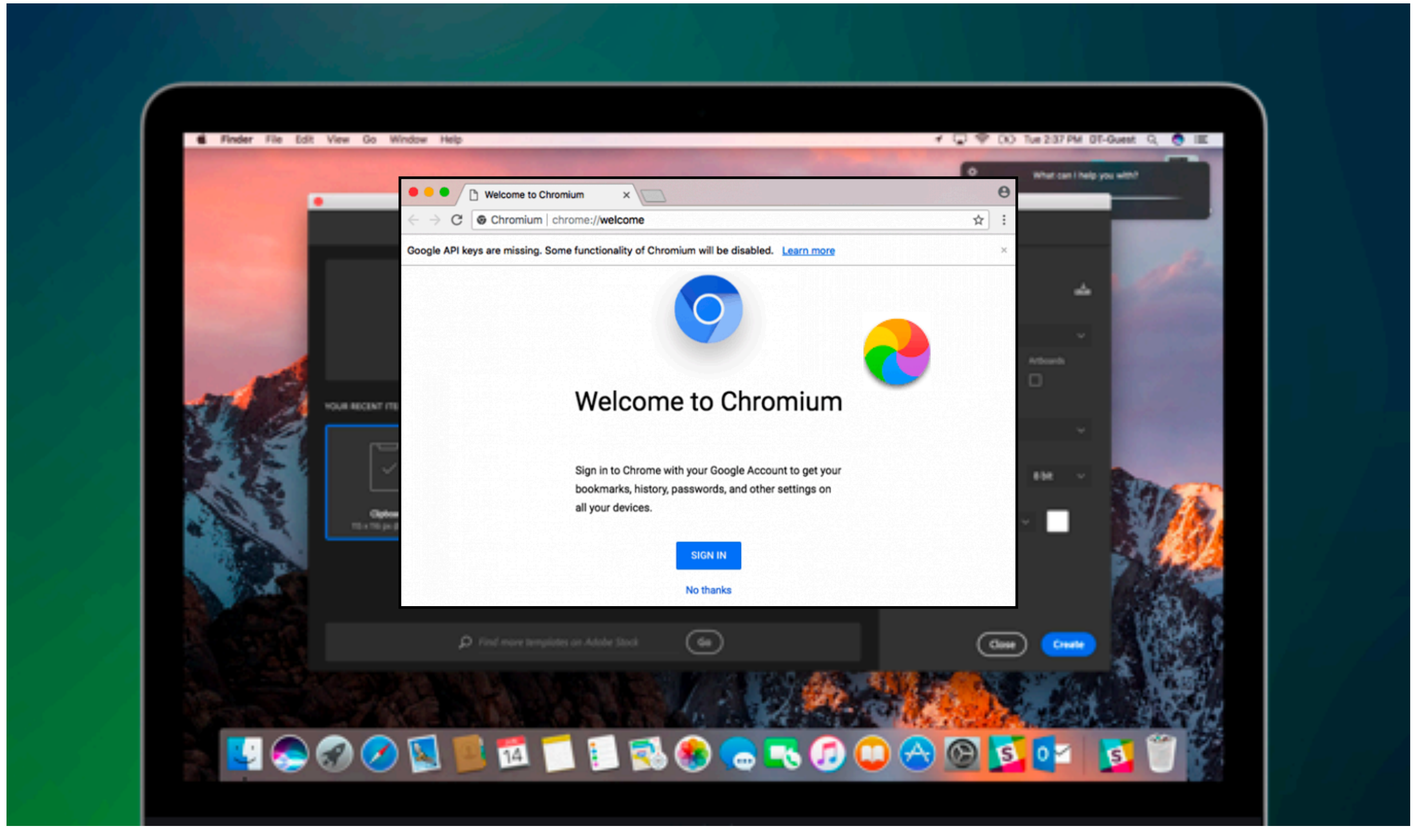
Argus: Debugging Performance Issues in Modern Desktop Applications with Annotated Causal Tracing

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!

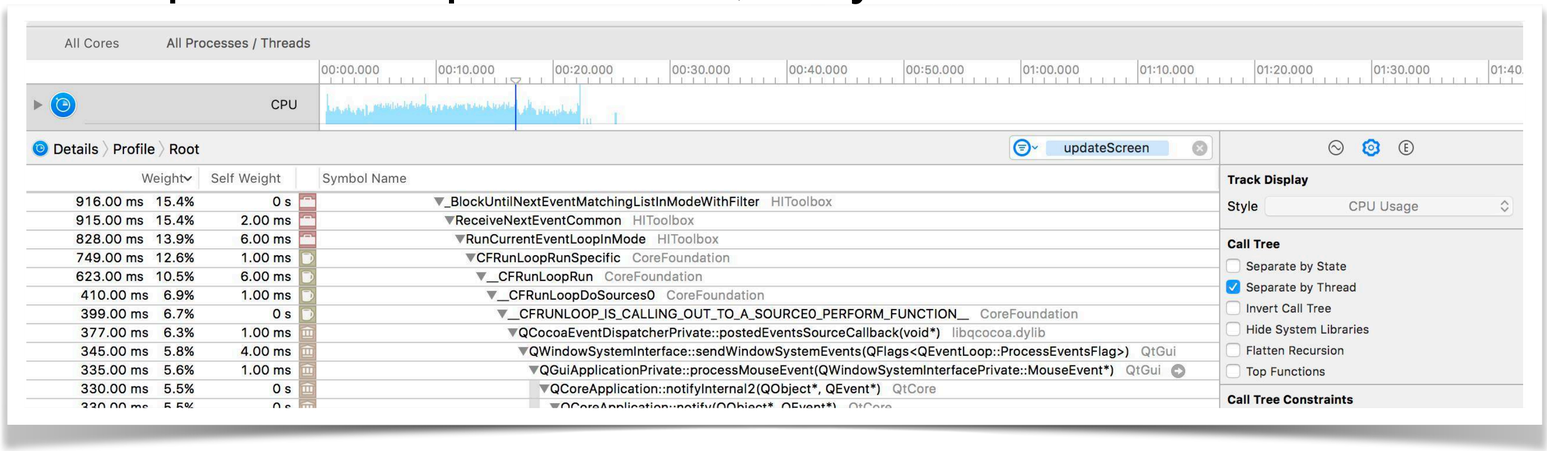
Existing tools for diagnosing desktop apps

