Exploiting Statistical Correlations for Proactive Prediction of Program Behaviors

Yunliang Jiang, Eddy Zhang, Kai Tian, Feng Mao, Malcom Getters, Xipeng Shen
CAPS Research Group, College of William and Mary, USA

Yaoqing Gao
IBM Toronto Lab, Canada
Program Optimizations

- Static
- Profile
- Dynamic

1950s 1980s 1990s
Prerequisite for Optimizations

Accurate prediction of how programs would behave.

Program Behaviors

(procedure calling freq, locality, loop trip counts...)

CAPS @ William & Mary
## Program Behavior Predictions

<table>
<thead>
<tr>
<th>Property</th>
<th>Accuracy</th>
<th>Scope</th>
<th>Timing (Proactivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Opt</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static Compilation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Profile Feedback</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runtime Adaptive Optimization</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Importance of Proactivity

while (...){
  foo ();
}

Reactive approach

inferior performance caused by local view-based optimizations

47% on J9 [Arnold+’ 05]
21% on JikesRVM [Mao+’ 09]
Adaptivity-Proactivity Dilemma

Adaptivity-Proactivity Dilemma

- **Dynamic Optimization**
- **Offline Prof.**
- **Static Compilation**
Prior Solution: Input-Based Prediction

[Mao+:CGO’09]

**Idea:** Predicting behavior from inputs as program starts

**Problem:** Requiring manual characterization of inputs
Our Solution

Exploit correlations among program components for proactive runtime prediction and optimization
```c
main(int argc, char * argv){
    mesh_init (dataFile, mesh, refMesh);
    genMesh (mesh, 0, mesh->vN);
    verify (mesh, refMesh);
}

Mesh * mesh_init (char * initInfoF, Mesh* mesh, Mesh* refMesh) {
    // open vertices file, read # of vertices
    FILE * fdata = fopen (initInfoF, "r");
    fscanf (fdata, "%d, %
", &vN);
    mesh->vN = vN;
    v = (vertex*) malloc (vN*sizeof(vertex));
    // read vertices positions
    for (i=0; i<vN; i++) {
        fscanf (fdata, "%f %f
", &v[i].x, &v[i].y);
    }
    // sort vertices by x and y values
    for (i=1; i< vN; i++) {
        for (j=vN-1; j>=i; j--)
            if (v[j].x < v[i].x) {
                // swap vertices
            }
    }
    // read edges into refMesh for later verification
}

void genMesh (Mesh *m, int left, int right){
    if (right>3+left){
        genMesh (m, left, (left+right)/2);
        genMesh (m, (left+right)/2+1, right);
    }
    ...}

void verify (Mesh *m, Mesh *mRef){
    for (i=0, j=0; i< m->edgesN; i++){
        for (j=vN; j>=i; j--){
            ...}
        }
    // read edges into refMesh for later verification
}
```
Questions to Answer

- Do such correlations exist commonly?
- How can they be automatically identified?
- Are they useful for program optimizations?
Outline

• A systematic measurement of correlations
• A framework for identification and modeling
• A demonstration of uses for optimizations
• Related work and conclusion
Behaviors under Study

- Loop trip-counts (by a modified GCC)
- Procedure calling frequencies (by GNU gpof v2.19)
- Block access freq. (data profiles) (by IBM XL C 10.1)
- Edge profiles and node profiles (by IBM XL C 10.1)

Correlations to Measure

- Among same types of behaviors of different components
  - E.g. trip-counts of two loops
- Among different types of behaviors of different components
  - E.g. trip-counts vs procedure calling freq.
## Benchmarks [spec 2000 & 2006]

<table>
<thead>
<tr>
<th>name</th>
<th>Program lines</th>
<th>inputs</th>
<th>loops</th>
<th>Factor of changes caused by inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ammp</td>
<td>13263</td>
<td>20</td>
<td>425</td>
<td>$9.9 \times 10^1$</td>
</tr>
<tr>
<td>art</td>
<td>1270</td>
<td>108</td>
<td>101</td>
<td>$4.0 \times 10^4$</td>
</tr>
<tr>
<td>crafty</td>
<td>19478</td>
<td>14</td>
<td>425</td>
<td>$4.6 \times 10^8$</td>
</tr>
<tr>
<td>equake</td>
<td>1513</td>
<td>100</td>
<td>106</td>
<td>$1.0 \times 10^2$</td>
</tr>
<tr>
<td>gap</td>
<td>59482</td>
<td>12</td>
<td>1887</td>
<td>$1.1 \times 10^8$</td>
</tr>
<tr>
<td>gcc</td>
<td>484930</td>
<td>72</td>
<td>7615</td>
<td>$1.1 \times 10^6$</td>
</tr>
<tr>
<td>gzip</td>
<td>7760</td>
<td>100</td>
<td>223</td>
<td>$4.3 \times 10^7$</td>
</tr>
<tr>
<td>h264ref</td>
<td>46152</td>
<td>20</td>
<td>2074</td>
<td>$2.1 \times 10^9$</td>
</tr>
<tr>
<td>lbm</td>
<td>875</td>
<td>120</td>
<td>27</td>
<td>$6.0 \times 10^6$</td>
</tr>
<tr>
<td>mcf</td>
<td>1909</td>
<td>64</td>
<td>76</td>
<td>$1.4 \times 10^5$</td>
</tr>
<tr>
<td>mesa</td>
<td>50230</td>
<td>20</td>
<td>995</td>
<td>$2.0 \times 10^1$</td>
</tr>
<tr>
<td>milc</td>
<td>12837</td>
<td>10</td>
<td>473</td>
<td>$2.1 \times 10^9$</td>
</tr>
<tr>
<td>parser</td>
<td>10924</td>
<td>20</td>
<td>1350</td>
<td>$2.1 \times 10^6$</td>
</tr>
<tr>
<td>vpr</td>
<td>16976</td>
<td>20</td>
<td>435</td>
<td>$3.9 \times 10^6$</td>
</tr>
</tbody>
</table>

