An Input-Centric Paradigm for Program Dynamic Optimizations

Kai Tian, Yunlian Jiang, Eddy Zhang,
Xipeng Shen (presenter)

College of William and Mary

(Published in OOPSLA’10)
Program Optimizations

Static 1950s
Profile 1980s
Dynamic 1990s
Dynamic Optimizations

Widely used in Java, C#, etc.
Current Limitations

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

Runtime 
overhead

47% on J9 [Arnold+’ 05]
21% on JikesRVM [Mao+’ 09]

Reactive scheme

Inferior performance 
caused by 
local view-based 
optimizations

College of William & Mary
Adaptivity-Proactivity Dilemma

- Dynamic optimization
- Input-centric paradigm
- Offline profiling
- Static compilation
Outline

• Why input-centric?

• Input-centric paradigm

• Evaluation

• Related work

• Conclusion
Prerequisite for Optimizations

Accurate prediction of how programs would behave.

Program Behaviors

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

College of William & Mary
What Decide Program Behavior?

Prog Beh = Code + Inputs + Running Environments

only deciding factor given a program on a machine

- Command-line arguments
- Interactively input data
- Input files
- ...

College of William & Mary
Idea: Use program inputs to trigger runtime behavior prediction and proactive optimizations

Proactivity: Early optimize based on prediction

Adaptivity: Input-specific optimization
Benefits for JIT

- JIT in JikesRVM

-1 0 1 2
deeper optimization
larger overhead

* Thanks to coghillcartoning.com for the image.
Challenges

- Complexities in inputs
- Complexities in relations
- Integration in runtime
Techniques to Realize Input-Centric Paradigm
Input Characterization

● **Goal**

raw input

input feature vector

< feature 1, feature 2, ..., feature k >

● **Solution**

● Seminal Behaviors [Jiang+: CGO’10]
  
  ● Exploit strong correlations among program behaviors
main(int argc, char * argv) {
    ...
    mesh_init (dataFile, mesh, refMesh);
    genMesh (mesh, 0, mesh->vN);
    verify (mesh, refMesh);
}

// recursive mesh generation
void genMesh (Mesh *m, int left, int right) {
    if (right > left + 3) {
        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++) {
        ...
    }
}

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--){
            ...
        }
    }
    while (!feof(fd)) {
        ...
    }
    // read edges into refMesh for later verification
}

Seminal Behaviors
Seminal Behaviors Identification

- Through statistical learning
- Fully automatic framework
- Details in [Jiang+:CGO’10].
Techniques to Realize Input-Centric Paradigm
Input Behavior Modeling

● Problem formulation
  ● To construct predictive models
    ● Target Behaviors = f (Seminal Behaviors)

● Solution: Cross-run machine learning
  ● Target beh. is categorical (e.g., opt. levels)
    ● Classification Trees
  ● Target beh. is numerical (e.g., calling freq.)
    ● Linear Regression (LMS)
    ● Regression Trees
Special Challenges

- Categorical vs. numerical features
  - Data types
  - Number of unique values in training data sets

- Feature selection
  - Classification & regression trees
    - Filter out unimportant features automatically
  - LMS regression
    - PCA (when all features numerical)
      - Select directions showing large variations
    - Stepwise selection (otherwise)
      - Continuously add features that improve prediction
Risk Control

- Prevent effects of wrong predictions
  - Fine-grained discriminative prediction

Keep assessing confidence level of each input subspace;
if (confidence_level > Threshold)
    Do prediction;
else
    Fall back to default reactive strategy;

Details in [Tian+:OOPSLA’10].
Techniques to Realize Input-Centric Paradigm

input-centric adaptation

...
Evaluation 1: JikesRVM opt

- **Machine**
  - Intel Xeon E5310, Linux 2.6.22

- **Java Runtime**
  - Modified JikesRVM 3.1.0

- **Benchmarks**
  - 10 Java programs from Dacapo, Grande, JVM98

- **Inputs**
  - Extra inputs from [Mao+:CGO’09]
## Prediction Accuracy for Java

<table>
<thead>
<tr>
<th>Program</th>
<th># of inputs</th>
<th># of sem.beh.</th>
<th>Prediction accuracy</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Compress</td>
<td>20</td>
<td>2</td>
<td>0.99</td>
<td>0.93</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>Db</td>
<td>54</td>
<td>4</td>
<td>0.84</td>
<td>0.98</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>Mtrt</td>
<td>100</td>
<td>2</td>
<td>0.97</td>
<td>0.89</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>Antlr</td>
<td>175</td>
<td>39</td>
<td>0.95</td>
<td>0.95</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>Bloat</td>
<td>100</td>
<td>7</td>
<td>0.96</td>
<td>0.76</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>Euler</td>
<td>14</td>
<td>1</td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>MolDyn</td>
<td>15</td>
<td>2</td>
<td>0.98</td>
<td>0.83</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>MonteCarlo</td>
<td>14</td>
<td>1</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td>9</td>
<td>2</td>
<td>0.99</td>
<td>0.97</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>RayTracer</td>
<td>12</td>
<td>1</td>
<td>0.98</td>
<td>0.9</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>51.3</strong></td>
<td><strong>6.1</strong></td>
<td><strong>0.96</strong></td>
<td><strong>0.92</strong></td>
<td><strong>0.97</strong></td>
<td></td>
</tr>
</tbody>
</table>