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

```
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 2.1
  * frame #0: 0x0000000104667db AutoTestPlatform`-[ViewController viewDidLoad]
    frame #1
    frame #2
    frame #3
    frame #4
    frame #5
```

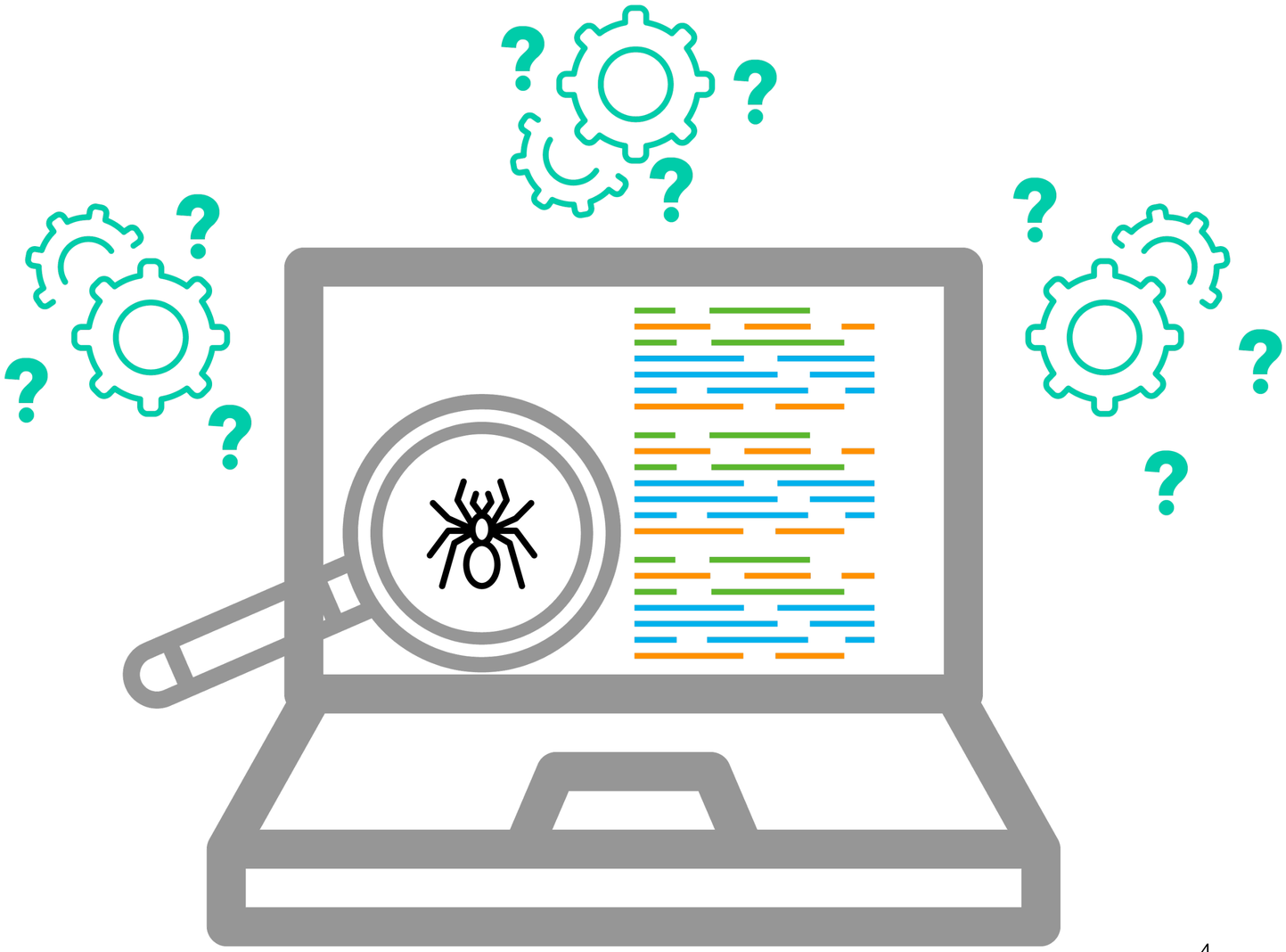
```
thread 0x33dfaf    DispatchQueue "com.apple.main-thread"(1),"com.apple.root.user-interactive-qos"(14),"CGImageProviderC
ority 47 (base 47)    cpu time 0.280s (716.2M cycles, 573.0M instructions, 1.25c/i)
000  start + 1 (libdyld.dylib + 109769) [0x7fff6e04dcc9]
 1000  NSApplicationMain + 777 (AppKit + 15238) [0x7fff31422b86]
  972  -[NSApplication run] + 658 (AppKit + 204158) [0x7fff31450d7e]
   971  -[NSApplication(NSEvent) _nextEventMatchingEventMask:untilDate:inMode:dequeue:] + 1352 (AppKit + 262256) [0x
    971  _DPSNextEvent + 883 (AppKit + 268329) [0x7fff31460829]
     971  _BlockUntilNextEventMatchingListInModeWithFilter + 64 (HIToolbox + 193913) [0x7fff32e18579]
      965  ReceiveNextEventCommon + 584 (HIToolbox + 194517) [0x7fff32e187d5]
       965  RunCurrentEventLoopInMode + 292 (HIToolbox + 195261) [0x7fff32e18abd]
        965  CFRunLoopRunSpecific + 462 (CoreFoundation + 532174) [0x7fff341e9ece]
         925  __CFRunLoopRun + 1319 (CoreFoundation + 535122) [0x7fff341eaa52]
          925  __CFRunLoopServiceMachPort + 247 (CoreFoundation + 540549) [0x7fff341ebf85]
           925  mach_msg_trap + 10 (libsystem_kernel.dylib + 3578) [0x7fff6e18edfa]
```

