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

Fortran
C/C++

Parallel Hardware

Multi-Core CPU

Accelerator
Design Goals

Automatic accelerator mapping
- How close can we get?

“Regression Free”
High Performance
Tool: Polyhedral Modeling

Program Code

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

Iteration Space

\[
D = \{ (i,j) \mid 0 \leq i \leq N \land 0 \leq j \leq i \}
\]

N = 4
(i, j) = (4,4)

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

Iteration Space

Device Blocks & Threads

\[ BID = \{(i, j) \rightarrow (\left\lfloor \frac{i}{4} \right\rfloor \mod 2, \left\lfloor \frac{j}{3} \right\rfloor \mod 2)\} \]

\[ TID = \{(i, j) \rightarrow (i \mod 4, j \mod 3)\} \]
Memory Hierarchy of a Heterogeneous System

Main Memory

Device Memory

Shared Memory

Registers

CPU

CPU

CPU

CPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU

GPU
Host-device date transfers
Host-device date transfers

Main Memory

Device Memory

Shared Memory

Registers

CPU

GPU

CPU

GPU
Mapping onto fast memory

Main Memory

Device Memory

Shared Memory

Registers

CPU

GPU

GPU

GPU

GPU

GPU

CPU

GPU

GPU

GPU

GPU

CPU

GPU

GPU

GPU

GPU

CPU

GPU

GPU

GPU

GPU

CPU
Mapping onto fast memory

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

- **Modeling**
  - All Loop Nests
  - Trivial
  - Unsuitable

- **Execution**
  - static
  - dynamic
  - Insufficient Compute
  - GPU

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

Host Memory

- `initialize()`
- `setCenter()`
- `average()`

Device Memory

- `D → H`
- `D → H`
- `H → D D → H`
- `H → D D → H`
- `H → D D → H`

```c
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);
    }
}
```
**Data Transfer – Inter Kernel Caching**

**Host Memory**

- `initialize()`
- `setCenter()`
- `average()`

**Device Memory**

- `average()`
- `average()`

---

```c
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
Mobile: 4 core Haswell
NVIDIA Titan Black (Kepler)
NVIDIA GT730M (Kepler)
LLVM Nightly Test Suite

# Compute Regions / Kernels

<table>
<thead>
<tr>
<th></th>
<th>No Heuristics</th>
<th>Heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCoPs</td>
<td>10000</td>
<td>10000</td>
</tr>
<tr>
<td>0-dim</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>1-dim</td>
<td>10000</td>
<td>10000</td>
</tr>
<tr>
<td>2-dim</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>3-dim</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

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

- Essentially my 4-core x86 laptop with the (free) GPU that’s in there.

---

Runtime (m:s)

Mobile
- ICC
- ICC-openmp
- Clang
- Polly ACC

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

- ~20% performance improvement
- ~4x speedup
Cactus ADM (SPEC 2006)

T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS'16
Cactus ADM (SPEC 2006) - Data Transfer

Mobile

Time used for data transfers [s]

Workstation

Time used for data transfers [s]

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

Mapping Computation to Device

Data Transfer – Per Kernel

Host Memory

Automatic

“Regression Free”

High Performance

Profitability Heuristic

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

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