Motivation

- **Ideal:** Perfectly understand how an executing program interacts with the underlying architecture so we can adapt to its needs

- **Challenge:** To make sense of this sea of data and extract important information

- **How:** Use the notion of Phases to build a high-level model of program behavior that we can then use for optimization
Outline of This Talk

- Analyzing Program Phase Behavior
  - Definition
  - Detection
  - Classification
- Using Phase Behavior
- Tracking Phase Behavior in HW
- Future Work

Common Assumptions

✘ Assumption: The average statistics sufficiently characterize behavior
  Instead program behavior varies wildly as phases change over time

✘ Assumption: A sample from the middle of a program will be representative
  Behaviors change over even the largest time scales and we need to be able to quantify these changes
How Programs Change

- Goal: Capture Time Varying Behavior

- Program Behavior is neither
  - Completely Homogenous
  - nor Totally Random

- Program Behavior is
  - typically quite structured

- Discover and represent this structure
Behavior of gzip

**Definitions**

- **An interval** is a selection of contiguous instructions in program execution order
  - Think of it as a slice in time

- **A phase** in a program is a set of intervals that similar to each other
  - The program should have similar behavior (no matter how it is measured) in all intervals of a single phase
Program Phases

<table>
<thead>
<tr>
<th>Oblivious</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard to model behavior</td>
<td>High Level Picture</td>
</tr>
<tr>
<td>Optimization have too wide of a target</td>
<td>Target individual phases of execution</td>
</tr>
<tr>
<td>Adaptive optimizations constantly readjusting</td>
<td>Feedback can be broken down by phase</td>
</tr>
</tbody>
</table>

Challenges

- **Difficult to Gather Data**
  1 minute real time = 50 days of simulation
  
  ➡️ Find Phases without cycle-level detail

- **Phases are Complex**
  Gzip is a simple example
  
  ➡️ Find Phases Automatically

- **Must be General Purpose**
  Independent of optimization
  
  ➡️ Cannot rely on performance metric

• The key: Track the Code
Outline of This Talk

- Analyzing Program Phase Behavior
  - Definition
  - Detection
    - Classification
- Using Phase Behavior
- Tracking Phase Behavior in HW
- Future Work

You are what you execute

- **Goal** - track behavior of a program
  - Behavior *caused by* the path through code
- **How** - Track the code that is executing
  - Detect changes and similarities in code
- **Basic Block Distribution Analysis**
Basic Block Distribution Analysis

Very fast to capture

We can now compare vectors

Start with simple manual analysis

– Compare all $N^2$ pairs of intervals

• Enter the Similarity Matrix…
Behavior of gzip

Similarity Matrix

- Compare $N^2$ intervals
- Executed Instructions on Diagonal axis
- To compare 2 points go horizontal from one and vertically from the other
- Darker points indicate similar vectors
- Clearly shows the phase-behavior without any detailed simulation
A More Complex Matrix – gcc

- Still much structure
- Dark boxes show phase-behavior
- Boxes in interior show recurring phases
  Strong diagonal line indicates first half is similar to second half (indicate potential predictability)
- Manual inspection is not feasible or scalable

Outline of This Talk

- Analyzing Program Phase Behavior
  - Definition
  - Detection
  - Classification
- Using Phase Behavior
- Tracking Phase Behavior in HW
- Future Work
Finding the Phases

- Basic Block Vector is a point in space
- The problem is to find groups of vectors/points that are all similar
  - Making sure that all points in a group are similar to one another
  - And ensuring all points that are different, are put into different groups
- This is a Clustering Problem
- A Phase is a Cluster of BBVectors

Phase-finding Algorithm

I. Profile Program and track BB Vectors

II. Use the K-means algorithm to find clusters in the data for many different values of K

III. Score the likelihood of each clustering

IV. Pick the best clustering based on the likelihood and K
Improving Performance

- K-means requires many manipulations
  - Basic Block Vectors are very long
    - > 100,000 for gcc; 800,000 for microsoft apps
  - Need to
    - make the Vectors smaller
    - Still preserve relative distances

- Random Projection
  - Can safely reduce down to 15 dimensions
  - Reduce run-time from days to minutes

Example: gzip Revisited

![Graph showing various performance metrics over time](image)
gzip – Phases Discovered

**Performance (IPC)**
- Energy used per interval
- Instruction cache misses
- Data cache misses
- Branch misprediction
- 2nd level cache misses

gcc - A Complex Example

**Performance (IPC)**
- Energy used per interval
- Instruction cache misses
- Data cache misses
- Branch misprediction
- 2nd level cache misses
Outline of This Talk

- Analyzing Program Phase Behavior
  - Definition
  - Detection
  - Classification
- Using Phase Behavior
  - Tracking Phase Behavior in HW
  - Future Work
Phase-Guided Simulation

- **Problem**: Detailed simulation is very costly
  - Yet essential to computer architecture
  - What effect will a change have on my processor?