- Profiler (e.g., macOS Instruments)
  - more about potential optimization, may not a root cause

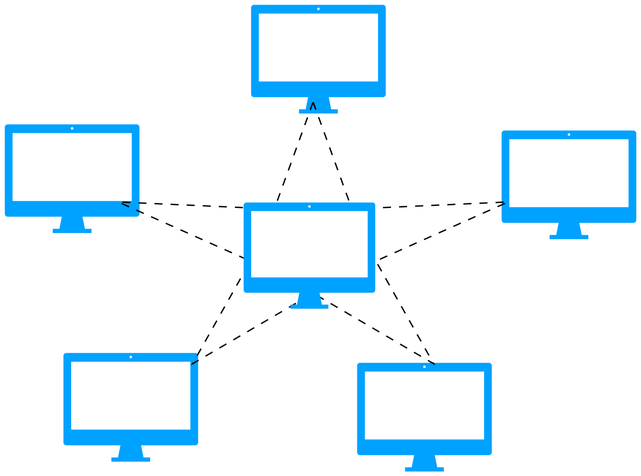


Why diagnosing desktop apps is so hard?

- Multiple components
- High concurrency

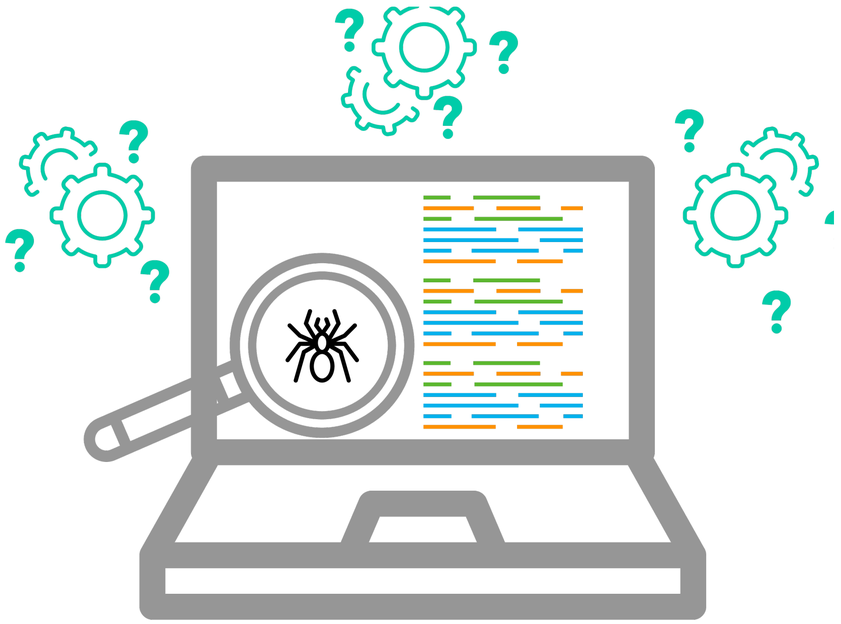


Desktop app diagnosis is under-investigated



Distributed Systems

Mobile Apps



Desktop Apps

Causal Tracing

An example of existing causal tracing

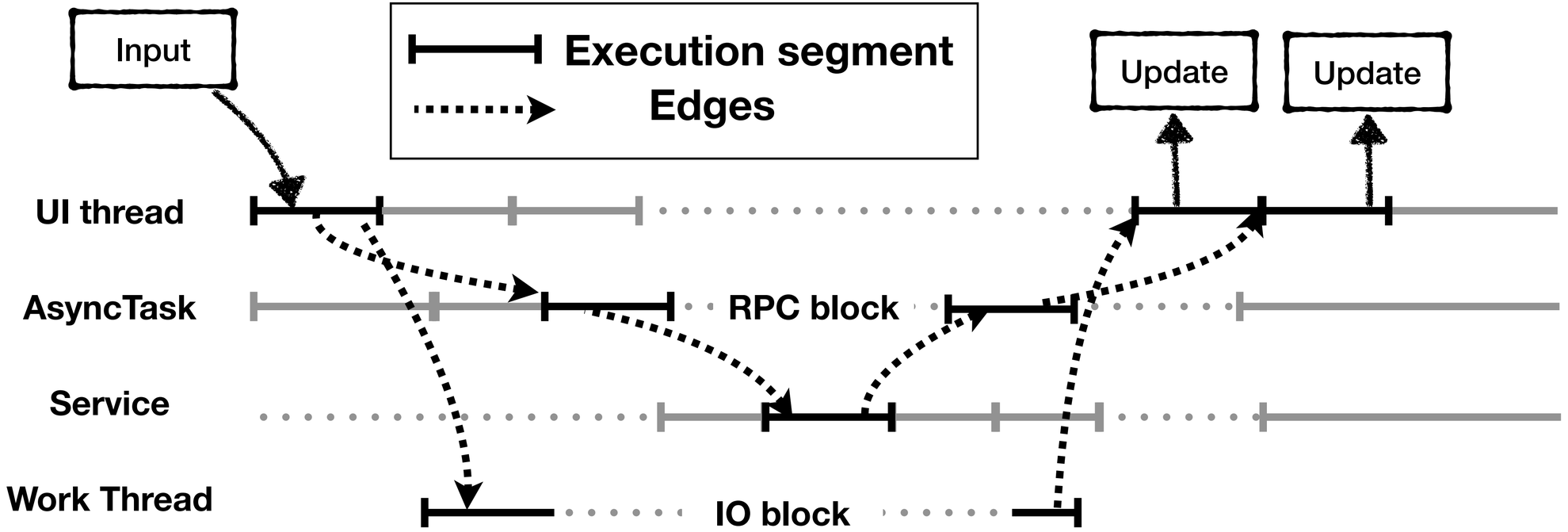
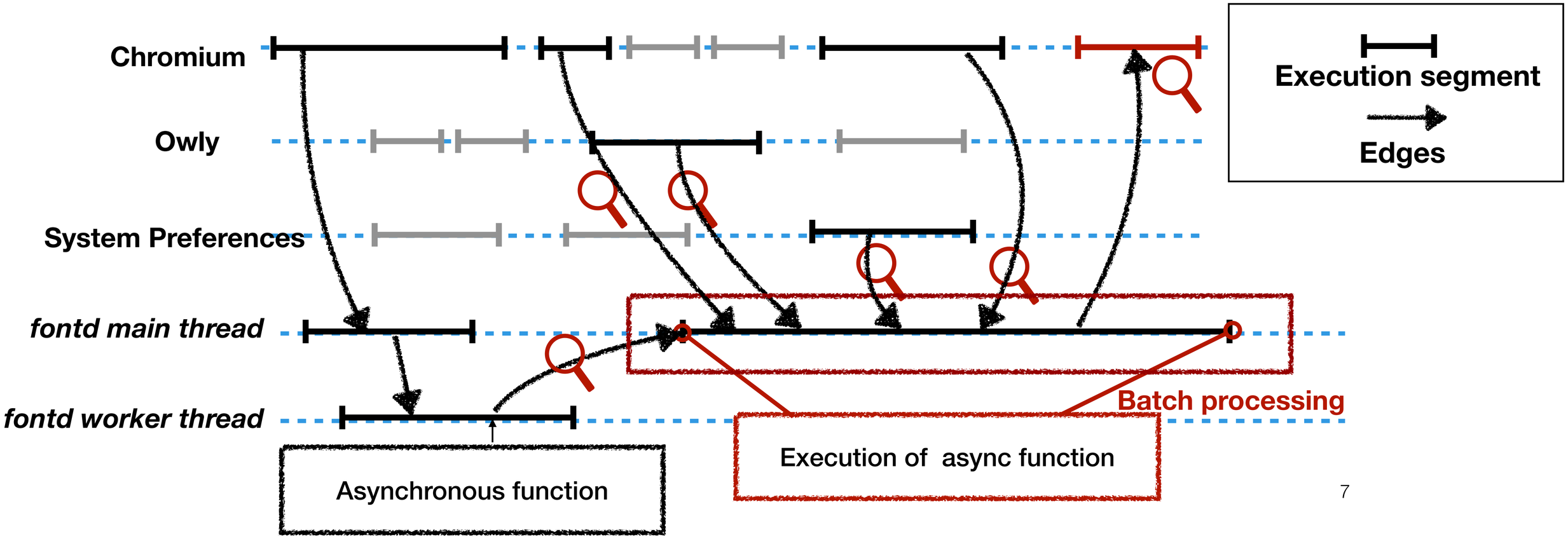


