Failure Sketches: A Better Way to Debug

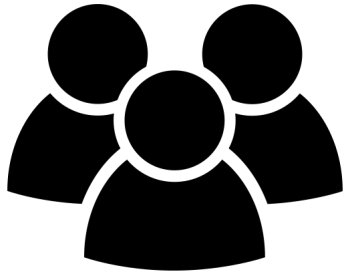
Baris Kasikci, Cristiano Pereira, Gilles Pokam,
Benjamin Schubert, Madan Musuvathi, George Candea

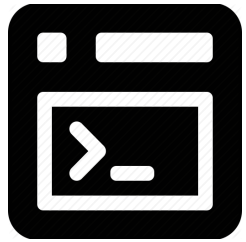
Failure and Root Cause

- Failure
  - Violation of a program specification
  - Memory errors, hangs, etc
- Root cause
  - “The real reason” behind the failure
  - When removed from the program, the failure does not recur

Debugging In-Production Software Failures Today

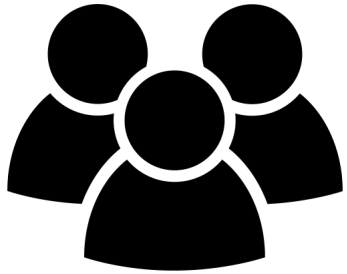
Debugging In-Production Software Failures Today

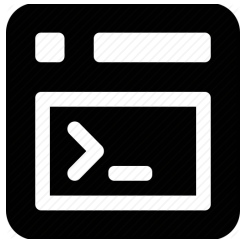




Debugging In-Production Software Failures Today

```
#0  0x00007f51abae820b in raise (sig=11) at ../nptl/
sysdeps/unix/sysv/linux/pt-raise.c:37
#1  0x000000000042d289 in ap_buffered_log_writer
(r=0x7f51a40053d0, handle=0x20eeba0,
strs=0x7f51a4003578, strl=0x7f51a40035e8, nelts=14,
len=82) at mod_log_config.c:1368
#2  0x000000000042b10d in config_log_transaction
(r=0x7f51a40053d0, cls=0x20b9d50,
default_format=0x20ee370) at mod_log_config.c:930
#3  0x000000000042aad6 in multi_log_transaction
(r=0x7f51a40053d0) at mod_log_config.c:950
#4  0x000000000046cb2d in ap_run_log_transaction
(r=0x7f51a40053d0) at protocol.c:1563
#5  0x0000000000436e81 in ap_process_request
(r=0x7f51a40053d0) at http_request.c:312
#6  0x000000000042e9da in ap_process_http_connection
(c=0x7f519c000b68) at http_core.c:293
#7  0x0000000000465cdd in ap_run_process_connection
(c=0x7f519c000b68) at connection.c:85
#8  0x00000000004661f5 in ap_process_connection
(c=0x7f519c000b68, csd=0x7f519c000a20) at
connection.c:211
#9  0x0000000000451ba0 in process_socket
(p=0x7f519c0009b8, sock=0x7f519c000a20,
my_child_num=0, my_thread_num=0,
bucket_alloc=0x7f51a4001348) at worker.c:632
#10 0x0000000000451221 in worker_thread
(thd=0x210fa90, dummy=0x7f51a40008c0) at worker.c:946
#11 0x00007f51ac87c555 in dummy_worker
(opaque=0x210fa90) at thread.c:127
#12 0x00007f51abae0182 in start_thread
(arg=0x7f51aa8ef700) at pthread_create.c:312
#13 0x00007f51ab80d47d in clone () at ../sysdeps/
unix/sysv/linux/x86_64/clone.S:111
```

