Program Behavior Sequence Prediction

Bo Wu, Yunlian Jiang, Xipeng Shen  
(The College of William & Mary)  
Yaoqing Gao, Raul Silvera, Graham Yiu  
(IBM Toronto)
Outline

- Motivation
- Our perspectives
- Behavior sequence prediction framework
- Some results of loop trip count prediction
- Possible uses
- Summary
Motivation

- Accurate and proactive prediction of program behaviors is essential for many optimizations
  - Loop trip counts for loop unrolling
  - Function hotness for function optimization level in JIT
  - Profitability for speculative parallelization
  - Cache miss rates for prefetching aggressiveness
  - Loop coldness for outlining
  - ...
Motivation

- The usefulness is not limited to program optimizations
  - OS level
    - Provision in cloud computing
    - Scheduling to reduce resource contention
  - Architecture level
    - Voltage scaling
Motivation

- However, the prediction of program behaviors is challenging
Motivation

- Opportunities do exist

```c
int contains_underbar(char * s) {
    /* finding an underscore */
    while (*s != 0) {
        if (*s == '_') return TRUE;
        s++;
    }
}

int numberfy(char * s){
    /* finding a decimal point */
    Loop 2: for (; (*s != 0) && (*s != '.'); s++)
    {
    }
```

(a) loop 1 trip-count sequence

(b) loop 2 trip-count sequence
Our Perspectives

- Difference between instance prediction and sequence prediction
  - Instance prediction: the next one or several instances
  - Sequence Prediction: the whole sequence of the considered behavior
- Statistical correlation among different behaviors
  - Trip counts of two different loops
  - Loop trip counts and function hotness
- Context awareness
  - Loop stack and call stack
  - Correlated behaviors happened before
Our Perspectives

- Three requirements for behavior prediction
  - Accuracy
  - Proactivity
  - Scope

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Accuracy</th>
<th>Scope</th>
<th>Proactivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline profile-based pred</td>
<td>o</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>runtime instance pred</td>
<td>✔</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>goal of sequence pred</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
The initial study is on loop trip counts prediction
- Loops are dominant parts
- Resource requirements
- Inlining
- Computation granularity
- ...
Sequence Prediction Framework

- Loop trip count sequences follow patterns

(a) A loop in h264ref (line 1502 of mbuffer.c)

(b) A loop in h264ref (line 79 of memalloc.c)

(c) A loop in mcf (line 52 of pstart.c)
Sequence Prediction Framework

- Three steps
  - Simplification
    - Recognize the pattern of a sequence and use several features to represent it
  - Prediction
    - Predict the sequence features through correlation
  - Generation
    - Reconstruct sequences from the predicted features
Sequence Prediction Framework

- Pattern Recognition

A behavior sequence

- C1: a single primary pattern
  - C1.1: constant
  - C1.2: logarithm
  - C1.3: square root
  - C1.4: linear
  - C1.5: square

- C2: multiple phases
  - C2.1: non-repetitive phases
    - C2.1.1: each phase ∈ C1
    - C2.1.2: otherwise
  - C2.2: repetitive phases
    - C2.2.1: first phase ∈ C1
    - C2.2.2: first phase ∈ C2.1.1
  - C2.2.3: otherwise

- C3: others
  - C3.1: having a partner
  - C3.2: dominant values
  - C3.3: otherwise
Sequence Prediction Framework

- Pattern Recognition

(a) C1.4
(b) C2.1.1
(c) C2.1.2
(d) C2.2.1

\(<c1.4,1,1,17>\quad <c2.1.1,c1.4,10,31,-1,c1.1,18,23>\)

(e) C2.2.3
(f) C3.2
(g) C3.3
Sequence Prediction Framework

- Correlation Prediction

```c
// A: the training dataset
for each behavior b
    for each behavior b' that b'.id < b.id
        for each dimension d of b's pattern vector
            Let y be a vector containing all values of d of b in A
            Let X be a matrix containing all pattern vectors of b' in A
            Do regression: corRegress(y, X, err, model);
            if (err < minErr)
                minErr = err; b.partners[d] = b'; b.model[d] = model;
            end if
        end for
    end for
end for
```
Sequence Prediction Framework

- **Offline**
  - Training sequences ➔ Pattern recognition ➔ Pattern vectors ➔ Correlation analysis ➔ Predictive models

- **Online**
  - Program execution ➔ Runtime prediction ➔ Predicted pattern vectors ➔ Seq. generation ➔ Predicted sequences
Results

<table>
<thead>
<tr>
<th>Pred. Acc.</th>
<th>mcf</th>
<th>libquantum</th>
<th>bzip2</th>
<th>gobmk</th>
<th>hmmer</th>
<th>milc</th>
<th>parser</th>
<th>sphinx3</th>
<th>h264ref</th>
<th>sjeng</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday, November 10, 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Possible Uses

- Aggressive Optimizations
  - Loop unrolling for non-countable loops

```c
While(!p) {
    if(satisfySomeCondition(p)) {
        result = p;
        break;
    }
    else
        p = p->next;
}
```

- Need runtime check and recovery support
Possible Uses

- Loop parallelization

```plaintext
loop1 {
  loop2 {
    ...
    }
  }
}

loop2 {
  ...
  }
}
```

loop interchange?
Possible Uses

- From loop trip counts to other behaviors
  - Function hotness
  - Prefetching aggressiveness
  - Software pipelining
  - Trace selection in trace JIT
Summary

- Program behavior prediction is useful for many compiler optimizations, and even for OS and architecture level

- Behavior Sequences show extreme complexity, but correlation provides an opportunity to predict them

- Three requirements for useful predictions

- High prediction accuracy is possible for many loops
Thanks!