**Figure from Panappticon for Android system*

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

vs

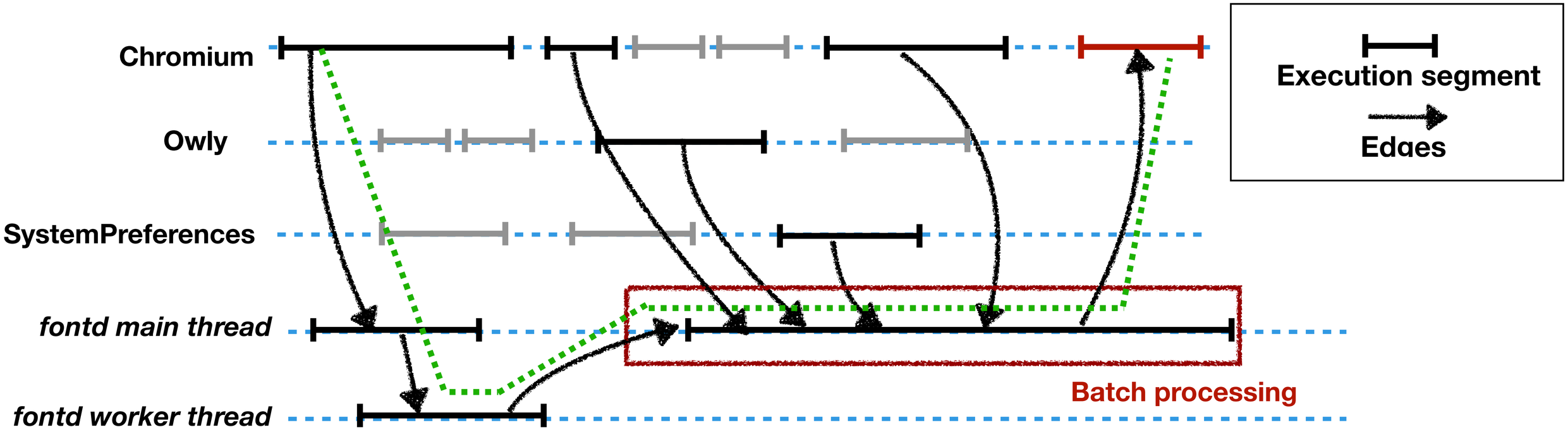
Desktop apps

- 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

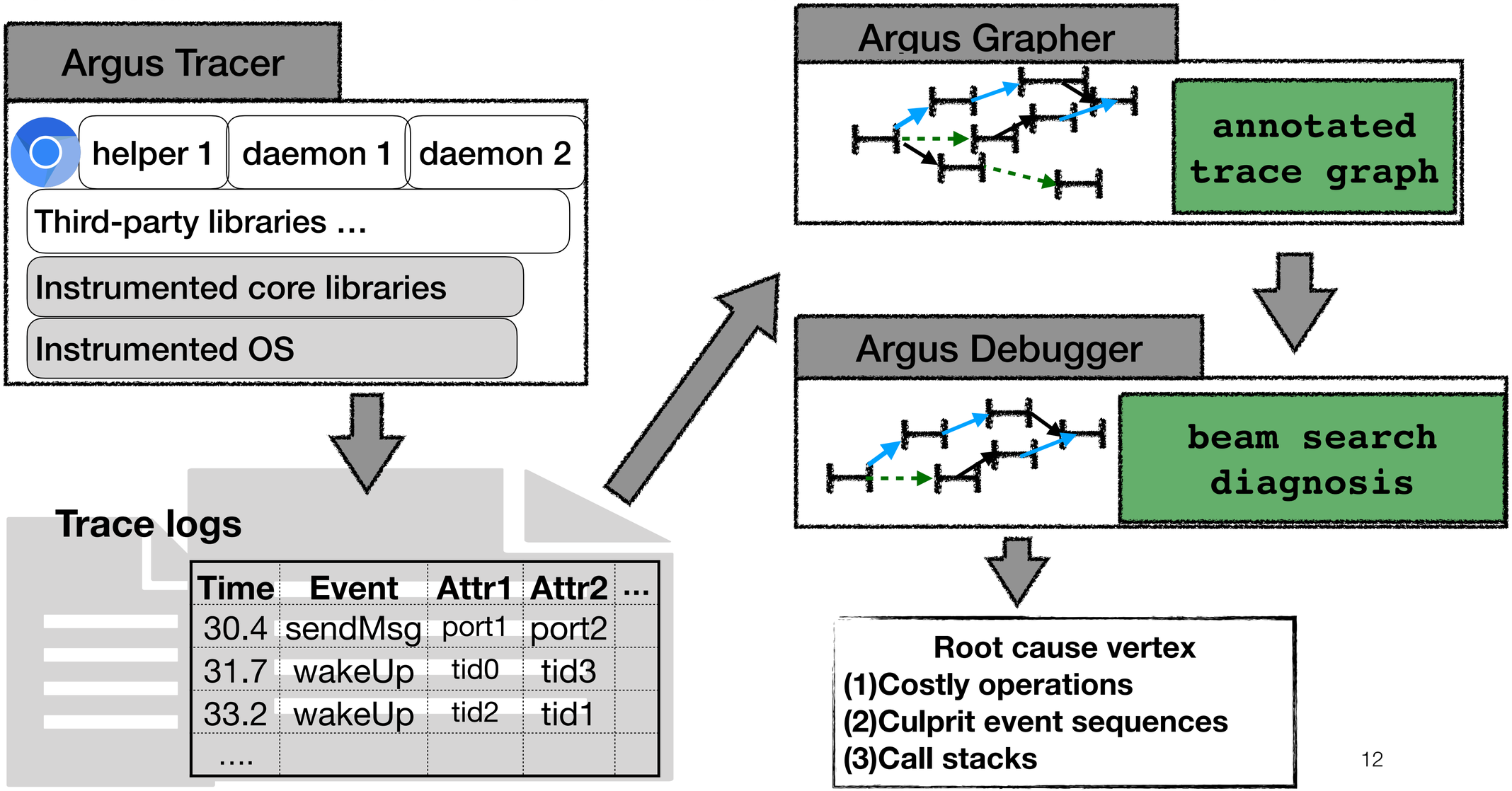
- The result of critical path analysis is easily distorted by inaccurate graphs



