Run-Time Program-Specific Phase Prediction for Python Programs

Meng-Chieh (Joe) Chiu
Eliot Moss

{joechiu,moss}@cs.umass.edu
University of Massachusetts, Amherst
Introduction

Program Phase
Introduction

• Goal: run-time phase prediction
• For Python programs
• Low run-time overhead
• High quality prediction

Phase Change
Why do this?

• Software engineers: *program analysis* and understanding
• Computer architects: *impact* on micro-architectural components (*performance*)
• Practical applications:
  • Saving energy
  • Scheduling garbage collections, etc.
The Challenge

• Phase prediction needs information

• Too much info: high overhead

• Too little info: low quality
Event Traces & Feature Vectors
Tracing Tool → Event Traces

• BranchProfiler (customized)
  
github.com/MengChiehChiu/python_tracing_tool

• Traces record every branch in a program’s execution

• Benchmark: 13 of 44 Pyperformance benchmarks
Event Traces → Feature Vectors

**Program**

- **Tracing Tools**

- **Feature Vectors**

| Interval: 10k branches | F1 | F2 | F3 | ...
|------------------------|----|----|----|-----
| T1                     | 0  | 2  | 20 |     
| T2                     | 3  | 5  | 0  |     
| T3                     | 1  | 4  | 0  |     
| .                      | .  | .  | .  | .   
| .                      | .  | .  | .  | .   
| .                      | .  | .  | .  | .   

- **Call A**
  - Branch 1
  - Branch 1
  - Branch 2

- **Exit A**
- **Call B**
- **Branch 3**
- **Exit B**
Ground Truth of Phases
Ground Truth of Phases
Ground Truth of Phases

- Use phase transition ground truth
  - Nagpurkar 2016
- Partition based on ground truth
- L1 norm on each chunk of features
- Use GMM to cluster
Gaussian Mixture Model

• Choosing number of multi-dimension Gaussian distributions
• A feature vector has probability
• A feature vector associates with Cluster with highest probability
Gaussian Mixture Model

Cluster 1

Cluster 2
Ground Truth of Phases

Phase Transition (Nagpurker 2006)

Feature f1

f2

fn

L1-norm

GMM

3 ~ 5 clusters
Feature Selection

- Program
- Trace
- Feature vectors
- Feature Selection
- Instrumented Programs
- Outputs
- Compare
- Ground Truth
- Evaluation
- Model
- Instrument
- ML
- Feature Sets
- TT
- FE
Feature Selection

Need to select features with:

• High diversity
  • Low correlation with each other

• Low run-time overhead
  • Low cost to count at run time
Feature Selection

- Loss function
- Sort features by loss
- Select first $k$ features lowly-correlated with each other
- Replace each feature with lower cost, highly correlated proxy
# Feature Selection

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Sorted by loss function

COST
## Feature Selection

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### COST

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>7</th>
<th>3</th>
<th>1</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- F4 and F5 are selected with the lowest cost.
- F3 is the highest cost.
## Feature Correlation

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>F4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COST</th>
<th>F4</th>
<th>F6</th>
<th>F3</th>
<th>F1</th>
<th>F2</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Correlation threshold: 0.8

Correlation: 0.3
Low Correlation with F4

Correlation threshold: 0.8

Correlation: 0.1

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>F4</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
High Correlation with F6

Correlation: 0.9

Correlation threshold: 0.8

Selected Features

| COST | 5 | 7 | 3 | 1 | 6 | 3 |

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>F4</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
Low Correlation with F4

Selected Features

<table>
<thead>
<tr>
<th>Cost</th>
<th>F4</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
Low Correlation with F6

Correlation threshold: 0.8

Selected Features

<table>
<thead>
<tr>
<th></th>
<th>F4</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
Reducing Cost

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F3</th>
<th>F5</th>
<th>F4</th>
<th>F2</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Correlation threshold: 0.8

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>F4</th>
<th>F6</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>
Reducing Cost

Correlation threshold: 0.8

Selected Features | F4 | F6 | F1
---|---|---|---
Cost | 5 | 7 | 1
Correlated and Lower Cost

<table>
<thead>
<tr>
<th>COST</th>
<th>F1</th>
<th>F3</th>
<th>F5</th>
<th>F4</th>
<th>F2</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Correlation threshold: 0.8

Correlation: 0.85

<table>
<thead>
<tr>
<th>Selected Features</th>
<th>F4</th>
<th>F6</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>
Reducing Cost

Correlation threshold: 0.8

Selected Features | F4 | F3 | F1
---|---|---|---
Cost | 5 | 3 | 1
Loss Function

- How do we decide how good the feature is for predicting a cluster?

- Loss = E + αD
  - E: Prediction Error
  - D: Tree’s average depth
  - α = 0.01
Prediction

Program \rightarrow \text{T T} \rightarrow \text{Trace} \rightarrow \text{F E} \rightarrow \text{Feature vectors}

\text{FEATURE SELECTION}

\text{Feature Sets} \rightarrow \text{M L} \rightarrow \text{INSTRUMENT} \rightarrow \text{Instrumented Programs} \rightarrow \text{RUN} \rightarrow \text{Outputs}

\text{Model} \rightarrow \text{Evaluation} \rightarrow \text{Ground Truth} \rightarrow \text{Compare}
Prediction

• Use (new) Decision Trees over all chosen features

• Predict various horizons into the future
  • One Decision Tree for each horizon

\[
\begin{align*}
\text{current} & \quad P_{k=1} & \quad P_{k=2} & \quad \ldots & \quad P_{k=n} \\
\end{align*}
\]
12 features is enough for Python
Offline Evaluation: Depth

Tree depth of 10 is good for Python
System

Program → TT → Trace → FE → Feature vectors

FEATURE SELECTION

Feature Sets → ML → INSTRUMENT

Instrumented Programs → RUN → Outputs

Model → Evaluation

Ground Truth → Compare
System

• Counting occurrences of events: produces a feature vector

• Run Decision Tree model over the feature vector

• Record Results and Dump out

• Implement in Cpython
  • Trace module of the Python standard library
Run-time Overhead

Running time of the instrumented program

Program + Collecting features + Detection function

Baseline
## Run-time Overhead

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>overhead</td>
<td>0.7%</td>
</tr>
</tbody>
</table>
Predictor Quality

• How to trigger the prediction function?
• Problem: not counting all the branches
• Solution:
  • Choose the feature with smallest variation
  • Use it as a proxy for time
  • Choose the phase that is closest to the interpolated point to compare
Predictor Quality

• Definition
  • \( f_i = \sum 1[\varphi_t = i] \) : total number of phases labeled \( i \)
  • \( f_{i,j} = \sum 1[\varphi_t = i] 1[\varphi_{t+k} = j] \) : given current phase labeled \( i \), total number of phases labeled \( j \) in the future time step

• Compare our model with
  • Uniform: choosing \( \text{argmax}_i f_i \)
  • Random: choosing \( i \) with probability \( f_i / \sum_j f_j \)
  • CondUniform: \( \text{argmax}_j f_{i,j} \)
  • CondRandom: choosing \( j \) with probability \( f_{i,j} / \sum_j f_{i,j} \)
  • ProbSim: Iterate CondRandom for one step \( k \) times
DT is a good choice for these benchmarks.
Conclusion

• We built run-time future phase prediction systems for Python

• Low Cost - 0.7% overhead

• High quality prediction
Acknowledgments

- NSF Grant CCF-1320498

- Eliot Moss