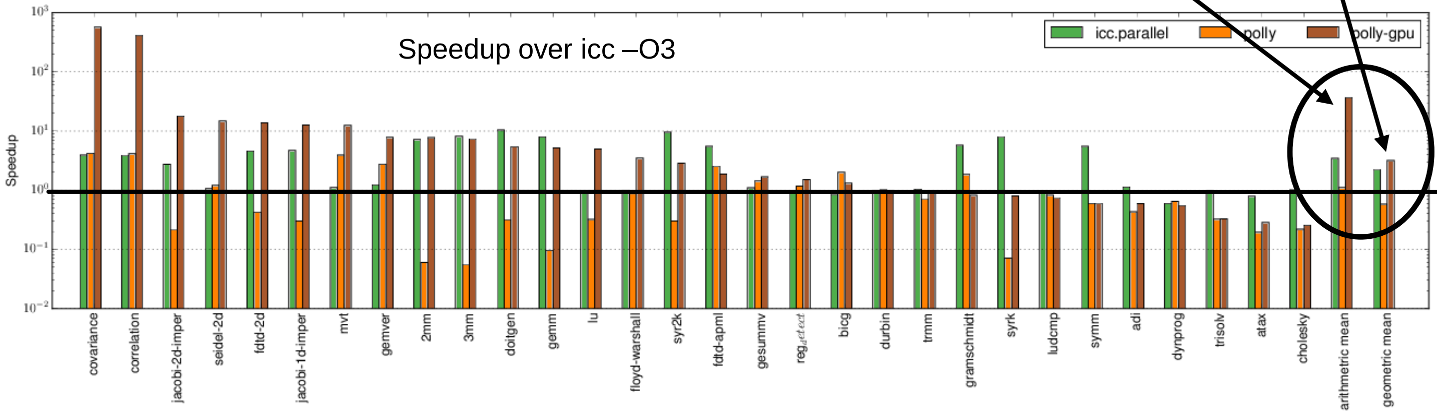


T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

Some results: Polybench 3.2

geomean: ~6x

arithmean: ~30x

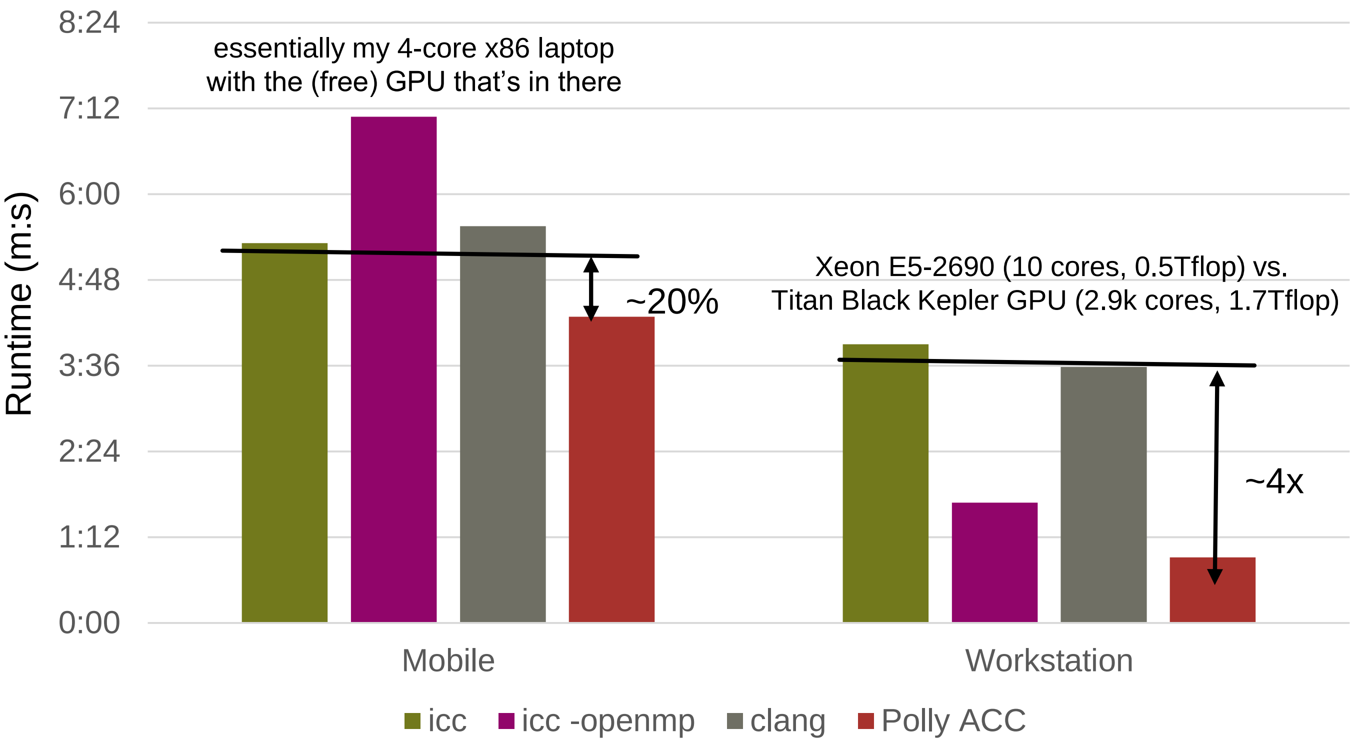


Speedup over icc –O3