10-fold cross-validation
Speedup in JikesRVM

Baseline: default JikesRVM

College of William & Mary
Evaluation 2: Dynamic Version Selection

- Input-centric adaptation
  - Models from inputs to suitable versions
  - Predict the best version to run in a new execution

- Reactive approach [Chuang+:07]
  - Timing each version and use the best for the remaining execution
Experiment Setting

● Versions creation
  ● IBM XL C compiler
  ● 5 code versions from feedback-driven opt

● Machines
  ● IBM Power5, AIX 5.3

● Benchmarks
  ● 14 C programs from SPEC2000 & SPEC2006

● Inputs
  ● 10--120
  ● Some from University of Alberta (J. N. Amaral’s group)
  ● Others collected or created by us
Speedup by Version Selection

Baseline: static compilation at highest opt level
### Discussions

- **Three steps for input-centric optimizations**
  - Profile collection *(offline)*
  - Seminal beh recog. & input-beh model construction *(offline)*
  - Proactive behavior prediction & optimizations *(online)*

- **Input-centric paradigm is fundamental**
  - May benefit many other optimizations
    - Anywhere runtime adaptation is needed

- **Not conflict with phase changes**
- **Complement to reactive dynamic optimizations**
Related Work

• Phase-based adaptive recompilation
  • [Gu & Verbrugge: CGO’08]

• Benchmark design
  • [Berube & Amaral: SPEC’07]

• Library development
  • ATLAS [Whaley+:+01], Sorting [Li+:CGO04], FFTW [Frigo+: IEEE’05], SPIRAL [M. Puschel+: IEEE’05], STAPL [Thomas+: PPOPP’05]

• General-purpose programming
  • Seminal behavior exploration [Jiang+: CGO’10]
  • Specification language (XICL) to capture input features [Mao+: CGO’09]
Conclusions

input-centric adaptation
input-behavior modeling
input characterization

dynamic optimization
Input-centric dyn opt
offline prof.
static compilation

Adaptivity vs. Proactivity

Static
Profile
Dynamic
Input Centric
Thanks!
Potential Speedup in Version Selection

Baseline: static compilation at highest opt level
<table>
<thead>
<tr>
<th>Name</th>
<th># of lines of code</th>
<th># of inputs</th>
<th>factors of changes caused by inputs</th>
<th># of sem. beh.</th>
<th>accuracy edge</th>
<th>node</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ammp</td>
<td>13263</td>
<td>20</td>
<td>$9.9 \times 10^1$</td>
<td>1</td>
<td>100</td>
<td>91.1</td>
<td>99.7</td>
</tr>
<tr>
<td>art</td>
<td>1270</td>
<td>108</td>
<td>$4.0 \times 10^4$</td>
<td>4</td>
<td>100</td>
<td>80.0</td>
<td>96.1</td>
</tr>
<tr>
<td>crafty</td>
<td>19478</td>
<td>14</td>
<td>$4.6 \times 10^8$</td>
<td>2</td>
<td>90.8</td>
<td>44.5</td>
<td>79.3</td>
</tr>
<tr>
<td>equake</td>
<td>1513</td>
<td>100</td>
<td>$1.0 \times 10^2$</td>
<td>1</td>
<td>100</td>
<td>96.3</td>
<td>99.3</td>
</tr>
<tr>
<td>gap</td>
<td>59482</td>
<td>12</td>
<td>$1.1 \times 10^8$</td>
<td>7</td>
<td>56.3</td>
<td>69.7</td>
<td>88.5</td>
</tr>
<tr>
<td>gcc</td>
<td>484930</td>
<td>72</td>
<td>$1.1 \times 10^6$</td>
<td>54</td>
<td>93.6</td>
<td>95.4</td>
<td>95.6</td>
</tr>
<tr>
<td>gzip</td>
<td>7760</td>
<td>100</td>
<td>$4.3 \times 10^7$</td>
<td>6</td>
<td>83.5</td>
<td>69.0</td>
<td>94.5</td>
</tr>
<tr>
<td>h264ref</td>
<td>46152</td>
<td>20</td>
<td>$2.1 \times 10^9$</td>
<td>4</td>
<td>97.0</td>
<td>97.8</td>
<td>99.7</td>
</tr>
<tr>
<td>ibm</td>
<td>875</td>
<td>120</td>
<td>$6.0 \times 10^6$</td>
<td>3</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>mcf</td>
<td>1909</td>
<td>64</td>
<td>$1.4 \times 10^5$</td>
<td>10</td>
<td>100</td>
<td>89.5</td>
<td>97.5</td>
</tr>
<tr>
<td>mesa</td>
<td>50230</td>
<td>20</td>
<td>$2.0 \times 10^1$</td>
<td>1</td>
<td>99.5</td>
<td>12.2</td>
<td>100</td>
</tr>
<tr>
<td>milc</td>
<td>12837</td>
<td>10</td>
<td>$2.1 \times 10^9$</td>
<td>18</td>
<td>100</td>
<td>52.0</td>
<td>99.7</td>
</tr>
<tr>
<td>parser</td>
<td>10924</td>
<td>20</td>
<td>$2.1 \times 10^6$</td>
<td>2</td>
<td>79.2</td>
<td>78.0</td>
<td>90.8</td>
</tr>
<tr>
<td>vpr</td>
<td>16976</td>
<td>20</td>
<td>$3.9 \times 10^6$</td>
<td>9</td>
<td>64.0</td>
<td>82.2</td>
<td>95.8</td>
</tr>
</tbody>
</table>