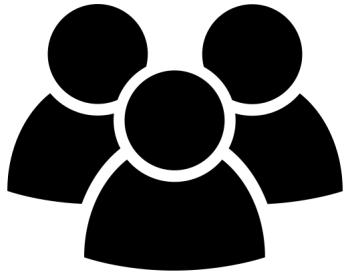


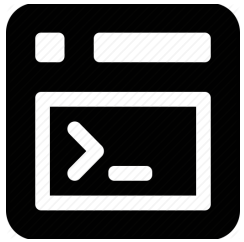


Debugging In-Production Software Failures Today

Understand root cause

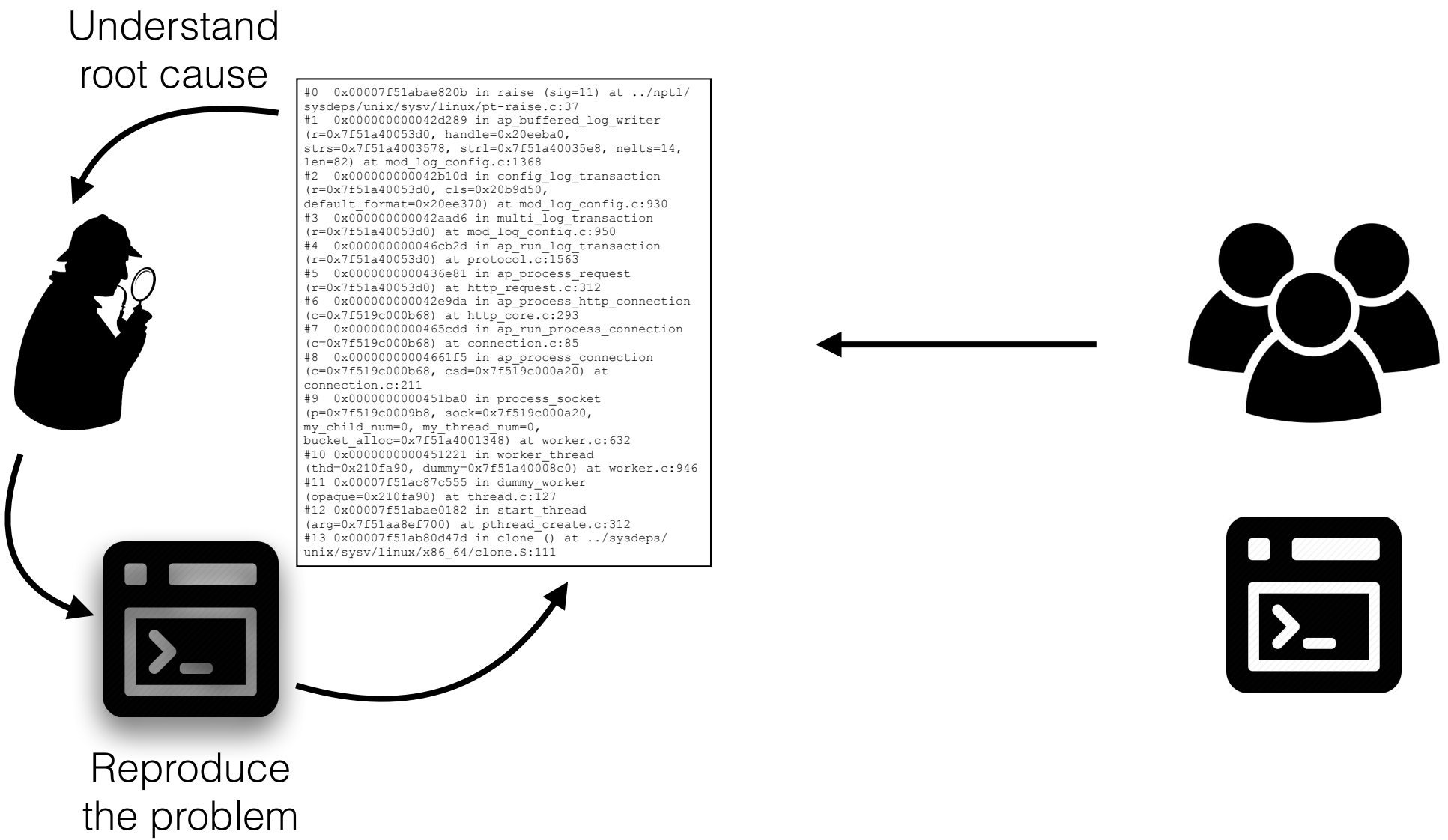
```
#0  0x00007f51abae820b in raise (sig=11) at ../nptl/
sysdeps/unix/sysv/linux/pt-raise.c:37
#1  0x000000000042d289 in ap_buffered_log_writer
(r=0x7f51a40053d0, handle=0x20eeba0,
strs=0x7f51a4003578, strl=0x7f51a40035e8, nelts=14,
len=82) at mod_log_config.c:1368
#2  0x000000000042b10d in config_log_transaction
(r=0x7f51a40053d0, cls=0x20b9d50,
default_format=0x20ee370) at mod_log_config.c:930
#3  0x000000000042aad6 in multi_log_transaction
(r=0x7f51a40053d0) at mod_log_config.c:950
#4  0x000000000046cb2d in ap_run_log_transaction
(r=0x7f51a40053d0) at protocol.c:1563
#5  0x0000000000436e81 in ap_process_request
(r=0x7f51a40053d0) at http_request.c:312
#6  0x000000000042e9da in ap_process_http_connection
(c=0x7f519c000b68) at http_core.c:293
#7  0x0000000000465cdd in ap_run_process_connection
(c=0x7f519c000b68) at connection.c:85
#8  0x00000000004661f5 in ap_process_connection
(c=0x7f519c000b68, csd=0x7f519c000a20) at
connection.c:211
#9  0x0000000000451ba0 in process_socket
(p=0x7f519c0009b8, sock=0x7f519c000a20,
my_child_num=0, my_thread_num=0,
bucket_alloc=0x7f51a4001348) at worker.c:632
#10 0x0000000000451221 in worker_thread
(thd=0x210fa90, dummy=0x7f51a40008c0) at worker.c:946
#11 0x00007f51ac87c555 in dummy_worker
(opaque=0x210fa90) at thread.c:127
#12 0x00007f51abae0182 in start_thread
(arg=0x7f51aa8ef700) at pthread_create.c:312
#13 0x00007f51ab80d47d in clone () at ../sysdeps/
unix/sysv/linux/x86_64/clone.S:111
```





