Context-Sensitive Delta Inference for Identifying Workload-Dependent Performance Bottlenecks

Xusheng Xiao¹, Shi Han², Dongmei Zhang², Tao Xie¹,³

¹North Carolina State University
²Microsoft Research Asia
³University of Illinois at Urbana-Champaign

¹xxiao2@ncsu.edu, ²{shihan, dongmeiz}@microsoft.com, ¹³taoxie@illinois.edu
A webpage is not responding on the following website: facebook.com

You can wait for the webpage to respond, or choose one of the following options:

- Recover this page
- Close this page
This program is not responding.

To return to Windows and check the status of the program, click Cancel.

If you choose to end the program immediately, you will lose any unsaved data. To end the program now, click End Now.
### Windows Task Manager

<table>
<thead>
<tr>
<th>Image Name</th>
<th>User Name</th>
<th>CPU</th>
<th>Memory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskhost.exe</td>
<td>Admin...</td>
<td>00</td>
<td>936 K</td>
<td>Host Proc...</td>
</tr>
<tr>
<td>wuaudt.exe</td>
<td>Admin...</td>
<td>00</td>
<td>180 K</td>
<td>Windows ...</td>
</tr>
<tr>
<td>dwm.exe</td>
<td>Admin...</td>
<td>00</td>
<td>8,220 K</td>
<td>Desktop ...</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>Admin...</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogMeInSyst...</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SynTPEnh.exe</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>igfxtray.exe</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMIGuardian...</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hkcmd.exe</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>igfxsrvc.exe</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>igfxpers.exe</td>
<td>Admin...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Open File Location**
  - End Process
  - End Process Tree
  - Debug
  - UAC Virtualization
  - Create Dump File
Performance Problems

• Widely exist in released software
  – Mozilla developers fixed 5–60perf bugs monthly over the past 10 years [Jin et al. PLDI 12]

Software Hangs in daily tools: file managers, office tools, browsers, ...
Software Hangs

• Three major categories
  – Correctness, e.g., infinite loops
  – Blocking operations, e.g., sending files
  – Expensive operations

With root of disk (C:\) selected in drop-down path selector, attempting to enable Flat View under the top-level View menu causes 7-Zip to hang ...

The process had to be forcibly stopped using Windows Task Manager.

contribute to 27% of 233 hang bugs

[Song et al. DSN 2010]
Workload-Dependent Performance Bottlenecks (WDPB)

- Expensive operations depending on input workloads, e.g., data processing

**Insight:** caused often by **workload-dependent loops** with **expensive** operations, e.g.,
  - Temp-object creation/destruction
  - File I/O
  - UI updates

- Fixed by
  - Spawning new threads
  - Limiting workload/processing sizes
Target Problem: WDPB-Loop Prediction

Traditional **Single**-Profile Setting

*Test Generation:* Hard to know triggering workload

*Test Oracle:* Hard to know how slow is slow enough

Target **Multi**-Profile Setting

Profiles
Example WDPB: 7-Zip File Manager

WDPB Context: 1. Select all items; 2. Click any item

Significant challenges
• Implicit loop
  • selection changed event fired for each selected item
• Mixture with ok context:
  1. Select one item;
  2. Click another item
Inside the Example WDPB

void CPanel::OnRefreshStatusBar() {
    ...
    GetOperatedItemIndices(indices);
    _statusBar.SetText(...);
    ...
}

Expensive UI updates
Background: **StackMine**

**Perf Debugging in the Large**

- Trace collection
- Trace Storage
- Pattern Matching
- Problematic Pattern Repository
- Bug Database

How many issues are still unknown?

Which trace file should I investigate first?

Key to issue discovery

Bottleneck of scalability

Trace analysis

Bug update

Bug filing

[Han et al. ICSE 12]
Proposed Approach: DeltaInfer

**Context-Sensitive** Delta Inference

- To gain the power of prediction

  **Temporal Inference**: differences between executions as complexity models

- To identify WDPB loops

  **Spatial Inference**: differences between program locations as WDPB loops
Insight: Temporal Inference

Regression learning + refinement to produce complexity models of program locations
Insight: Spatial Inference

WDPB loop raises complexity models of inside-loop locations to higher order
Insight: Context Sensitivity

Complexity Transitions from message distribution call to application code (handler)
Overview of DeltaInfer

Temporal Inference
- Model Inference & Refinement
- Profiles
- Workload Generation & Execution

