An Input-Centric Paradigm for Program Dynamic Optimizations

Kai Tian, Yunlian Jiang,
Eddy Zhang, Xipeng Shen

College of William and Mary
Program Optimizations

1950s 1980s 1990s

Static
Profile
Dynamic

Kai Tian @ William & Mary
Dynamic Optimizations

Widely used in Java, C#, etc.

Observe → Behavior → Optimize

Runtime system

Kai Tian @ William & Mary
Drawback of Dynamic Opt

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

Inferior performance caused by local view-based optimizations

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

- **Adaptivity**
  - Dynamic optimization
  - Offline profiling

- **Proactivity**
  - Input-centric paradigm
  - Static compilation
Outline

- Why input-centric?
- Input-centric paradigm
- Evaluation
- Related work
- Conclusion
Accurate prediction of how programs would behave.

Program Behaviors

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

Prerequisite for Optimizations
What Decide Program Behavior?

\[ \text{Prog Beh} = \text{Code} + \text{Inputs} + \text{Running Environments} \]

- Command-line arguments
- Interactively input data
- Input files
- ...
Input-Centric Opt Paradigm

**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

input

opt(-1)
foo()
foo()
opt(0)
foo()
foo()
opt(1)

input

1
opt(1)
foo()
foo()
foo()
foo()
Challenges

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

- **Goal**
  
- **Solution**
  - Seminal Behaviors [Jiang+: CGO’10]
    - Exploit strong correlations among program behaviors

input feature vector
< feature 1, feature 2, ..., feature k >
```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, %\n", &vN);
    mesh->vN = vN;
    v = (vertex*) malloc (vN*sizeof(vertex));
    // read vertices positions
    for (i=0; i<vN; i++) { fscanf (fdata, "%f %f\n", &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--){ ...
    }...}

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

    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++){
            ...
        }
    }
```
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 = \( f(\text{Seminal Behaviors}) \)

• Solution: Cross-run incremental learning
  ● Target is categorical beh. (e.g., opt levels)
    ● Classification Trees
  ● Target is numerical beh. (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

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

- Model evolvement
  - Behavior models / confidence values evolve on new runs
  - Model retrained using expanded data set
Techniques to Realize Input-Centric Paradigm

input-centric adaptation

input-behavior modeling

input-characterization
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></td>
<td></td>
<td></td>
<td>opt level</td>
<td>call freq</td>
<td>min heap</td>
<td></td>
</tr>
<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>
Speedup in JikesRVM

Baseline: default JikesRVM

![Graph showing speedup in JikesRVM with various benchmarks and metrics like Min, Mean, and Max.](image-url)
Evaluation 2: Dynamic Version Selection

- **Input-centric adaptation**
  - Model from input features to suitable versions
  - Predict best version to run during runtime

- **Reactive approach [Chuang+:07]**
  - Timing each version to select the best during runtime
Experiment Setting

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

- Machines
  - IBM Power5
  - AIX 5.3
Speedup in Version Selection

Baseline: static compile with highest opt level

Runtime Reactive
Input-centric
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

• 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]

• Connections among program behaviors
  • Between loops and method hotness [Namjoshi+: VEE’10]
Conclusions
Thanks!
Questions?

Kai Tian
ktian@cs.wm.edu