Pin: Building Customized Program Analysis Tools with Dynamic Instruction

Chi-Kueng Luk, et al.
Intel Corporation
University of Colorado
Presenter: Osama Minhas
Summary

- Built an Instrumentation System
  - **Instrumentation**: Technique for inserting extra code into an application to observe its behavior

- **Design**:
  - easy-to-use
  - portable
  - transparent
  - efficient
  - robust
Solution: Pin

- Architecture independent API
- Uses just-in-time compilation
- Instrumentation tools (called Pintools) built with Pin’s API
- Significantly better instrumentation performance than similar tools
Background

- **Goal**: Provide an instrumentation platform for building program analysis tools
- Needs to fulfill the main design principles: Ease-of-use, portability, transparency, efficiency, robustness
Before Pin

- Valgrind, DynamoRIO
  - Without instrumentation, these tools are great
  - Require tool writer to do certain manual tasks

- Pin
  - Built to be used with instrumentation
  - Fully automated approach to instrumentation
Pin software architecture

- Consists of
  - Virtual machine
  - Code cache
  - Instrumentation API
About Pin

- A type of JIT compiler
  - Input is an executable
  - Pin intercepts first instruction and generates new code for the instruction
  - Transfers control to generated sequence
  - Pin regains control after and moves to next instruction

- Code Reuse
- User can inject their own code (instrumentation!)
Pintools

- Consists of **Instrumentation** and **Analysis** Code
  - **Instrumentation**: Called from Pin when new code generated
  - **Analysis**: Gathers data about application
- Can modify Pin’s code generation process
- Shares address space with Pin and executable
<table>
<thead>
<tr>
<th>Hardware</th>
<th>Linux</th>
<th>Compiler</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>IA32 1.7GHz Xeon™, 256KB L2</td>
<td>2.4.9</td>
<td>gcc 3.3.2, -O3 for SPECint (except</td>
<td>Shared</td>
</tr>
<tr>
<td>cache, 2GB Memory</td>
<td></td>
<td>in eon where we use icc)</td>
<td></td>
</tr>
<tr>
<td>EM64T 3.4GHz Xeon™, 1MB L2</td>
<td>2.4.21</td>
<td>icc 8.0 for SPECfp</td>
<td>Static</td>
</tr>
<tr>
<td>cache, 4GB Memory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Itanium® 1.3GHz Itanium®2, 6MB</td>
<td>2.4.18</td>
<td>Intel ® compiler (icc 8.0), with</td>
<td>Static</td>
</tr>
<tr>
<td>L2 cache, 12GB Memory</td>
<td></td>
<td>interprocedural &amp; profile-guided</td>
<td></td>
</tr>
<tr>
<td>ARM 400 MHz XScale®, 80200, 128</td>
<td>2.4.18</td>
<td>gcc 3.4.1, -O2</td>
<td>Static</td>
</tr>
<tr>
<td>MB Memory</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Experimental setup.
Experimental Evaluation

- Pin without instrumentation
  - IA32 and EM64T run-time overhead
    - 60% overhead for integer
    - 5% overhead for floating point
  - Itanium: same trend but generally larger
  - ARM: Worse than others because indirect linking not implemented and fewer resources

- Compilation time reflected in run time
Experimental Evaluation

- Pin with instrumentation
  - Basic-block counting
  - Without optimization, overhead fairly large
  - Adding inlining helps significantly
  - eflags liveness analysis reduces slowdown
  - Optimizations together reduce slowdown from 10.4x to 2.8x for integer and 3.9x to 1.4x for floating point
Figure 6. Performance of Pin with basic-block counting instrumentation on the IA32 architecture.
Comparison with Valgrind and DynamoRIO

- **Without** instrumentation, Pin and DynamoRIO significantly outperform Valgrind
  - DynamoRIO 12% faster than Pin
- **With** instrumentation, Pin significantly outperforms both DynamoRIO and Valgrind
  - Valgrind slows application by 8.3x
  - DynamoRIO by 5.1x
  - Pin by 2.5x
Limitations (At time of Article)

- Pin is **not strictly better** than other program analysis tools
- Shown and tested on IA32, EM64T, Itanium and ARM Architectures
- Only uses basic-block counting for instrumentation
- Linux only
- **Limited** building heavyweight tools
Progress

- Pin now works for Intel 64 and Intel Many Integrated Core architectures
- Linux, macOS, and Windows OS now supported
Impact

- 4275 Citations
- Used to develop and test a variety of tools (McPAT, Rodinia, etc.)
- Still in use and updated today
References

Thank You! Any Questions?