Debugging In-Production Software Failures Today

Debugging In-Production Software Failures Today



Tackling the Debugging Challenge

- Record/replay
- Special runtime support[1]
  - VM checkpointing
- Custom hardware[2]
  - Not widely available

[1] J. Tucek et al., Triage: Diagnosing Production Run Failures at the User's Site, SOSP 2007

[2] G. Pokam et al., QuickRec: prototyping an intel architecture extension for record and replay of multithreaded programs, ISCA 2013

Tackling the Debugging Challenge

- Record/replay
- Special runtime support[1]
  - VM checkpointing
- Custom hardware[2]
  - Not widely available

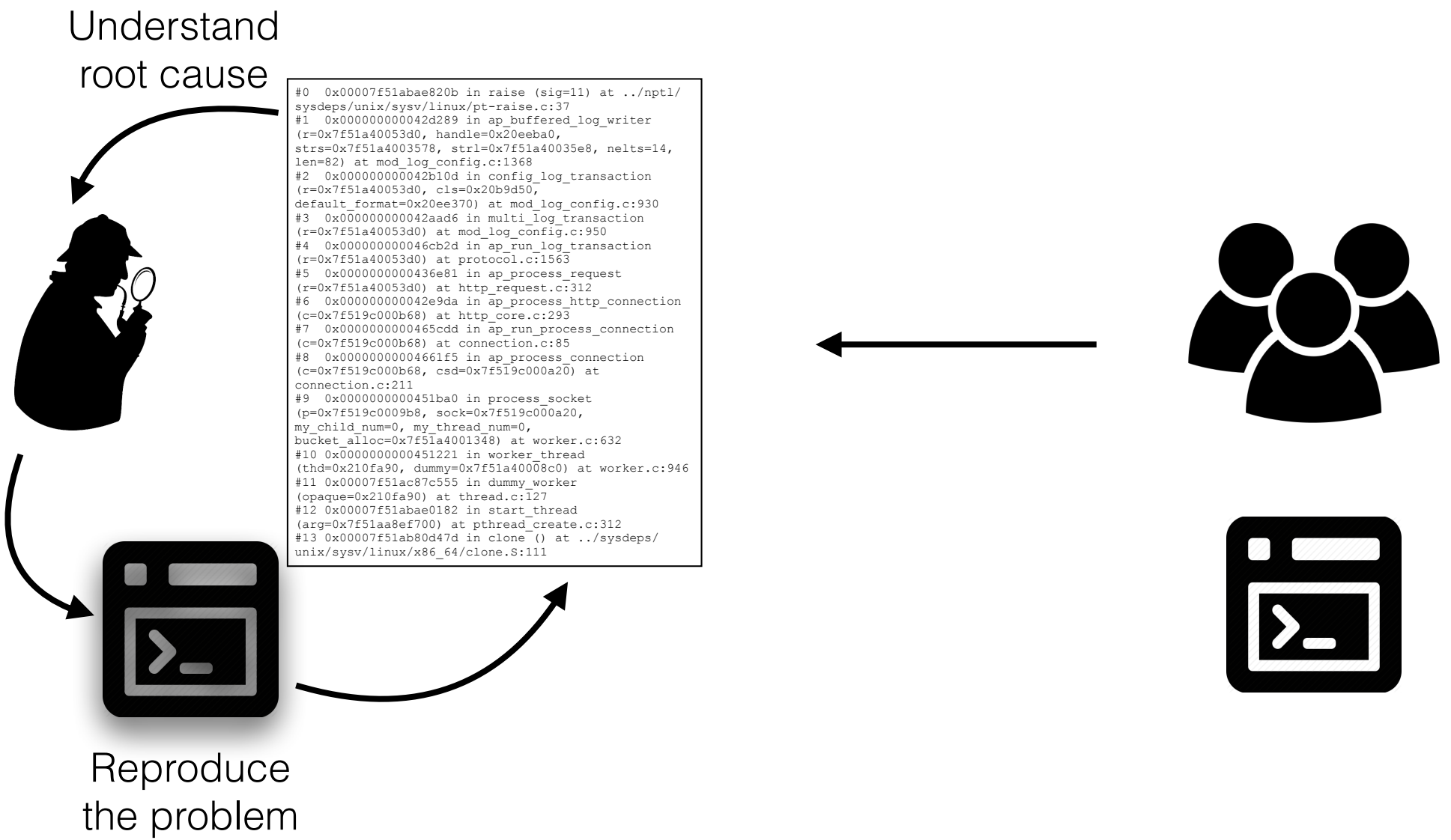
Existing tools don't help debugging in-production failures[3]