Spatial Inference
- Model Abstraction
- Abstracted Models

Initial Workloads

Complexity Transitions
Workload Generation & Execution

Example scenario: open a file in text editor

• Performance metrics
  – execution time

• Performance-relevant workload parameters
  – # of lines (focused parameter)
    • Rep value range (RVR): [1, 1280]
    • Initial value/variations (sorted/random inputs)
  – # of character

Example workloads

  – # of lines (100, 200, ..., 500)
  – # of character (100 chars for a line)
Model Inference

- Linear Regression
  \[ y = A + Bw \]

- Power-law Regression
  \[ y = Aw^B \]

- Quality of the model
  - Correlation coefficient \( R^2 \)
Model Validation

• Model validation measures
  – relative prediction error of inferred models

• Example validation workloads:
  open a file in text editor
  – Validation value range (VVR): $[1, 2560]$
  – Guideline: $\geq 2$ times larger than the RVR
  – Caveat: too large RVR is not cost-effective
Iterative Refinement

Iterate till
- Accuracy acceptable
- Improvement < threshold

Select a new workload
- Rationale: new workload at highest-prediction-error areas improves most

Highest Prediction Error ($P_e$)

Closest Training Point ($P_t$) to $P_e$

New Training Point

RVR: Representative Value Range

VVR: Validation Value Range

$w$: Workload

Mean($P_e$, $P_t$)
Overview of DeltaInfer

Temporal Inference
- Model Inference & Refinement
  - Profiles
  - Models
- Workload Generation & Execution

Spatial Inference
- Model Abstraction
  - Abstracted Models
- Abstracted Model Comparison

Initial Workloads → Temporal Inference → Spatial Inference → Complexity Transitions
Model Abstraction: Complexity Orders

- Linear model \(y = A + Bw\)
  - \(1\), if \(B > 0\)
  - \(0\), otherwise

- Power-law model \(y = Aw^B\)
  - \(\text{Round}(B)\)

- Model w/ \(R^2\) below threshold \(R^2\) (e.g., workload-independent noise)
  - \(0\)
Inference of Complexity Transitions

Abstracted Model Comparison

Caller (Order 0, Constant)

Callee (Order 1, linear)

Complexity Transition

( RefreshListCtrl, \{GetItemRelPath, _listView.InsertItem, ...\} )

constant \(\Rightarrow\) linear
Cost Prediction of Complexity Transitions

Initial /Generated Workloads

Profile $p_1$

$Avg_{lc,p_1}$

$Avg_{lc,p_2}$

... $Avg_{lc,p_n}$

$lc$: OnRefreshStatusBar

Future Large Workload

Complexity Model

Predicted Execution Count

Predicted Cost

Ranked WDPB Loops
Evaluations of *DeltaInfer*

- **Subjects**: open-source GUI applications from SourceForge
  - 7-Zip: file manager 7,280 LOC
  - Notepad++: text editor 155,300 LOC

- **RQs**
  - RQ1: Effectiveness of WDPB Identification
  - RQ2: Effectiveness of Model Inference/Refinement
  - RQ3: Effectiveness of Context-Sensitive Analysis
## Evaluation Setup - Scenarios

<table>
<thead>
<tr>
<th>ID</th>
<th>Scenario</th>
<th>W. Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S1)</td>
<td>Open a folder</td>
<td># files</td>
</tr>
<tr>
<td>(S2)</td>
<td>Rename a file</td>
<td># files</td>
</tr>
<tr>
<td>(S3)</td>
<td>Select all items and then select the first item</td>
<td># files</td>
</tr>
<tr>
<td>(S4)</td>
<td>Create a folder</td>
<td># files</td>
</tr>
<tr>
<td>(S5)</td>
<td>Delete a file</td>
<td># files</td>
</tr>
<tr>
<td>(S6)</td>
<td>Open a file</td>
<td># lines</td>
</tr>
<tr>
<td>(S7)</td>
<td>Enter a character and save the file</td>
<td># lines</td>
</tr>
<tr>
<td>(S8)</td>
<td>Go to the last line</td>
<td># lines</td>
</tr>
<tr>
<td>(S9)</td>
<td>Find a word not present in the file</td>
<td># chars</td>
</tr>
<tr>
<td>(S10)</td>
<td>Cut and past the first character</td>
<td># lines</td>
</tr>
</tbody>
</table>
Evaluation Setup – Cont.

