Peruse and Profit
Estimating the Accelerability of loops

Snehasish Kumar, Vijayalakshmi Srinivasan, Amirali Sharifian, Nick Sumner, Arrvindh Shriraman
Accelerator design cycle

App

HW/SIM

Kernels

Behavior

How to filter?
What do we need to know from our program?

• Where should I start?
  • Finding regions of interest

• What is acceleratable?
  • Correlating characteristics with execution models

• How to prioritize candidates?
  • Speedup classification
Accelerator design cycle
Peruse

App

HW/SIM

Behavior

Kernels

Filter

Speedup estimation
Where should I start?

• Peruse focuses on *loops*

• Loop characteristics:
  • General
  • Instruction
  • Code
  • Data structure

[Diagram: Loop characteristics:
  • General
  • Instruction
  • Code
  • Data structure
  • Static instructions
    • Memory Allocations
    • Inter-structure
      • Data footprint
    • Annotated Parallel
    • Communications
    • Trip Count
    • Communicate Ratio
    • Min
    • Mutators
    • Loop Exits
    • Function Calls
    • Strided Accesses
    • Vectorizable
  • Dependents
  • Carried Memory
  • Dependencies]
What is Acceleratable?

for (i = 0; i <= N; i++){
    for (j = 0; j <= M; j++){
        data[i][j] -= mean[j];
        data[i][j] /= sqrt(float_n) * stddev[j];
    }
}
What is Acceleratable?

```c
for (i = 0; i <= N; i++){
    for (j = 0; j <= M; j++){
        data[i][j] -= mean[j];
        data[i][j] /= sqrt(float_n) * stddev[j];
    }
}
```

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Memory Dependency: 0</td>
</tr>
<tr>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>Data structure</td>
<td></td>
</tr>
</tbody>
</table>
What is Acceleratable?

for (i = 0; i <= N; i++){
    for (j = 0; j <= M; j++){
        data[i][j] -= mean[j];
        data[i][j] /= sqrt(float_n) * stddev[j];
    }
}

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Memory Dependency: 0</td>
</tr>
<tr>
<td>Instruction</td>
<td>Static IR Ins: 21</td>
</tr>
<tr>
<td>Code</td>
<td>Data structure</td>
</tr>
</tbody>
</table>
What is Acceleratable?

```c
for (i = 0; i <= N; i++){
    for (j = 0; j <= M; j++){
        data[i][j] -= mean[j];
        data[i][j] /= sqrt(float_n) * stddev[j];
    }
}
```

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Memory Dependency: 0</td>
</tr>
<tr>
<td>Instruction</td>
<td>Static IR Ins: 21</td>
</tr>
<tr>
<td>Code</td>
<td>C-to-C Ratio: 4.31 bytes/IR</td>
</tr>
<tr>
<td>Data structure</td>
<td></td>
</tr>
</tbody>
</table>
What is Acceleratable?

```c
for (i = 0; i <= N; i++){
    for (j = 0; j <= M; j++){
        data[i][j] -= mean[j];
        data[i][j] /= sqrt(float_n) * stddev[j];
    }
}
```

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Memory Dependency: 0</td>
</tr>
<tr>
<td>Instruction</td>
<td>Static IR Ins: 21</td>
</tr>
<tr>
<td>Code</td>
<td>C-to-C Ratio: 4.31 bytes/IR</td>
</tr>
<tr>
<td>Data structure</td>
<td>Strided Array Access: 0</td>
</tr>
</tbody>
</table>
Query based interface

• Peruse output for *astra*:

<table>
<thead>
<tr>
<th>Workload</th>
<th>Total</th>
<th>Inner most</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astar</td>
<td>119</td>
<td>44</td>
</tr>
<tr>
<td>Bzip2</td>
<td>244</td>
<td>100</td>
</tr>
<tr>
<td>namd</td>
<td>623</td>
<td>222</td>
</tr>
</tbody>
</table>
Peruse

- App
- HW/SIM
- Behavior
- Kernels

Peruse

- Filter
  - Speedup estimation
Speedup Estimation

PERUSE

Training Phase

• Excellent
• Good
• Moderate
• Poor

PROG.C

Static IR

Filter

Static Features

Dynamic Trace

Machine-Learning

Labels

Perf model

Dynamic Features

Dynamic Tracer

Static IR

Dynamic Features

Dynamic Trace
Speedup Estimation

- PROG.C
- Static Features
- Dynamic Features
- Static IR
- Dynamic Tracer
- Filter
- Classifier
- Test Phase
  - Excellent
  - Good
  - Moderate
  - Poor

PERUSE
Evaluation

• Training set: *Polybench* and *SHOC* (~3200)
• Test set: *470.lbm, 433.milc* (48)

<table>
<thead>
<tr>
<th>Class</th>
<th>True</th>
<th>Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Moderate</td>
<td>0.667</td>
<td>0.095</td>
</tr>
<tr>
<td>Good</td>
<td>0.455</td>
<td>0.081</td>
</tr>
<tr>
<td>Excellent</td>
<td>0.935</td>
<td>0.176</td>
</tr>
</tbody>
</table>

Avg: 79%

Easily predict the best candidates
Summary

- Where should I start?
- What is acceleratable?
- What is speedup estimation?
Question?
Query based interface

Condition:

\[
\text{IsInnermost} = \text{True AND Mem-Deps-Count} = 0
\]

ORDER BY:

\[
\text{Loop-Data-Tile} \text{ ASC, Branch-Ins-Count DESC}
\]

LIMIT:

10

"QUERY" :

{ 
  "limit" : 10,
  "be-true" : [IsInnermost],
  "be-false" : [],
  "where" : [Mem-Deps-count = 0],
  "order-by" : [
    { "loop-data-tile" : ASC },
    { "Branch-Ins-Count" : DESC}
  ]
}
Peruse: Estimating Loop Accelerability

Where should I start?
What is acceleratable?
How to prioritize candidates?

Thursday, 11:50   Room: Lalezar