[1] J. Tucek et al., Triage: Diagnosing Production Run Failures at the User's Site, SOSP 2007

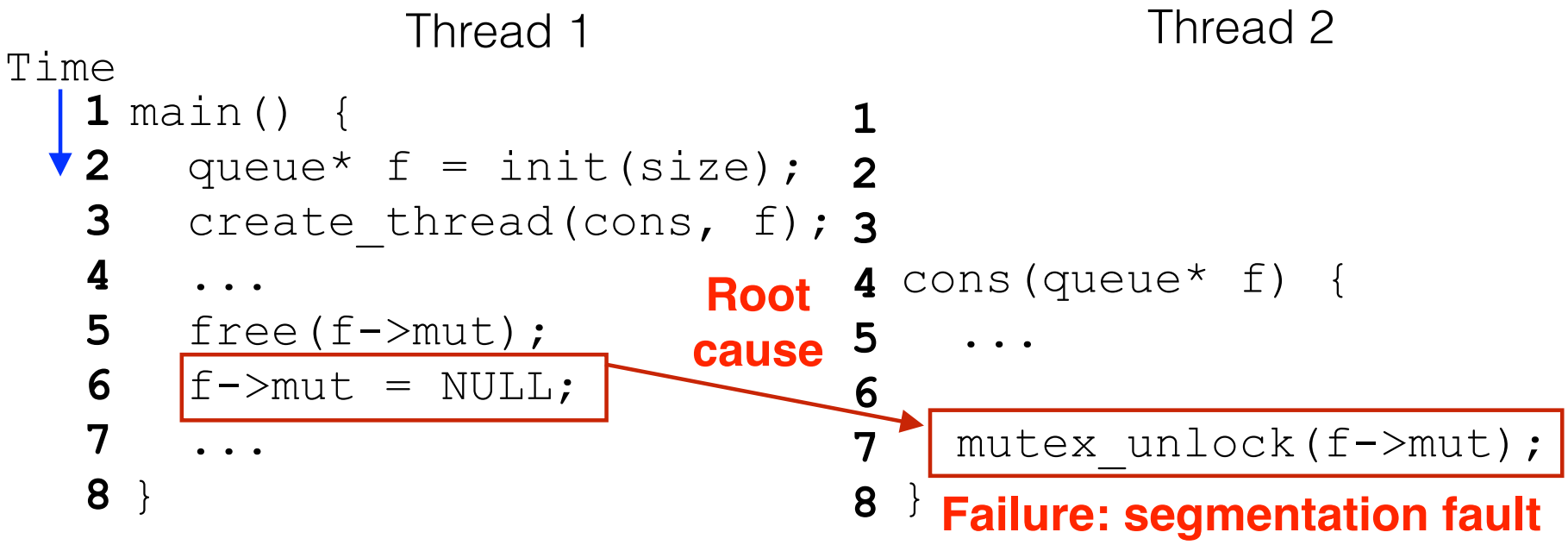
[2] G. Pokam et al., QuickRec: prototyping an intel architecture extension for record and replay of multithreaded programs, ISCA 2013

[3] C. Sadowski et al. ,How developers use data race detection tools, Workshop on Evaluation and Usability of Programming Languages and Tools 2014

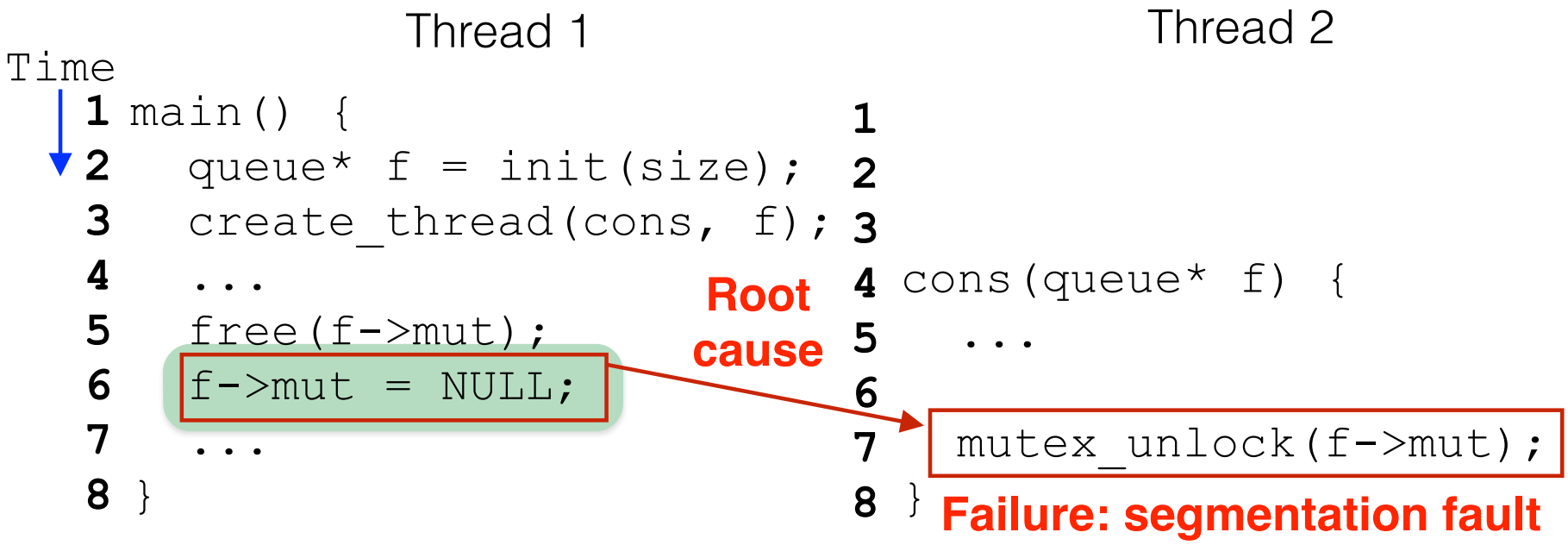
Debugging In-Production Software Failures Today