Key insights

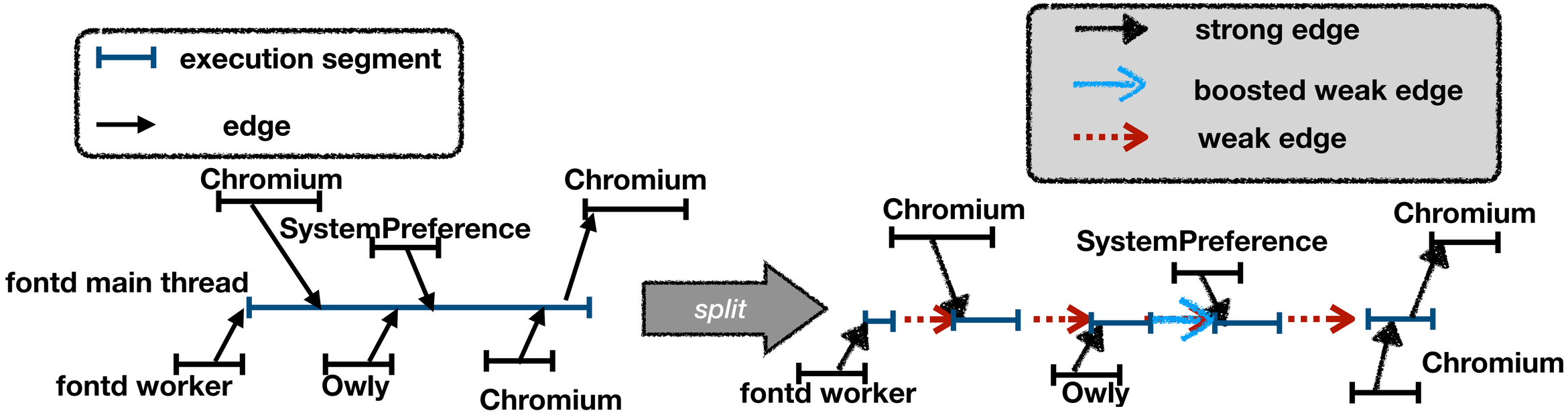
- Tracing graphs from existing causal tracing are not accurate enough to effectively diagnose performance issues in desktop applications.
- Completely eliminating inaccuracies is impractical, we should make causal tracing and diagnosis algorithm **inaccuracy-tolerant**.