- Workload Selection
  - Representative Value Range (RVR)
    - Representative usage \([1, 1280]\)
  - Validation Value Range (VVR)
    - Two times as RVR \([1, 2560]\)
  - Initial workloads
    - Initial workload groups: \(\{20, 40, 80\}, \{100, 200, 400\}\)

- Thresholds
  - Max refinement iterations: 20
  - Threshold for \(R^2\): 0.9; Improvement threshold: 2\%
  - Prediction-error threshold: 5\%
RQ₁: WDPB Identification

- Manually inspect top-rank complexity transitions
- Report identified performance bugs for confirmation from developers
- Measure cost coverage of WDPBs as the workloads increase
Example Bugs of 7-Zip

Complexity Transition (constant to linear)
(RefreshListCtrl, {GetItemRelPath, _listView.InsertItem, ...}).

HRESULT RefreshListCtrl(...) {
...
for (UInt32 i = 0; i < numItems; i++) {
    const UString relPath = GetItemRelPath(i);
    ...
    if (_listView.InsertItem(&item) == -1)
        return E_FAIL;
}
...

_listView.SortItems(CompareItems, (LPARAM) this);
}

Complexity Transition (constant to power-law)
(listviewProcedure, {CompareItems, ...}).

This bug is triggered in S1, S2, S3, S5
Example Bugs of Notepad++

```cpp
bool WrapLines(...) {
    ...
    while (lineToWrap < lastLineToWrap) {
        if (WrapOneLine(surface, lineToWrap)) {
            wrapOccurred = true;
        }
        lineToWrap++;
    }
    ...
}
```

Complexity Transition (constant to linear) (WrapLines, {WrapOneLine, ...}).

This bug is triggered in S6, S7 S10
Cost Coverage of WDPBs

- Identified WDPBs account for **75%+** of costs ➔ low probability of missing impactful WDPBs
- Cost coverage increases very fast for nested loops
Bug Confirmations

• 7-Zip (5 bugs)
  • Bugs of RefreshListCtrl (S1, S2, S3, S5) confirmed with fix planned for next version
  • Bug of RefreshStatusBar (S4) introduced in release on Aug 05 and still remaining in latest release on Aug 11.

• Notepad++: (5 bugs)
  • Bug of wraplines (S6) found at the forum
  • Bugs caused by wraplines (S7, S8, S10), pending
  • New bug on search for a not-present word (S9), pending
### RQ2: Model Inference and Refinement

<table>
<thead>
<tr>
<th>ID</th>
<th># Ite.</th>
<th># WorkL</th>
<th>E. I.</th>
<th>E. E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S1)</td>
<td>4</td>
<td>4</td>
<td>35.95</td>
<td>0.62</td>
</tr>
<tr>
<td>(S2)</td>
<td>4</td>
<td>4</td>
<td>62.31</td>
<td>0.47</td>
</tr>
<tr>
<td>(S3)</td>
<td>4</td>
<td>4</td>
<td>29.68</td>
<td>0.85</td>
</tr>
<tr>
<td>(S4)</td>
<td>4</td>
<td>4</td>
<td>4.71</td>
<td>0.20</td>
</tr>
<tr>
<td>(S5)</td>
<td>3</td>
<td>3</td>
<td>5.94</td>
<td>0.18</td>
</tr>
<tr>
<td>(S6)</td>
<td>6</td>
<td>6</td>
<td>536.74</td>
<td>8.62</td>
</tr>
<tr>
<td>(S7)</td>
<td>5</td>
<td>5</td>
<td>455.00</td>
<td>7.69</td>
</tr>
<tr>
<td>(S8)</td>
<td>7</td>
<td>7</td>
<td>17.12</td>
<td>5.51</td>
</tr>
<tr>
<td>(S9)</td>
<td>4</td>
<td>4</td>
<td>138.36</td>
<td>1.83</td>
</tr>
<tr>
<td>(S10)</td>
<td>7</td>
<td>7</td>
<td>7.38</td>
<td>1.86</td>
</tr>
</tbody>
</table>

- **5 iterations (7 workloads)** to reach avg relative err 2.8%
- **Insensitive** to potential variations of initial workloads