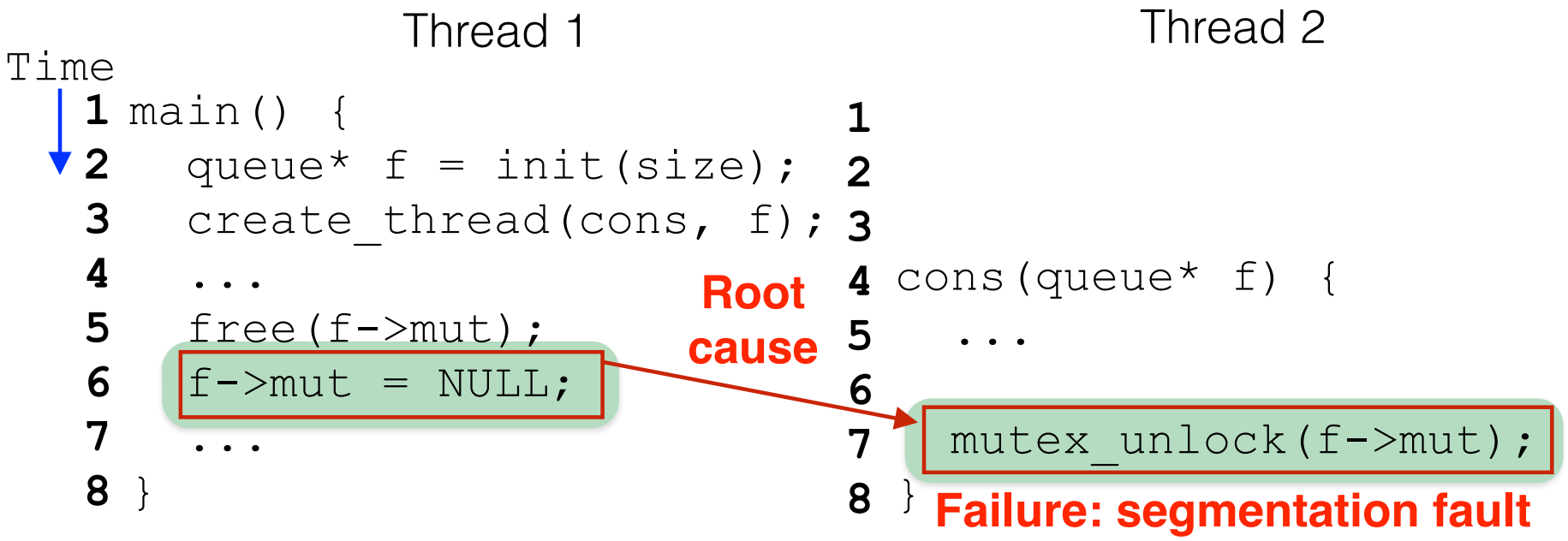
Failure Sketch