[Thanks to Amaral’s group for extra inputs]
Calculation of Correlations

\[ r_{XY} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{(n - 1)s_X s_Y} \]

Sample standard deviation

The higher \( r \) is, the easier to predict one from the other.
Strong correlations from loops to loops and to other behaviors

Uses for runtime behavior prediction
Outline

• A systematic measurement of correlations

• A framework for identification and modeling

• A demonstration of uses for optimizations

• Related work and conclusion
Two Goals

• Identify Seminal Behaviors

• Build predictive models

\[ \text{Target behavior value} = f(\text{values of seminal behaviors}) \]
Seminal Behaviors

- A small set of program behaviors
  - Predictive capability
    - Strongly correlate with target behaviors
  - Earliness
    - Values become known early in an execution
Identification of Sem Beh

Prog & inputs → Behavior collection → value sets of candidates
Candidate Seminal Behaviors

• Interface behaviors
  • Values directly obtained from program inputs
  • Ignore massive file content
    • Include corresponding loop trip-counts

• Loop trip-counts
  • Importance in programs and strong correlations with other behaviors
Recognition of Sem Beh

Prog & inputs → Behavior collection → value sets of candidates

→ Affinity list construction

→ Affinity lists
Behavior Affinity List

Header can predict body accurately.
Affinity List of \textit{mcf}

- Incremental construction
  - Start with interface behaviors
  - Iteratively find headers from remaining based on their \textit{predictive capability}
Predictive Capability

• Predictive models
  • LMS (Least Mean Square)
  • Regression Trees

• Compute predictive capability
  • 10-fold cross-validation
Refine by *predictive capability* & *earliness*

**Seminal Behaviors**
Seminal Behavior Based Predict

Num of seminal behaviors and prediction accuracy

<table>
<thead>
<tr>
<th>Prog</th>
<th>interface values</th>
<th>earliness ≥ 90%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>num</td>
<td>accuracy</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>call</td>
</tr>
<tr>
<td>ammp</td>
<td>199.5</td>
<td>96.7</td>
</tr>
<tr>
<td>art</td>
<td>491.0</td>
<td>96.8</td>
</tr>
<tr>
<td>crafty</td>
<td>189.9</td>
<td>58.9</td>
</tr>
<tr>
<td>equake</td>
<td>198.0</td>
<td>100</td>
</tr>
<tr>
<td>gap</td>
<td>297.5</td>
<td>44.9</td>
</tr>
<tr>
<td>gcc</td>
<td>482.9</td>
<td>38.9</td>
</tr>
<tr>
<td>gzip</td>
<td>392.2</td>
<td>87.0</td>
</tr>
<tr>
<td>h264ref</td>
<td>399.8</td>
<td>99.8</td>
</tr>
<tr>
<td>ibm</td>
<td>399.8</td>
<td>90.1</td>
</tr>
<tr>
<td>mcf</td>
<td>587.3</td>
<td>87.7</td>
</tr>
<tr>
<td>mesa</td>
<td>1100</td>
<td>100</td>
</tr>
<tr>
<td>milc</td>
<td>279.2</td>
<td>72.1</td>
</tr>
<tr>
<td>parser</td>
<td>190.2</td>
<td>85.4</td>
</tr>
<tr>
<td>vpr</td>
<td>393.3</td>
<td>95.1</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>2.4</td>
<td>92.9</td>
</tr>
</tbody>
</table>
Outline

• A systematic measurement of correlations

• A framework for identification and modeling

• A demonstration of uses for optimizations

• Related work and conclusion
Performance Improvement

(Baseline: highest static opt)

IBM Power5
XL 11.1
More Potential Uses

- Help JIT compilers make better decisions in managed environment
  - i.e. JVMs

- Boost performance through dynamic code version selection
  - for imperative languages such as C
Outline

• A systematic measurement of correlations

• A framework for identification and modeling

• A demonstration of uses for optimizations

• Related work and conclusion
Related Work

• Correlations between control flow signatures and hardware performance
  • [Sherwood+:ASPLOS’02, Annavaram+:Micro 04, etc.]

• Adaptive dynamic optimization
  • [Arnold+:OOPSLA’00, Chen+:PLDI’06, Lau+:PLDI’06, etc.]

• Exploiting inputs for optimization
  • [Wang+:PLDI’04, Mao+:CGO’09, Chen+:PLDI’10]
Conclusion

- Strong correlations exist among behaviors.
- Seminal behavior-based technique is promising.
- Significant potential for program optimizations.
Thanks!

Questions?

Kai Tian
ktian@cs.wm.edu