**Initial Workloads**

**After Refinement**

Avg relative errors of inferred complexity models
RQ2: Prediction Error of Cost

<table>
<thead>
<tr>
<th>ID</th>
<th>10 (%)</th>
<th>20 (%)</th>
<th>50 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S1)</td>
<td>3.18</td>
<td>4.45</td>
<td>6.16</td>
</tr>
<tr>
<td>(S2)</td>
<td>2.98</td>
<td>4.07</td>
<td>5.55</td>
</tr>
<tr>
<td>(S3)</td>
<td>*1.40</td>
<td>*1.60</td>
<td>*1.86</td>
</tr>
<tr>
<td>(S4)</td>
<td>1.65</td>
<td>2.29</td>
<td>3.08</td>
</tr>
<tr>
<td>(S5)</td>
<td>1.58</td>
<td>2.19</td>
<td>2.95</td>
</tr>
<tr>
<td>(Ave(7-Zip))</td>
<td>*2.35</td>
<td>*3.25</td>
<td>*4.44</td>
</tr>
<tr>
<td>(S6)</td>
<td>18.51</td>
<td>6</td>
<td>47.24</td>
</tr>
<tr>
<td>(S7)</td>
<td>16.84</td>
<td>5</td>
<td>36.28</td>
</tr>
<tr>
<td>(S8)</td>
<td>16.80</td>
<td>7</td>
<td>35.23</td>
</tr>
<tr>
<td>(S9)</td>
<td>11.15</td>
<td>4</td>
<td>39.09</td>
</tr>
<tr>
<td>(S10)</td>
<td>10.79</td>
<td>7</td>
<td>24.63</td>
</tr>
<tr>
<td>(Ave(Notepad++))</td>
<td>14.82</td>
<td>20.97</td>
<td>36.49</td>
</tr>
</tbody>
</table>

X RVR Upper Bound

Prediction error

- **4.4%** (7-Zip file manager): excluding S3
- **36.5%** (Notepad++): robust even under complex situations

Developers optimize message processing during idle time
RQ3: Context Sensitivity

<table>
<thead>
<tr>
<th>ID</th>
<th>DeltaInfer</th>
<th># L.</th>
<th>InSen.</th>
<th># Missed by InSen</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S1)</td>
<td>11</td>
<td>10</td>
<td>521</td>
<td>6</td>
</tr>
<tr>
<td>(S2)</td>
<td>21</td>
<td>19</td>
<td>579</td>
<td>12</td>
</tr>
<tr>
<td>(S3)</td>
<td>17</td>
<td>16</td>
<td>486</td>
<td>10</td>
</tr>
<tr>
<td>(S4)</td>
<td>21</td>
<td>19</td>
<td>640</td>
<td>12</td>
</tr>
<tr>
<td>(S5)</td>
<td>22</td>
<td>20</td>
<td>546</td>
<td>12</td>
</tr>
<tr>
<td>(S6)</td>
<td>10</td>
<td>10</td>
<td>509</td>
<td>3</td>
</tr>
<tr>
<td>(S7)</td>
<td>29</td>
<td>14</td>
<td>877</td>
<td>6</td>
</tr>
<tr>
<td>(S8)</td>
<td>10</td>
<td>10</td>
<td>526</td>
<td>5</td>
</tr>
<tr>
<td>(S9)</td>
<td>20</td>
<td>20</td>
<td>131</td>
<td>0</td>
</tr>
<tr>
<td>(S10)</td>
<td>12</td>
<td>11</td>
<td>861</td>
<td>3</td>
</tr>
</tbody>
</table>

- Context helps **reduce false positives & negatives**
  - No context: > 90% of identified WDPB loops being false positives
  - No context: 40% of DeltaInfer-identified WDPB loops being missed
- Context helps achieve only **14%** of identified WDPB loops being false positives (top/low-level sys lib calls)
Conclusion

• Predictive approach for WDPBs: *context-sensitive delta inference*
  – **Temporal inference** ➞ complexity models
    • Deltas of different executions (workloads)
  – **Spatial inference** ➞ complexity transitions
    • Order deltas of different locations

• Evaluations: effectively identifies impactful **WDPBs** (for causing 10 performance bugs)
Thank You!

Supported in part by NSF grants CCF-0845272, CCF-0915400, CNS-0958235, CNS-1160603
Conclusion

• Predictive approach for WDPBs: context-sensitive delta inference
  – Temporal inference ➔ complexity models
    • Deltas of different executions (workloads)
  – Spatial inference ➔ complexity transitions
    • Order deltas of different locations

• Evaluations: effectively identifies impactful WDPBs (for causing 10 performance bugs)
Discussion

• Generalization to Other Types of Applications

• Multiple Workload Parameters

• Value-Dependent Performance Bottlenecks

• Scalability of Scenario-Based Profiling
Software Performance

• An important quality of software
  – A kind of non-functional requirement
  – Characterized by the amount of work accomplished given time and resources

• Software performance matters
  – Software functionality and size grows faster than hardware
  – Little work has been done to help developers avoid performance-related mistakes

http://betanews.com/2012/05/09/software-performance-matters/
Two Significant Challenges in GUI Applications

• Complex contexts
  – GUI applications are event-driven applications
  – A program location may exhibit different complexities under different contexts

• Implicit WDPB loops
  – Event handlers can be invoked repetitively
    • E.g., selection change events for all items
  – No explicit loop statements
    • Cause challenges to manual inspection and static analysis
Limitations of Traditional Approaches