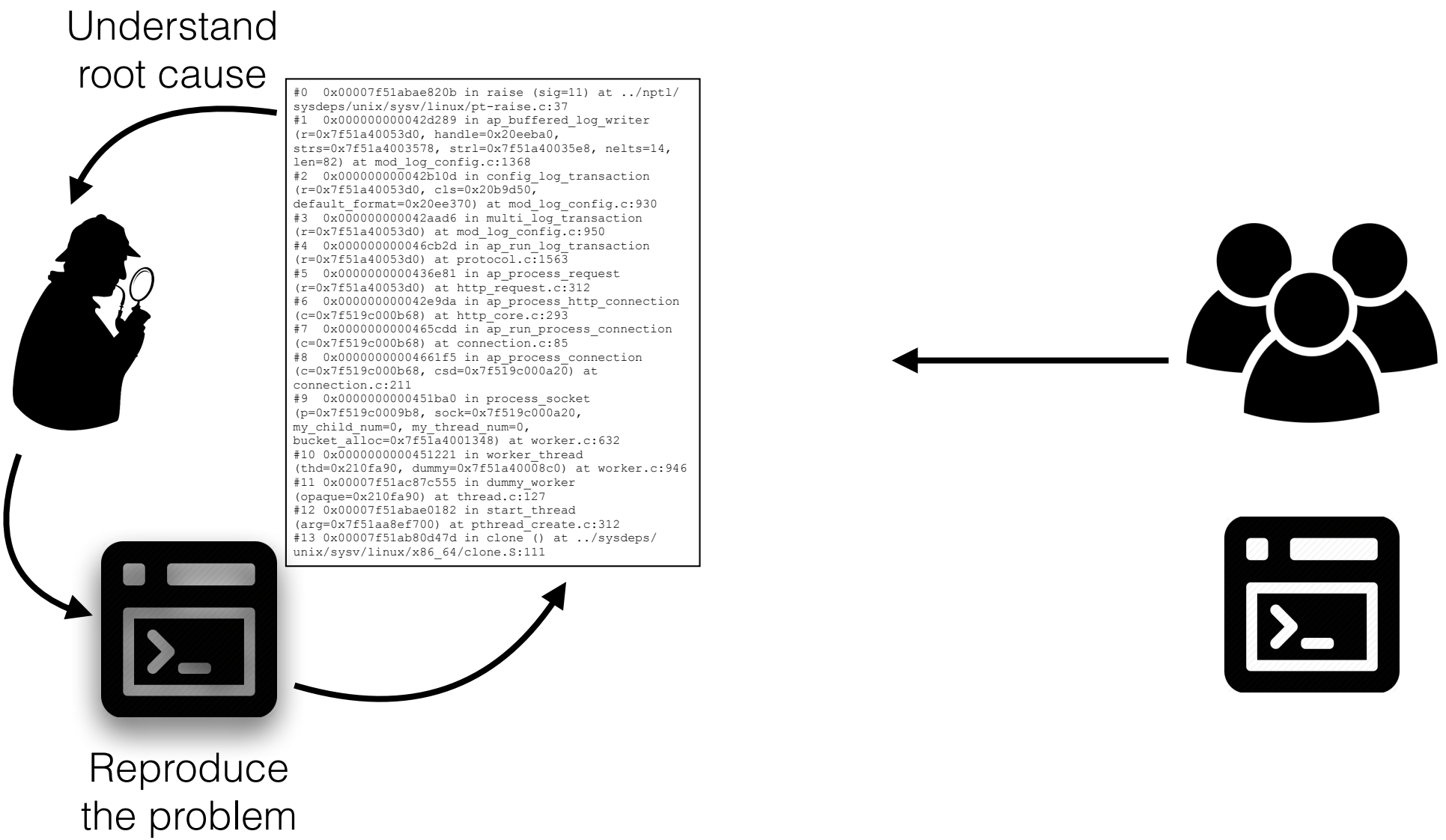
Failure Sketch



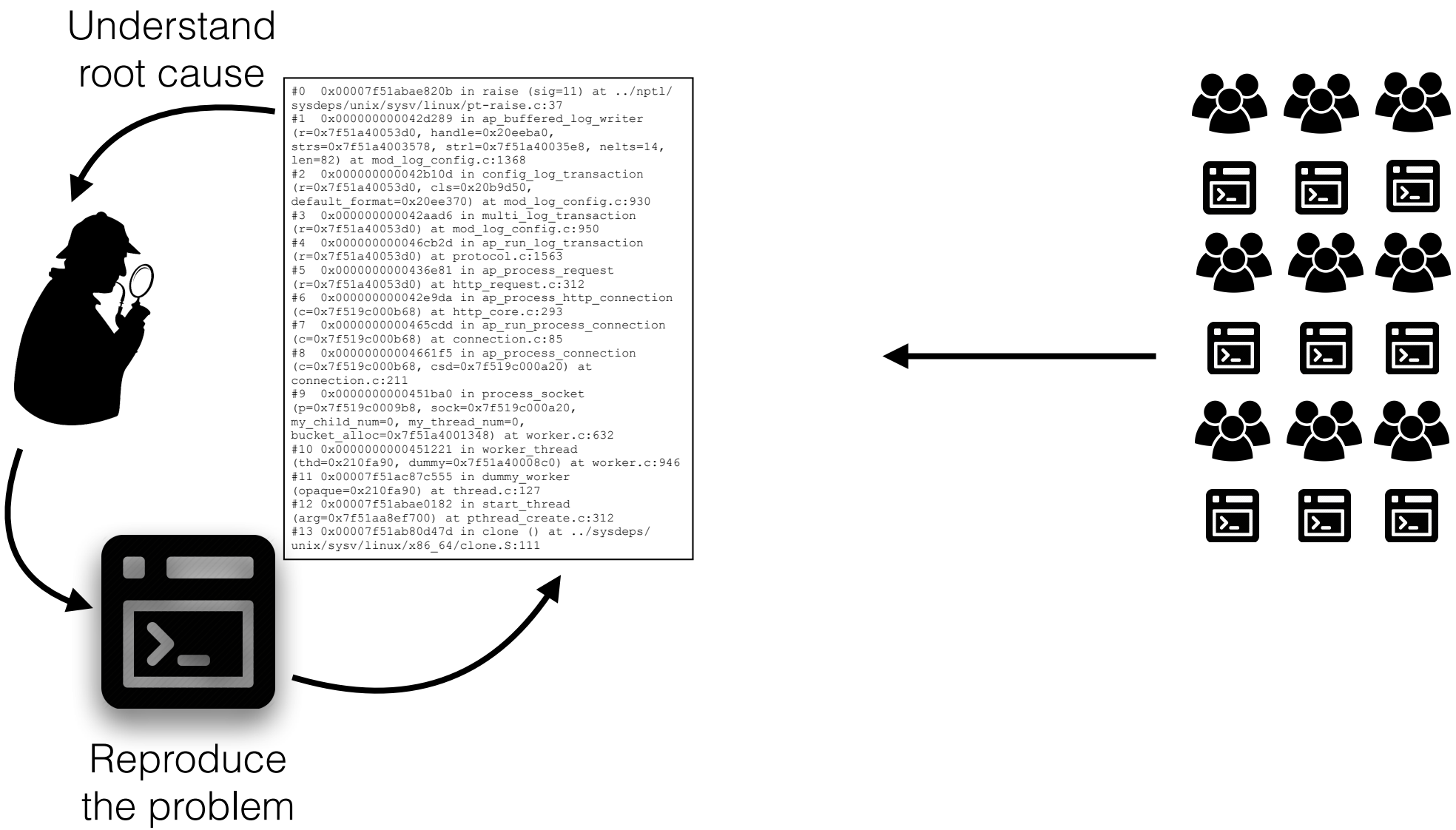
