Argus: Debugging Performance Issues in Modern Desktop Applications with Annotated Causal Tracing

Lingmei Weng, Columbia University
Peng Huang, Johns Hopkins University
Jason Nieh, Columbia University
Junfeng Yang, Columbia University
From a spinning pinwheel
To wait or to kill? It is a hard question to answer!
Existing tools for diagnosing desktop apps

- Debugger (e.g., macOS spindump, lldb)

- Profiler (e.g., macOS Instruments)
  - more about potential optimization, may not a root cause
Why diagnosing desktop apps is so hard?

- Multiple components
- High concurrency
Desktop app diagnosis is under-investigated
An example of existing causal tracing

*Figure from Panappticon for Android system*
Existing causal tracing fails to diagnose desktop apps

- It is hard to identify **accurate execution segment boundaries in some threads**
- Some execution segments have **multiple incoming edges** (large search space)
Where are the inaccuracies from?

- Over-connections: unnecessary searching paths
  - Batch processing
  - Piggyback optimization
  - Superfluous thread wake-up (mutual access VS causality)
  - ...

- Under-connections: missing edges
  - ad-hoc sync with data flags
  - Data dependencies
  - ...

Why the inaccuracies happen to the desktop apps?

**Existing causal tracing assumptions**
- White-box annotation
- Known programming paradigms

**Desktop apps**
- Closed sourced components
  (Inaccurate execution segment boundaries)
- Various custom programming paradigms
  (multiple incoming edges)

- Can we fix all inaccuracies with additional tracing in desktop apps?
  - hard to define all programming paradigms correctly
  - overhead
Critical path is sensitive to graph inaccuracy

- The result of critical path analysis is easily distorted by inaccurate graphs
Key insights

- Tracing graphs from existing causal tracing are not accurate enough to effectively diagnose performance issues in desktop applications.

- Completely eliminating inaccuracies is impractical, we should make causal tracing and diagnosis algorithm inaccuracy-tolerant.
Argus workflow

Argus Tracer
- helper 1
- daemon 1
- daemon 2
- Third-party libraries ...
- Instrumented core libraries
- Instrumented OS

Trace logs

| Time | Event   | Attr1 | Attr2 | ...
|------|---------|-------|-------|------
| 30.4 | sendMsg | port1 | port2 |      
| 31.7 | wakeUp  | tid0  | tid3  |      
| 33.2 | wakeUp  | tid2  | tid1  |      
| ....  |         |       |       |      

Argus Grapher
- annotated trace graph

Argus Debugger
- beam search diagnosis

Root cause vertex
- (1) Costly operations
- (2) Culprit event sequences
- (3) Call stacks
Annotated tracing graphs

- An example of edge annotation to mitigate over-connections

- execution segment
- edge

- strong edge
- boosted weak edge
- weak edge

fontd main thread

fontd worker

Chromium

SystemPreference

Owly

Chromium

Chromium

SystemPreference

Chromium

fontd worker

Owly
Back to the Chromium case

Anomaly segment

Batch processing
Causal search: beam search based

- Expanding phase: explore all possible paths
- Pruning phase: select paths based on

![Diagram showing the process of causal search with beam search, including expanding and pruning phases, and highlighting an anomaly segment. The beam width is set to 2 and lookback steps to 2. The diagram also illustrates the distinction between weak edges, boosted weak edges, strong edges, and execution segments, as well as selected states.](image-url)
Sub-graph comparison

- Diagnosing the complicated performance issue in Chromium
- why a similar vertex to A does not appear in the anomaly graph

![Diagram showing anomaly and normal sub-graphs with nodes and edges representing fontd, Chromium renderer, and Chromium browser processes.](image)
## Real world performance issues