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

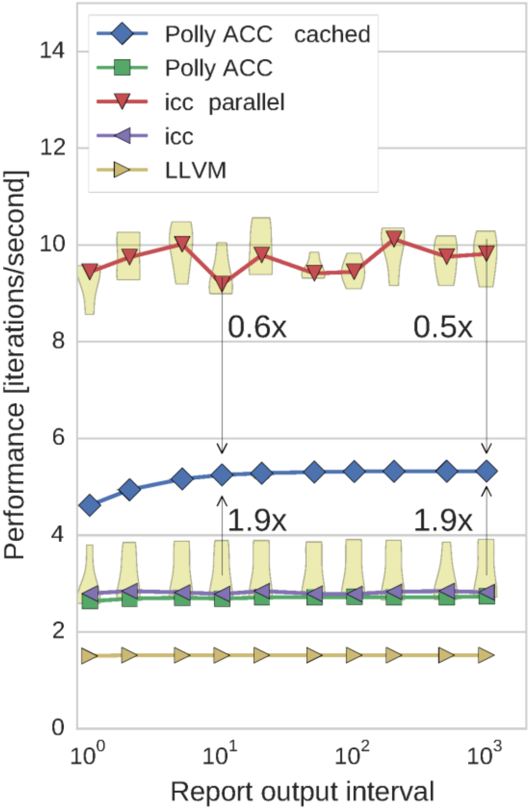
Compiles all of SPEC CPU 2006 – Example: LBM

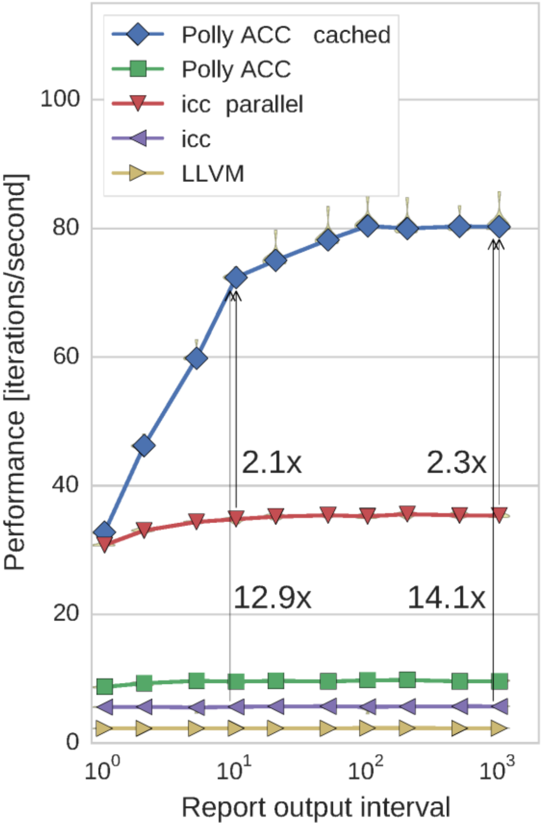


T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

Cactus ADM (SPEC 2006)

