Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications

Timothy Sherwood       Erez Perelman
Brad Calder
University of California, San Diego
Motivation

• Architecture researchers conduct detailed pipeline simulations

• Length of detailed pipeline simulation
  – Simple Scalar: 400 million instruction per hour
  – Spec programs: 300 billion instructions
  – Complete run: 1 month

• Limited simulation time and processing power

• Often only a subset of whole program is simulated
• Subset should represent the overall behavior of the program
Phases of Execution

• Initialization phase
  – Initialize data structures and set up for the rest of execution
  – Does not represent overall behavior of program
  – Current methods: fast forward or check points

• Steady state
  – Programs tend to be written in a nested loop fashion
  – Correlated with looping behavior of program
Cyclic Behavior of Wave

Instructions Executed (billions)

Data Miss Rate

Branch Miss Rate / IPC

DL164 branch IPC

Period

Initialization
Goals of Research

• Automatically generate:
  – Length of initialization phase
  – Period length
    • Cyclic portion of execution
  – Ideal starting simulation point
    • For a given number of instructions

• Confidence of simulation points
  – Estimation of accuracy
Outline

• Basic Block Distribution Analysis
• Initialization Phase
• Period
• Where to Simulate
• Conclusion
Approach

- A way to represent snapshots of program
- A metric that compares snapshots to whole program
- Uniquely identify phases of execution
- Signal processing for period computation
Program Fingerprint

• Metric independent method to represent program

• Basic Blocks uniquely identify the code executed
  – Directly affects program behavior

• Unique representation of program execution interval
  • BB vector
## Basic Block Vector

<table>
<thead>
<tr>
<th>BB</th>
<th>Assembly Code of bzip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>srl a2, 0x8, t4</code></td>
</tr>
<tr>
<td></td>
<td><code>and a2, 0xff, t12</code></td>
</tr>
<tr>
<td></td>
<td><code>addl zero, t12, s6</code></td>
</tr>
<tr>
<td></td>
<td><code>subl t7, 0x1, t7</code></td>
</tr>
<tr>
<td></td>
<td><code>cmpeq s6, 0x25, v0</code></td>
</tr>
<tr>
<td></td>
<td><code>cmpeq s6, 0, t0</code></td>
</tr>
<tr>
<td></td>
<td><code>bis v0, t0, v0</code></td>
</tr>
<tr>
<td></td>
<td><code>bne v0, 0x120018c48</code></td>
</tr>
<tr>
<td>2</td>
<td><code>subl t7, 0x1, t7</code></td>
</tr>
<tr>
<td></td>
<td><code>cmpeq t7, 0x3, t2</code></td>
</tr>
<tr>
<td></td>
<td><code>beq t2, 0x120018b04</code></td>
</tr>
<tr>
<td>3</td>
<td><code>ble t7, 0x120018bb4</code></td>
</tr>
<tr>
<td>4</td>
<td><code>and t4, 0xff, t5</code></td>
</tr>
<tr>
<td></td>
<td><code>srl t4, 0x8, t4</code></td>
</tr>
<tr>
<td></td>
<td><code>addl zero, t5, s6</code></td>
</tr>
<tr>
<td></td>
<td><code>cmpeq s6, 0x25, s0</code></td>
</tr>
<tr>
<td></td>
<td><code>cmpeq s6, 0, a0</code></td>
</tr>
<tr>
<td></td>
<td><code>bis s0, a0, s0</code></td>
</tr>
<tr>
<td></td>
<td><code>bne s0, 0x120018c48</code></td>
</tr>
<tr>
<td>5</td>
<td><code>subl t7, 0x1, t7</code></td>
</tr>
<tr>
<td></td>
<td><code>gt t7, 0x120018b90</code></td>
</tr>
</tbody>
</table>

## BB Vector

<table>
<thead>
<tr>
<th>BB#</th>
<th># times executed</th>
<th>Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>0.250626</td>
</tr>
<tr>
<td>2</td>
<td>89</td>
<td>0.223057</td>
</tr>
<tr>
<td>3</td>
<td>83</td>
<td>0.208020</td>
</tr>
<tr>
<td>4</td>
<td>71</td>
<td>0.177944</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>0.140350</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Basic Block Vector Comparison