Failure Sketch



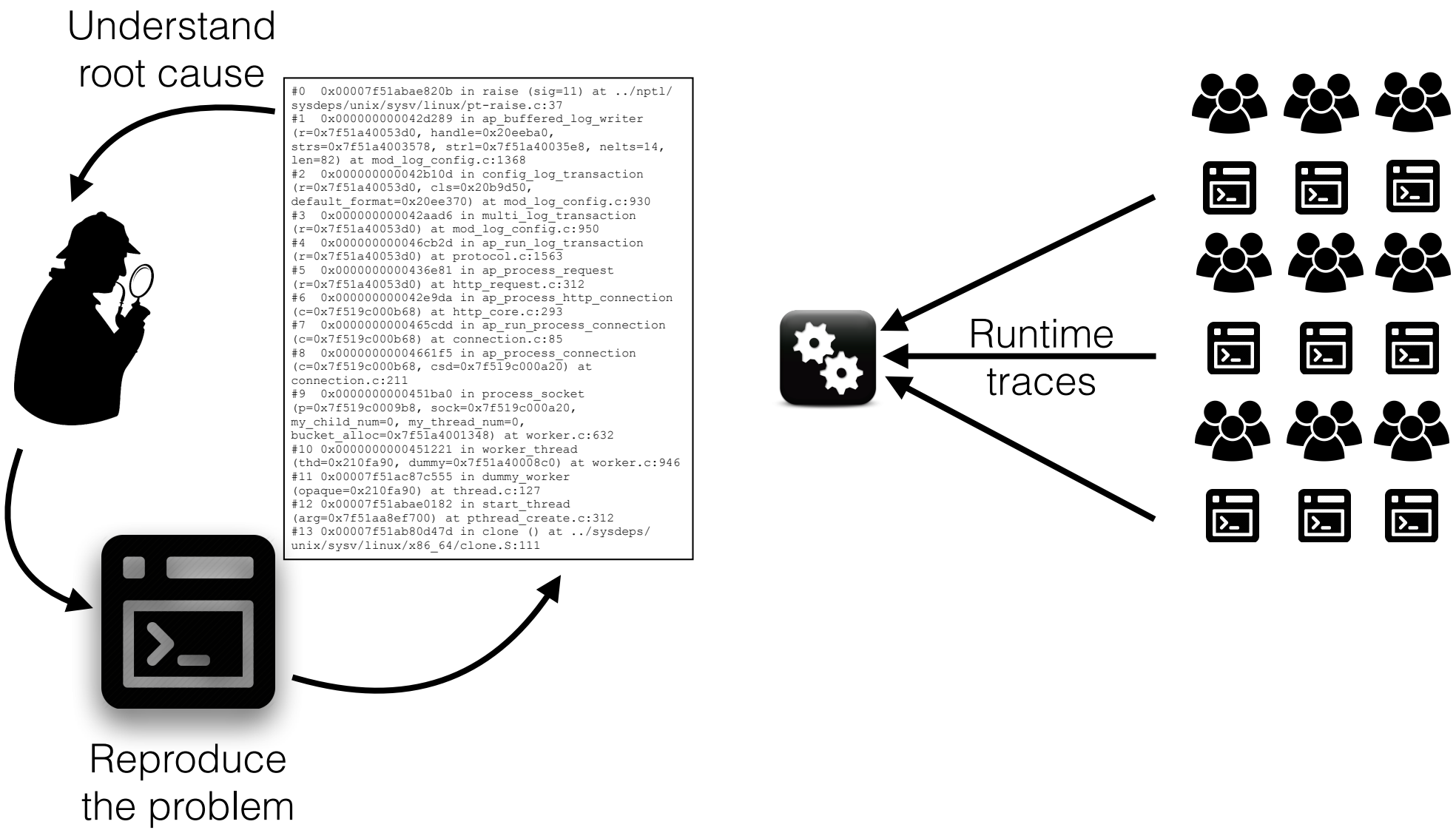
Failure Sketch Use Case



