Performance Diagnosis for Inefficient Loops

*Linhai Song*\(^1\) and Shan Lu\(^2\)

\(^1\)FireEye, Inc.

\(^2\)University of Chicago
What are Performance Problems?

• Definition of Performance Problems (PPs):
  – Implementation mistakes causing inefficiency

• An example

```c
char * ssh_xph_generate(node_t * aNode)
{
    int count = 0;
    for (n = aNode; n; n = aNode->prev)
        if(n->name == aNode->name)
            count ++;
    return count;
} //Mozilla#477564
```
What are Performance Problems?

• Definition of Performance Problems (PPs):
  – Implementation mistakes causing inefficiency
• An example

```c
char * ssh_xph_generate(node_t * aNode)
{
    int count = 0;
    for (n = aNode; n; n = n->prev)
        if(n->name == aNode->name)
            count ++;
    return count;
} //Mozilla#477564
```
What are Performance Problems?

• Definition of Performance Problems (PPs):
  – Implementation mistakes causing inefficiency

• An example

```c
char * ssh_xph_generate(node_t * aNode) {
    int count = 0;
    for (n = aNode; n; n = aNode->prev)
        if(n->name == aNode->name)
            count ++;
    return count;
} //Mozilla#477564
```
Fighting PPs is Important

• PPs widely exist in production run software
  – 5 to 60 Mozilla PPs are fixed each month
• PPs are getting more important
  – Hardware is not getting faster (per-core)
  – Software is getting more complex
  – Energy saving is getting more urgent
• Identifying the causes of inefficient execution
  – Different from testing & bug detection
  – In-house and on-line diagnosis
Diagnosing PPs is Challenging

• Three criteria for diagnosis tools
  – Coverage: handle a good portion of PPs
  – Accuracy: accurately tell
    • Which code regions are inefficient
    • Why they are inefficient
  – Performance: low run-time overhead
State of the Art

• No existing tools can satisfy the three criteria
Statistical Debugging

LDoctor

- A tool(kit) targets inefficient loops

Statistical Debugging[1]

Targets of LDoctor

Inefficient Loops (45/65)

```java
if (true) {
    ....
}
while (...)
{
    ....
}
func(...)
{
    ....
    func();
}
```
Contributions

• A root-cause taxonomy for inefficient loops
  – Guide the design of automated diagnosis tools
  – Resultless vs. redundant

• LDoctor: a toolkit to diagnose inefficient loops
  – Hybridize static and dynamic analysis
  – Identify inefficiency root causes
    • With high coverage, high accuracy, and low overhead
Outline

• Overview
• Root-cause Taxonomy
• LDoctor Design
• Evaluation of LDoctor
• Conclusion
Outline

- Overview
- Root-cause Taxonomy
- LDoctor Design
- Evaluation of LDoctor
- Conclusion
What Type of Taxonomy We Need?

• Inclusive: covering most/all inefficient loops

• General: Not application specific

• Actionable: helpful to design fix
Root-Cause Taxonomy

Inefficient Loops

15
Resultless Loops

• Produce no side effects
• Never produce results in any iteration
Resultless 0*1?

• Only produce results in the last iteration
Resultless [0|1] *

- May produce results in each iteration

Diagram:
- Inefficient Loops
  - Resultless
  - [0|1]*
    - Iteration without Side Effect
    - Iteration with Side Effect

[0|1]* Loop Instance
• Produce results in almost all iterations
Redundant Loops

- Produce already-available results
Cross-loop Redundancy

char * ssh_xph_generate(node_t * aNode)
{
    int count = 0;
    for (n = aNode; n; n = aNode->prev)
        if(n->name == aNode->name)
            count ++;
    return count;
} //Mozilla#477564
Cross-iteration Redundancy