Argus workflow



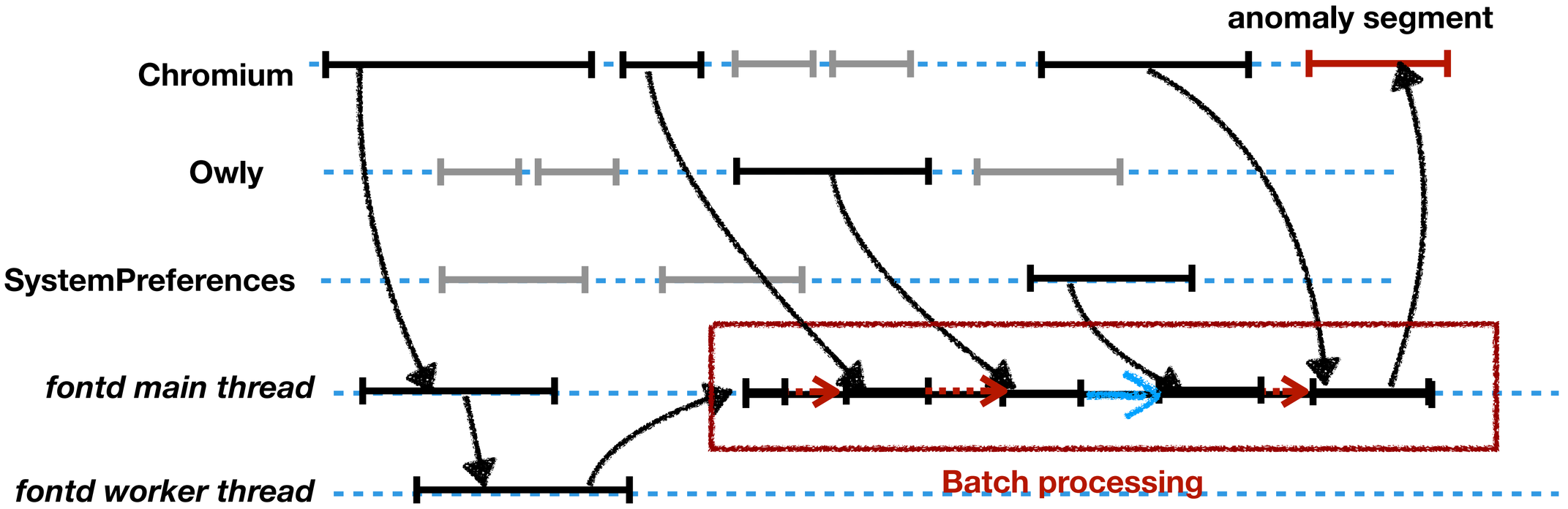
Annotated tracing graphs

- An example of edge annotation to mitigate over-connections



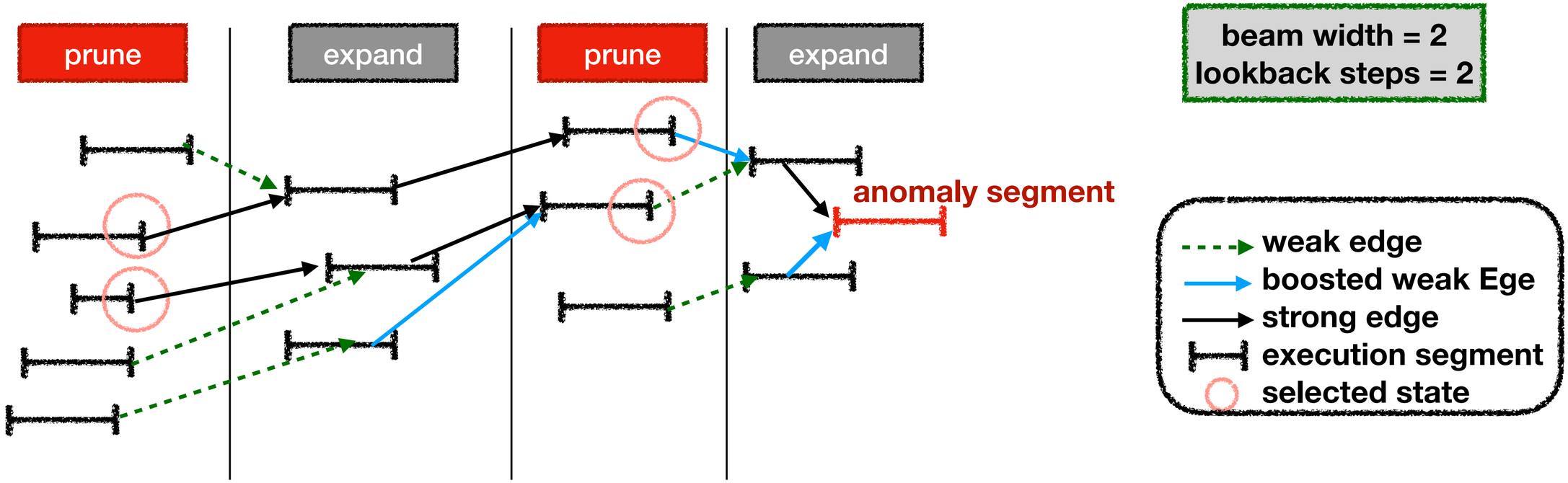
Annotated tracing graph

- Back to the Chromium case



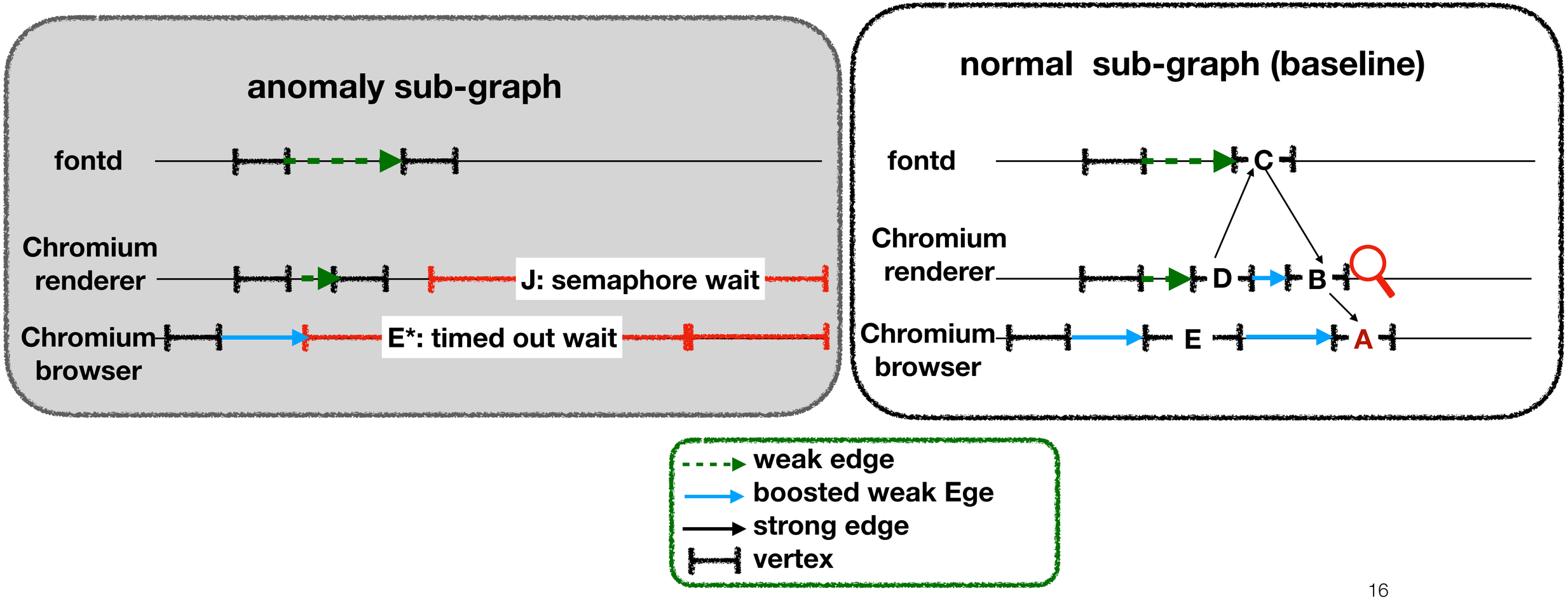
Causal search: beam search based