<table>
<thead>
<tr>
<th>ID</th>
<th>App</th>
<th>Bug Descriptions</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>Chromium</td>
<td>Typing non-English in searchbox, page freezes.</td>
<td>7 yr</td>
</tr>
<tr>
<td>B2</td>
<td>TeXstudio</td>
<td>Modifying Bib file in other app gets pinwheel.</td>
<td>2 yr</td>
</tr>
<tr>
<td>B3</td>
<td>BiglyBT</td>
<td>Launching BiglyBT installer gets pinwheel.</td>
<td>1 yr</td>
</tr>
<tr>
<td>B4</td>
<td>Sequel Pro</td>
<td>Reconnection via ssh causes freeze.</td>
<td>4 yr</td>
</tr>
<tr>
<td>B5</td>
<td>Quiver</td>
<td>Pasting a section from webpage as a list freezes.</td>
<td>5 yr</td>
</tr>
<tr>
<td>B6</td>
<td>Firefox</td>
<td>Connection to printer takes a long time.</td>
<td>1 mo</td>
</tr>
<tr>
<td>B7</td>
<td>Firefox</td>
<td>Some website triggers pinwheel in the DevTool.</td>
<td>3 yr</td>
</tr>
<tr>
<td>B8</td>
<td>Alacritty</td>
<td>Unresponsive after a long line rendering.</td>
<td>6 mo</td>
</tr>
<tr>
<td>B9</td>
<td>Inkscape</td>
<td>Zoom in/out shapes causes intermittent freeze.</td>
<td>1 yr</td>
</tr>
<tr>
<td>B10</td>
<td>VLC</td>
<td>Quick quit after playlist click causes freeze.</td>
<td>7 mo</td>
</tr>
<tr>
<td>B11</td>
<td>QEMU</td>
<td>Unable to launch on macOS Catalina.</td>
<td>1 mo</td>
</tr>
<tr>
<td>B12</td>
<td>Octave</td>
<td>Script editing in GUI gets pinwheel.</td>
<td>2 yr</td>
</tr>
</tbody>
</table>

* Diagnosis runs on binary releases even though some apps are open-sourced.*

17
## Evaluation 1: diagnosis effectiveness

<table>
<thead>
<tr>
<th>Tool</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>B10</th>
<th>B11</th>
<th>B12</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>spindump</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>5/12</td>
</tr>
<tr>
<td>Instruments</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>4/12</td>
</tr>
<tr>
<td>AppInsight</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>2/12</td>
</tr>
<tr>
<td>Panappticon</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>4/12</td>
</tr>
<tr>
<td>Argus</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>12/12</td>
</tr>
</tbody>
</table>
Evaluation 2: diagnosis cost

- Time cost of Argus on diagnosing the 12 real world performance issues

![Graph showing time cost of Argus on diagnosing performance issues]
Evaluation 3: tracing overhead

<table>
<thead>
<tr>
<th>Benchmark Id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
<th>k</th>
<th>l</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys time w/ Argus</td>
<td>2248</td>
<td>2298</td>
<td>242</td>
<td>2224</td>
<td>2505</td>
<td>2390</td>
<td>2390</td>
<td>15888</td>
<td>5234</td>
<td>18816</td>
<td>20245</td>
<td>59045</td>
<td>36524</td>
</tr>
<tr>
<td>User time w/ Argus</td>
<td>242</td>
<td>3806</td>
<td>150</td>
<td>129</td>
<td>1538</td>
<td>1538</td>
<td>4743</td>
<td>15234</td>
<td>40</td>
<td>56</td>
<td>277</td>
<td>277</td>
<td>2282</td>
</tr>
<tr>
<td>Sys time w/o Argus</td>
<td>17490</td>
<td>18186</td>
<td>333806</td>
<td>4980</td>
<td>4743</td>
<td>15888</td>
<td>20245</td>
<td>59045</td>
<td>36524</td>
<td>&lt; 5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User time w/o Argus</td>
<td>18186</td>
<td>18186</td>
<td>2298</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
<td>2224</td>
</tr>
</tbody>
</table>

a: webrtc  
b: dromaeo  
c: blink_perf  
d: speedometer  
e: octan.desktop  
f: memory_desktop  
g: smoothness.oop_rasterization.top_25_smooth  
h: v8.browsing_desktop  
i: page_cycler_v2.typical_2  
j: dummy_benchmark.histogram  
k: system_health.memory_desktop  
l: loading.desktop.network_serv  
m: rasterize&record_micro.top_25
Conclusions

- Diagnosing performance issues in desktop is important but was under-investigated.

- Existing causal tracing is inaccurate when applied to desktop apps.
  
  Finding 1: both over-connections and under-connections exist, and several programming patterns can lead to the inaccuracies.

  Finding 2: diagnosis algorithm needs to tolerate inaccuracies.

- We design Argus, an annotated causal tracing tool for diagnosing performance issues on desktop apps using inaccuracy-tolerant diagnosis algorithm.

- Source code is available https://github.com/columbia/ArgusDebugger.
Related work

Distributed systems

- Magpie [OSDI’04], XTrace [NSDI’07], Dappa [GoogleTechReport 2010], Pivot [SOSP’15], Canopy [SOSP’17], BaggageContext [EuroSys’18]

Mobile Apps

- AppInsight [OSDI’12], Panappticon [CODES+ISSS’13]

Performance profiling

- Gprof [SIGPLAN’82], COZ [SOSP’15], D4 [PLDI’18]