Loop Instance

X X X X ... X X

Iteration with same side effect

Inefficient Loops

Redundancy

Cross-iteration

Redundancy
Root-Cause Taxonomy

Inefficient Loops

Resultless
- 0*
- 0*1?
- [0|1]*
- 1*

Redundancy
- Cross-loop
- Cross-iteration
Outline

• Overview
• Root-cause Taxonomy
• LDoctor Design
• Evaluation of LDoctor
• Conclusion
Design Principles

• Taxonomy guided design

• Focused checking

• Static-dynamic hybrid analysis
char * ssh_xph_generate(node_t * aNode) {
    int count = 0;
    for (n = aNode; n; n = aNode->prev) {
        if (n->name == aNode->name) {
            count ++;
            Mozilla#477564
        }
    }
    // called for every node in a list
}

C.-L. Redundancy Analysis (I)

Identify Side-effect Instructions
Static Slicing to Source Instructions
Record & Compare Source Value

Side-effect Instruction
Source Instruction
C.-L. Redundancy Analysis (II)

- Identify Side-effect Instructions
- Static Slicing to Source Instructions
- Record & Compare Source Value

Static

Dynamic

PreHeader
Loop
PostHeader

Instrumented Loop

Unsampled instances
Sampled Instances
Outline

• Overview
• Root-cause Taxonomy
• LDoctor Design
• Evaluation of LDoctor
• Conclusion
Implementation & Evaluation

• Implementation using LLVM-3.4.2

• Experimental setting
  – Evaluating 39 bugs from two benchmark suite
  – Applying on look rank list from SD
  – Sampling rate: 1/100 (C.L.) & 1/1000 (C.I.)

• Evaluation metrics

Coverage
Accuracy
Performance
## Coverage & Accuracy

<table>
<thead>
<tr>
<th>BugID</th>
<th>Reported Root Cause</th>
<th>Fix Suggestion</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla347306</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Mozilla490742</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Mozilla35294</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Mozilla477564</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MySQL27287</td>
<td>✓</td>
<td>✗</td>
<td>01</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Apache32546</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Apache37184</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Apache29742</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Apache34464</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Apache47223</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

*Coverage*
## Coverage & Accuracy

<table>
<thead>
<tr>
<th>BugID</th>
<th>Reported Root Cause</th>
<th>Fix Suggestion</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla347306</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla416628</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla490742</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla35294</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla477564</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>✓</td>
<td>✗</td>
<td>01</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Apache32546</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Apache37184</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Apache29742</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Apache34464</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Apache47223</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Performance

<table>
<thead>
<tr>
<th>BugID</th>
<th>LDoctor</th>
<th>w/o sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Resultless</td>
<td>C-L R.</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>1.00</td>
<td>1.18</td>
</tr>
<tr>
<td>Mozilla416628</td>
<td>1.00</td>
<td>1.04</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>1.05</td>
<td>1.00</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>-</td>
<td>1.00</td>
</tr>
<tr>
<td>GCC46401</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>GCC1687</td>
<td>-</td>
<td>/</td>
</tr>
<tr>
<td>GCC27733</td>
<td>1.00</td>
<td>/</td>
</tr>
<tr>
<td>GCC8805</td>
<td>-</td>
<td>1.00</td>
</tr>
<tr>
<td>GCC21430</td>
<td>-</td>
<td>1.04</td>
</tr>
<tr>
<td>GCC12322</td>
<td>-</td>
<td>1.04</td>
</tr>
</tbody>
</table>
Conclusions

• A root-cause taxonomy for inefficient loops
  
  Our taxonomy are general and specific!

• A series of static-dynamic hybrid analysis
  
  Achieve good coverage, accuracy and performance
Thanks a lot!
Inefficient Loops

Resultless

- \(0^*\)
- \(0^*1?\)
- \([0|1]^*\)
- \(1^*\)

Redundancy

- Cross-loop
- Cross-iteration
Resultless Analysis

Identify Side-effect Instructions
- 0*
- 0*1?
- 1*

Instrument Counters

Calculate Resultless Ratio

with side effect
without side effect

Loop

Static
- 0*
- [0|1]*

Dynamic
- [0|1]*
- 1*

0*1?