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



Sub-graph comparison

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



Real world performance issues

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

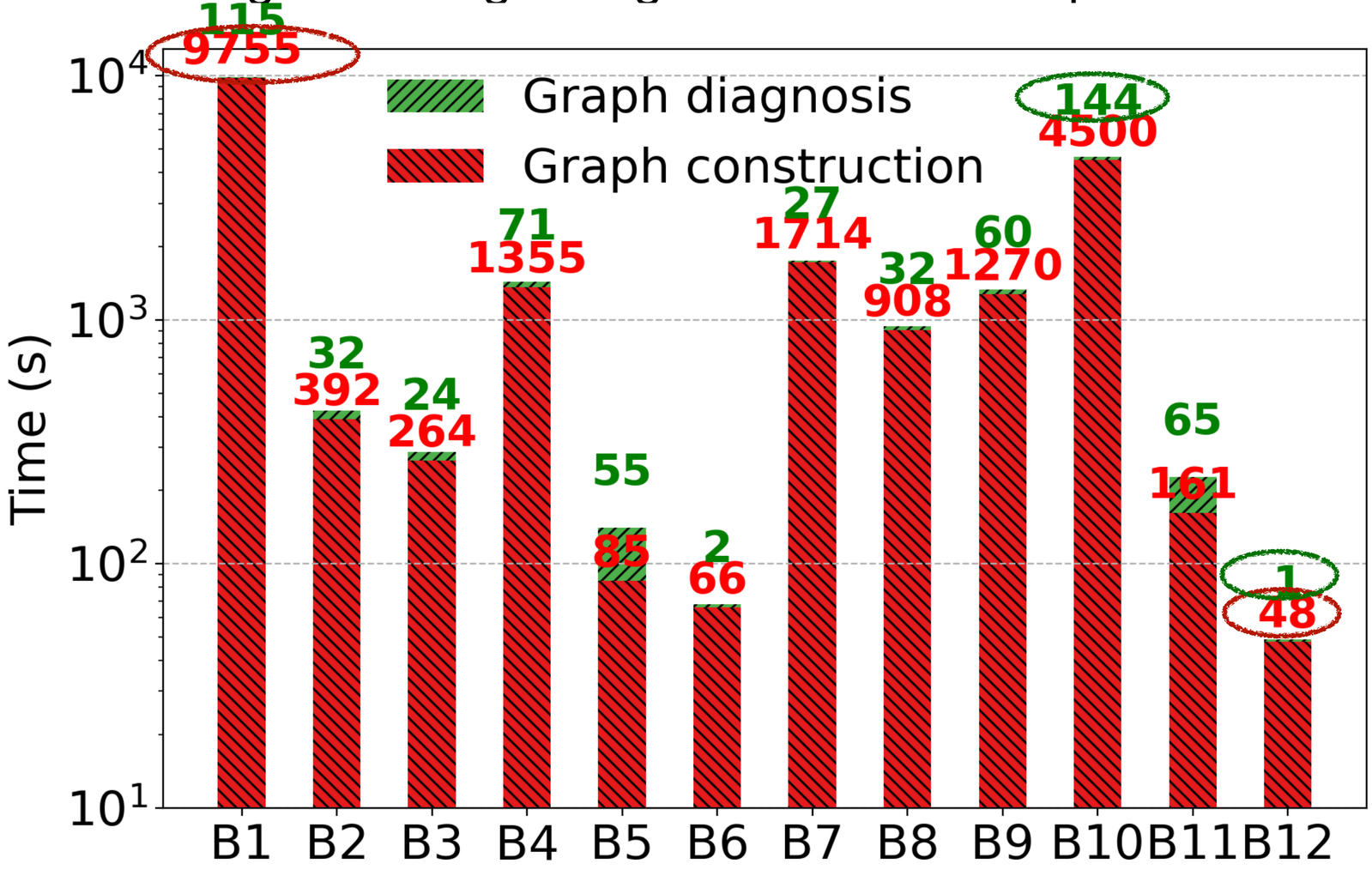
** Diagnosis runs on binary releases even though some apps are open-sourced.*