Mobile



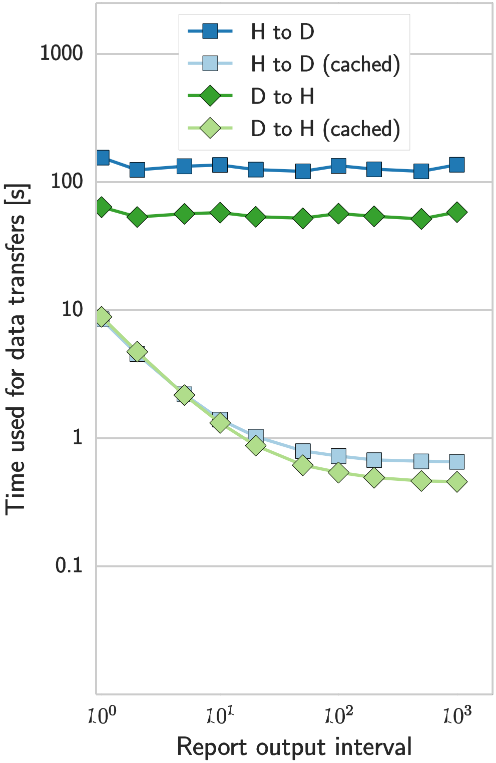


Workstation

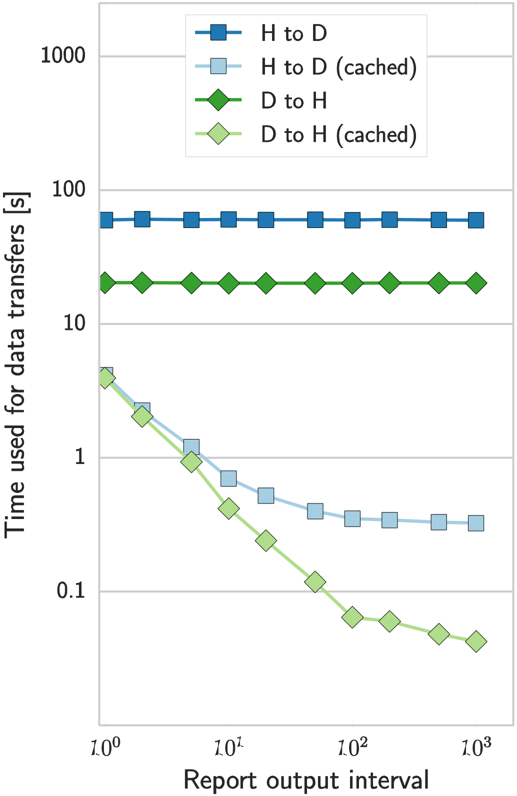
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS’16

Cactus ADM (SPEC 2006) - Data Transfer

Mobile



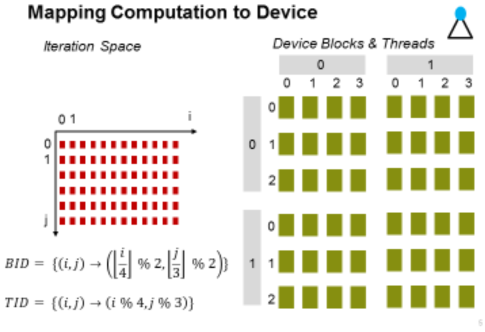
Workstation



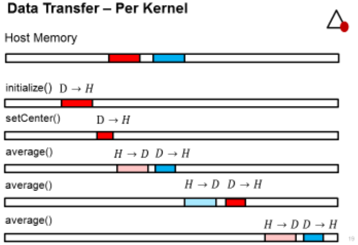
T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16

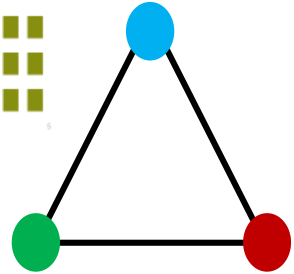
Polly-ACC

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


Mapping Computation to Device
Iteration Space
Device Blocks & Threads
0 1
i
j
0
1
2
3
$BID = \{(i,j) \to (\lfloor \frac{i}{4} \rfloor \% 2, \lfloor \frac{j}{3} \rfloor \% 2)\}$
$TID = \{(i,j) \to (i \% 4, j \% 3)\}$

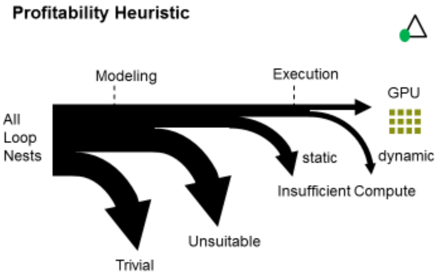
Automatic

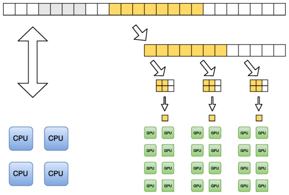


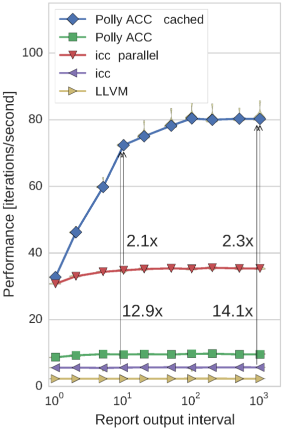


“Regression Free”

High Performance

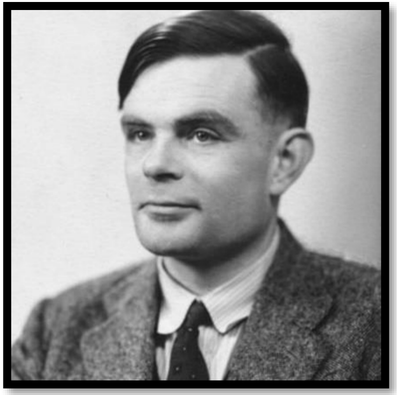






T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS’16

Brave new compiler world!?