• Traditional approaches
  – Performance testing (blackbox-random testing or manual testing)
  – Profiling (call-tree profiling and callstack sampling)

41 out of 109 studied performance bugs are due to wrong assumption of workloads. [Jin et al. PLDI 2012]

• Two major issues
  – Insufficiency
    • WDPBs may not surface on given workloads
    • Workload specifications are usually missing or outdated
  – Incompleteness:
    • WDPBs may overshadow other WDPBs
Least-Squares Regression

- **Linear Regression**
  - Infers: \( y = A + Bw \)
  - Minimizes: \( Q(A, B) = \sum_{i=1}^{k} (y_{l\text{c},i} - (A + Bw_i))^2 \)

- **Power-law Regression**
  - Infers: \( y = A w^B \)
  - Minimizes: \( Q(A, B) = \sum_{i=1}^{k} (y_{l\text{c},i} - (A w_i^B))^2 \)

- **How good does the model fit the data points?**
  - Correlation coefficient
    - \( R^2 = \frac{(\sum_{i=1}^{k} w y - k \bar{w} \bar{y})^2}{(\sum_{i=1}^{k} w^2 - k \bar{w}^2)(\sum_{i=1}^{k} y^2 - k \bar{y}^2)} \)
    - \( \bar{w} \) is the mean of \( w \), \( \bar{y} \) is the mean of \( y \)
A Few Definitions

• Application $A$, location $l$ and cost $y$
• Call graph $G(E,V)$, calling context $c$, and execution profile $P$
• $k$-profile Graph: an annotated call graph, $G(E,V)$, where a location $l$ with its corresponding vertex is annotated with a vector of counters for $l$ on $k$ workloads for each of its calling context $c$. 
Complexity Transitions

• A pair \((n, M)\), such that:
  – 1. \(n\) is a vertex (method) in the \(k\)-profile graph and \(M\) is a subset of children vertices (callees) of \(n\);
  – 2. \(f_{n,c}(W)\) is the complexity model of \(n\) under the calling context \(c\), and \(f_{li,ci}(W)\) is the complexity model of the location \(l_i\), where \(l_i\) is a location in \(M\) and the calling context \(c_i\) is \(c\) concatenated with \(n\).
  – 3. \(O(f_{li,c_i}(W))\) is at least 1 more than \(O(f_{n,c}(W))\);
  – 4. \(\forall l_i, l_j \in M, i \neq j, O(f_{li,ci}(W)) = O(f_{lj,cj}(W)).\)
Model Inference and Refinement

Align Profiles
- Align locations using calling contexts
- Extract execution vector for each location under each calling context

Regression Learning

Model Validation

Termination Checks

Select New Workloads
- Assumption: a new workload at the area with the highest prediction error improves most

```python
1: pc = -1 // previous model count
2: e_pre = -1 // previous error
3: for i = 0; i < max_itr; i = i + 1 do
4:    kG = AlignProfiles(P)
5:    x = GetWorkloads(P)
6:    M = RegressionLearning(x, kG)
7:    e_M = {}
8:    for all vp in VP do
9:        w = GetWorkload(vp)
10:       e_vp = 0.0
11:      for all m in M do
12:         r_w = Predict(w, m)
13:         a_m = GetActual(vp, m)
14:         e_vp = e_vp + Abs(a_m - r_w)
15:     end for
16:     e_M = e_M.Add(e_vp / M.Count)
17:     end for
18:     e_total = Sum(e_M) / NPM.Count
19:     if e_pre - e_total < threshold_imp then
20:        break // improvement is below the threshold
21:     end if
22:     if e_total < threshold_e AND pc == M.Count then
23:        break // accuracy is acceptable
24:     else
25:        np = NewWorkload(P, VP, e_M)
26:        P = P.Add(np)
27:        pc = M.Count
28:        e_pre = e_total
29:     end if
30: end for
31: return M
```
Cost Prediction of Complexity Transitions

- Compute $\text{avg}_{lc,p}$ for each location $l_c$ on each profile $p$
  - E.g., for $p_1$, $\text{Cost}(\text{refreshList}_c) = 1\text{s}$, $\text{ExeCount}(\text{refreshList}_c) = 100$, $\text{avg}_{lc,p} = 1/100 \text{s}$

- Compute $\text{avg}_{lc} = \text{average}(\text{avg}_{lc,p})$

- Given a workload value $w$, $\text{pred}_{lc,w} = f_{lc}(w)$

- Get $\text{Cost}_{lc,w} = \text{pred}_{lc,w} \times \text{avg}_{lc}$