Evaluation 1: diagnosis effectiveness

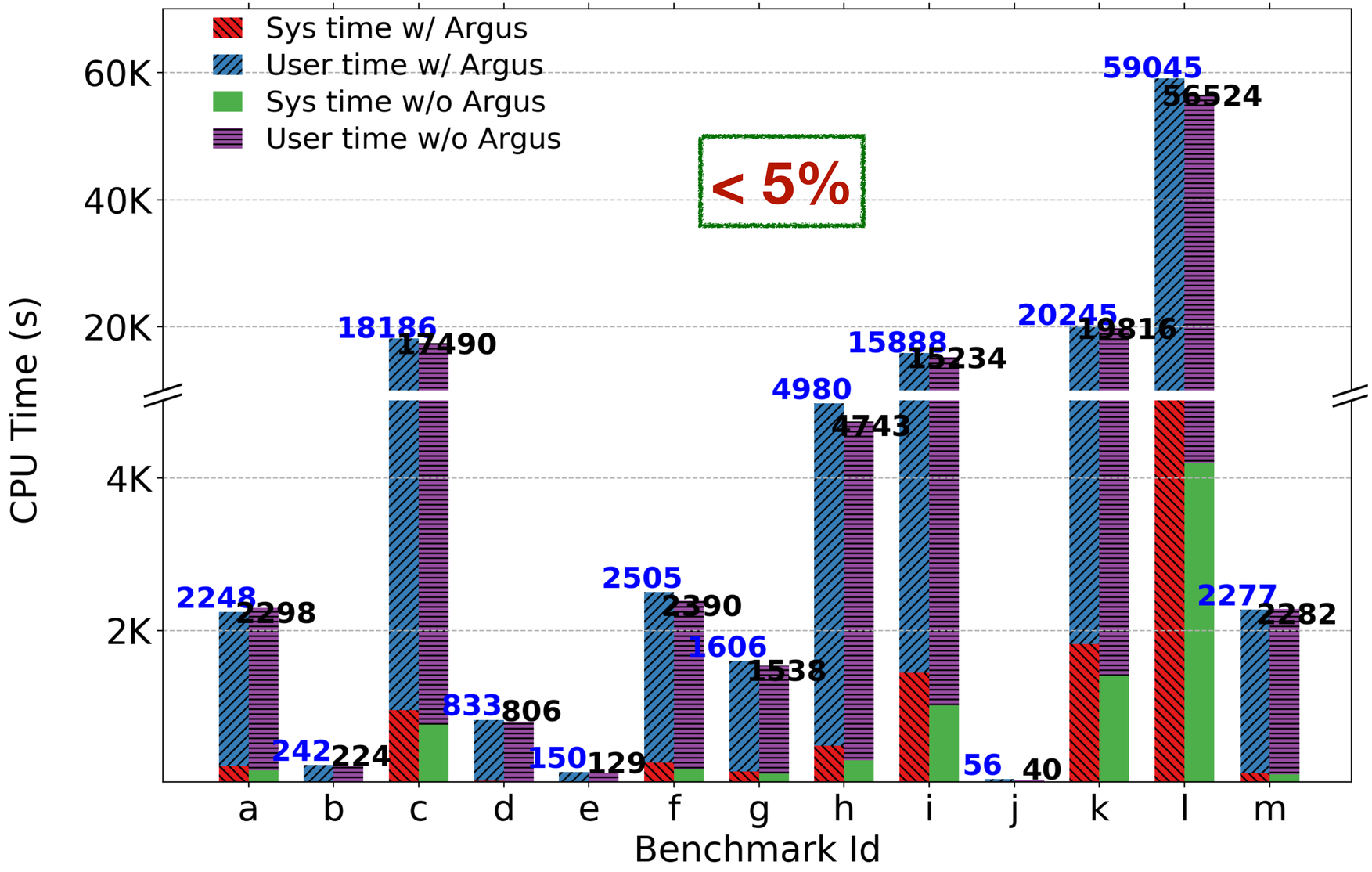
Tool	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	Total
spindump	X	X	X	X	X	✓	✓	✓	✓	X	✓	X	**5/12**
Instruments	X	X	X	X	X	✓	✓	✓	✓	X	X	X	**4/12**
AppInsight	X	X	X	X	X	X	✓	✓	X	X	X	X	**2/12**
Panappticon	X	X	X	X	✓	✓	✓	✓	X	X	X	X	**4/12**
Argus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**12/12**

Evaluation 2: diagnosis cost

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



Evaluation 3: tracing overhead



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]