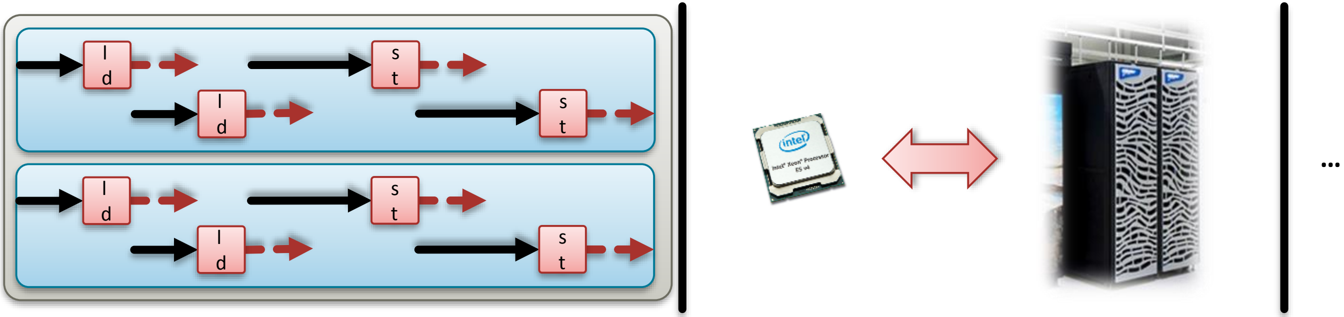
- **Unfortunately not …**
  - Limited to affine code regions
  - Maybe generalizes to control-restricted programs
  - No distributed anything!!
- **Good news:**
  - Much of traditional HPC fits that model
  - Infrastructure is coming along

LLVM COMPILER INFRASTRUCTURE

pasc Platform for Advanced Scientific Computing

- **Bad news:**
  - Modern data-driven HPC and Big Data fits less well
  - Need a programming model for **<u>distributed</u>** 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

device

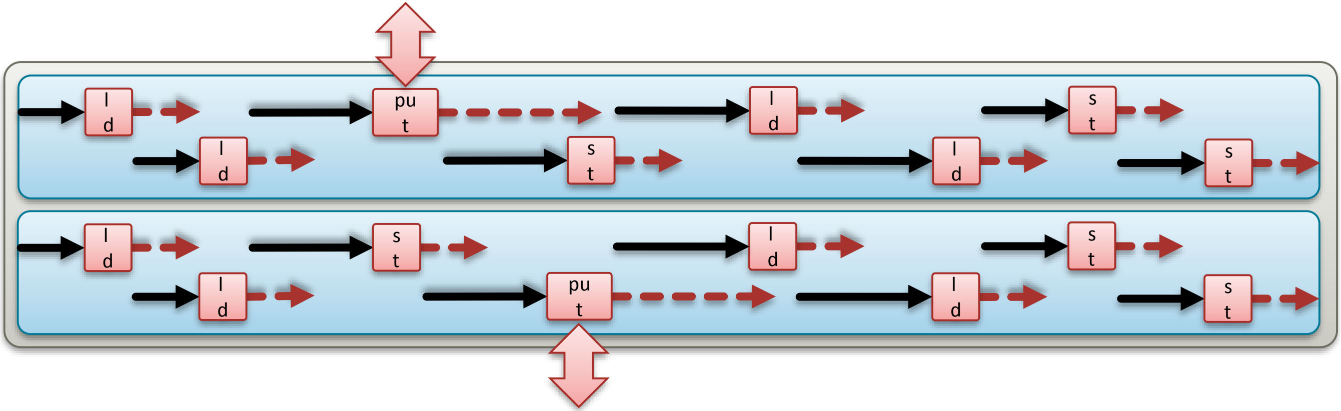
compute core

→ active thread

⇢ instruction latency

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

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)

Talk on Wednesday

Tobias Gysi, Jeremiah Baer, TH: “dCUDA: Hardware Supported Overlap of Computation and Communication”

Wednesday, Nov. 16th

4:00-4:30pm

Room 355-D