- **A Solution**: Use Phases to Guide Simulation
  - Quantify the phase behavior (quickly)
  - Study one element of each phase

- **How**: Allows a hierarchy of detail
  - View very detailed behavior
  - But puts it in larger context

Errors in Simulation Methods

<table>
<thead>
<tr>
<th></th>
<th>From Start</th>
<th>Skip 1 Billion</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>13%</td>
<td>68%</td>
</tr>
<tr>
<td>gcc</td>
<td>33%</td>
<td>51%</td>
</tr>
<tr>
<td>Median</td>
<td>23%</td>
<td>58%</td>
</tr>
</tbody>
</table>
Errors in Simulation Methods

From Start | Skip 1 Billion | Sample Per Phase

Error in Performance Estimation (IPC)

gzip | gcc | Median

- From Start
- Skip 1 Billion
- Sample Per Phase

Errors in Simulation Methods

From Start | Skip 1 Billion | Sample Per Phase

Error in Performance Estimation (IPC)

gzip | gcc | Median | Max

- From Start
- Skip 1 Billion
- Sample Per Phase

37x | 19x
Outline of This Talk

- Analyzing Program Phase Behavior
  - Definition
  - Detection
  - Classification

- Using Phase Behavior

- Tracking Phase Behavior in HW

- Future Work

Power Management

- **Problem**: Power is a first class design constraint
  - Much wasted power from unused resources

- **Ideal**: Power down unused chip resources

- **Challenge**: Turning off a section of chip is slow

- **A Solution**: Phases can allow use to make decisions that look at long term behavior
Tracking Phases at Runtime

- **We Need**: ability to classify phases
  - Online (no multiple pass methods)
  - With a minimum of resources
  - To predict future phases
- **Our Approach**: A simplification
  - Software approach is very effective
  - Trim it down, increases error
  - We need a Phase Predictor

Phase Detection Hardware

- Track Basic Blocks
- Assemble BB Vectors
- Random Projection
- Clustering
Histogram of IPC (gcc)

Histogram of IPC (gcc)
Into the Future: with phases

- Now we have a stream of Phase IDs
  - But to be active we need future behavior
- Architect’s Solution: Build a Predictor
- Typical way: Approximate Markov
  - Phase ID at $t, t-1, \ldots t-n \rightarrow t$
  - $n$ is the “Order” of the model
  - Phase IDs have long runs, not $n$-th order
  - Need a different stream

Encode the Phase ID stream

Phase: \texttt{AAAAABBBBBAAAAABBBBB}

RLE: \texttt{(A,5) (B,4) (A,5) (B,4)}

One last trick: As we go- \texttt{(A,1) (A,2) (A,3)}
This will eat up all the table space..

Instead: only encode rules that match other Phase IDs
A RLE-Markov Phase Predictor

Check for Last Phase

RLE Markov Predictor

Phase ID

Last ID

Run Count

+1

0

T

F

H

Count the RLE length

Throttling Back Execution

energy saved

0.0% 5.0% 10.0% 15.0% 20.0% 25.0% 30.0% 35.0% 40.0%

20.0% slowdown

gcc 2-wide

gzip 2-wide

avg 2-wide

gcc phase

gzip phase

avg phase

Timothy Sherwood 39

UCSB

Timothy Sherwood 40

UCSB
Throttling Back Execution

- GCC 2-wide
- Gzip 2-wide
- Avg 2-wide
- GCC phase
- Gzip phase
- Avg phase

Better than Voltage Scaling
Worse than Voltage Scaling

Energy Saved vs. Slowdown

Timothy Sherwood 41
UCSB
Other Applications

- Greatly reduce simulation time
- Platform for stable optimizations
  - Power management
  - Targeted profiling
- Optimizations on per-phase basis
- Using phases for scheduling
- Model the full program
  - Puts detailed level data in context
  - Visualization

More Information

- Brief overview of techniques

- Phase Analysis tool available for use
  - Portable code for tracking phases
  - google://simpoint

- Thanks to: Suleyman Sair, Erez Perelman, Greg Hamerly, and Brad Calder

- http://www.cs.ucsb.edu/~sherwood