- **Target Vector**: BB vector of complete run
- **Interval Vector**: BB vector of a continuous interval of execution in program
- **Vector Difference**: how close BB vector is to the target vector

<table>
<thead>
<tr>
<th>BB Interval Vector</th>
<th>BB Target Vector</th>
<th>Diff BB Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB#</td>
<td>Normalized</td>
<td>BB#</td>
</tr>
<tr>
<td>1</td>
<td>0.250626</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.223057</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0.208020</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0.177944</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0.140350</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

$S = 0.310221$
Basic Block Difference Graph

Wave

Hydro

Instructions Executed (100 millions)
• Basic Block Distribution Analysis
• Initialization Phase
  • Period
  • Where to Simulate
• Conclusion
Initialization Phase

• Create a Basic Block Difference Graph of initialization
  – Target vector is first 100 million instructions

• End of Initialization
  – The max vector diff point in graph
    • In most cases is 2
Initialization Phase

Wave

BB Diff

Hydro

BB Diff

Instructions Executed (100 millions)
• Basic Block Distribution Analysis
• Initialization Phase
• Period
• Where to Simulate
• Conclusion
Signal Processing Theory

• Treat BB Diff Graph as a signal
• Signal shift and comparison
  – Signal shift will go in-and-out of phase
  – Comparison to evaluate phase
• Period deduced from phase cycle
Signal Difference Example

Signal Phasing

Period Difference Graph
Period

- Start signal at end of initialization
  - Pick portion to shift to be quarter length of signal
- Shifting: generate Period Difference Graph
  - Minimums correlate to period-synchronized shifts
  - Amplifies the cycle over the BB Diff Graph
- Calculate period
  - Find all minimums
  - Calculate average distance between adjacent minimums
**Period Difference Graphs**

**Wave**

Phase Diff

Period = 6.8 billion

**Hydro**

Phase Diff

Phase shift (100 million instructions)

Period = 1.7 billion
Initialization and Period

- Instructions in billions:
  - bzip: 104.7
  - hydro: 14.4
  - tomcat: 125.9
Outline

• Basic Block Distribution Analysis
• Initialization Phase
• Period
• Where to Simulate
• Conclusion
Where to simulate

• Not always possible to simulate full period
• Basic Block Distribution Analysis generates best simulation point for desired simulation duration
  – User inputs desired simulation duration
  – BB Distribution Analysis generates a BB Difference Graph with BB vector length equal to sim duration
  – Take min point in BB Difference Graph
    • Start simulation at that point
Accuracy of Simulation Points

<table>
<thead>
<tr>
<th>% diff in IPC</th>
<th>bzip</th>
<th>hydro</th>
<th>tomcat</th>
<th>vortex</th>
<th>vpr</th>
<th>wave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>337%</td>
<td>380%</td>
<td>162%</td>
<td>244%</td>
<td>114%</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Full Period
- 300 million Sim Point
- 300 million after init
- First 300 million
Simulation Point Tool

• Input:
  – Program BB execution history
    • BB vector for every execution interval
  – Desired simulation duration

• Output:
  – End of initialization phase
  – Length of 1 period
  – Best simulation point
Key Points

• Focused on continuous simulation
• Basic Block approach is metric independent and correlates to program behavior
• Program behavior varies during execution
• Beneficial to find the best simulation point
• Not necessary to simulate full cycle for a good sample of overall program behavior
Conclusions

• BBDA is an effective method to find the initialization phase, period, and where to simulate programs
• BBDA is a time-conserving tool for researchers
• BBDA 300 million instructions simulation point produce average IPC error rates < 6%
Current Work

• Period with Fourier Analysis
  – Fast Fourier Transform
  – Breaks down signal into dominant frequencies
  – Period derived from dominant frequency

• Benefits
  – Multiple periods throughout execution
Fourier Analysis