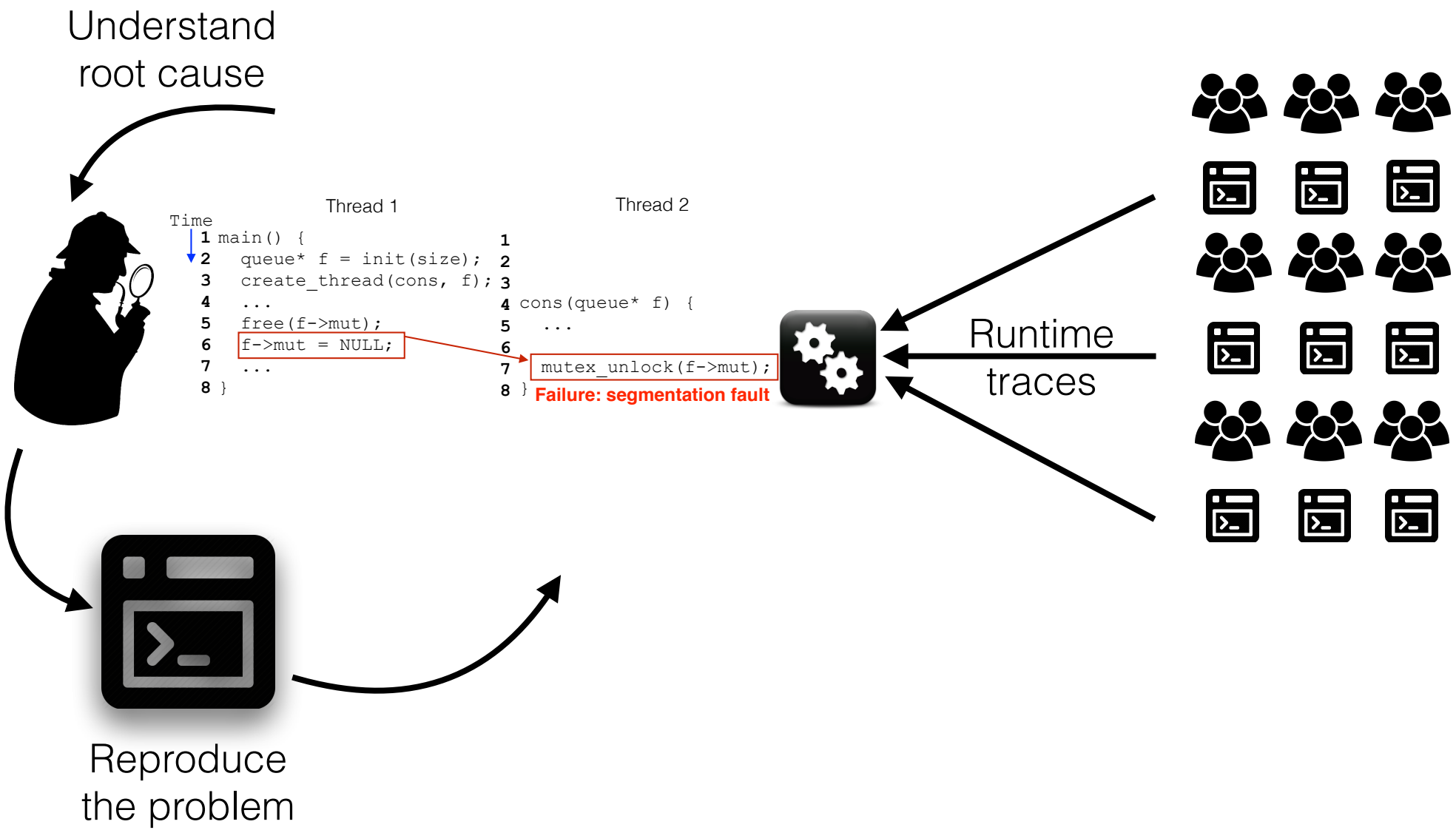
Failure Sketch Use Case



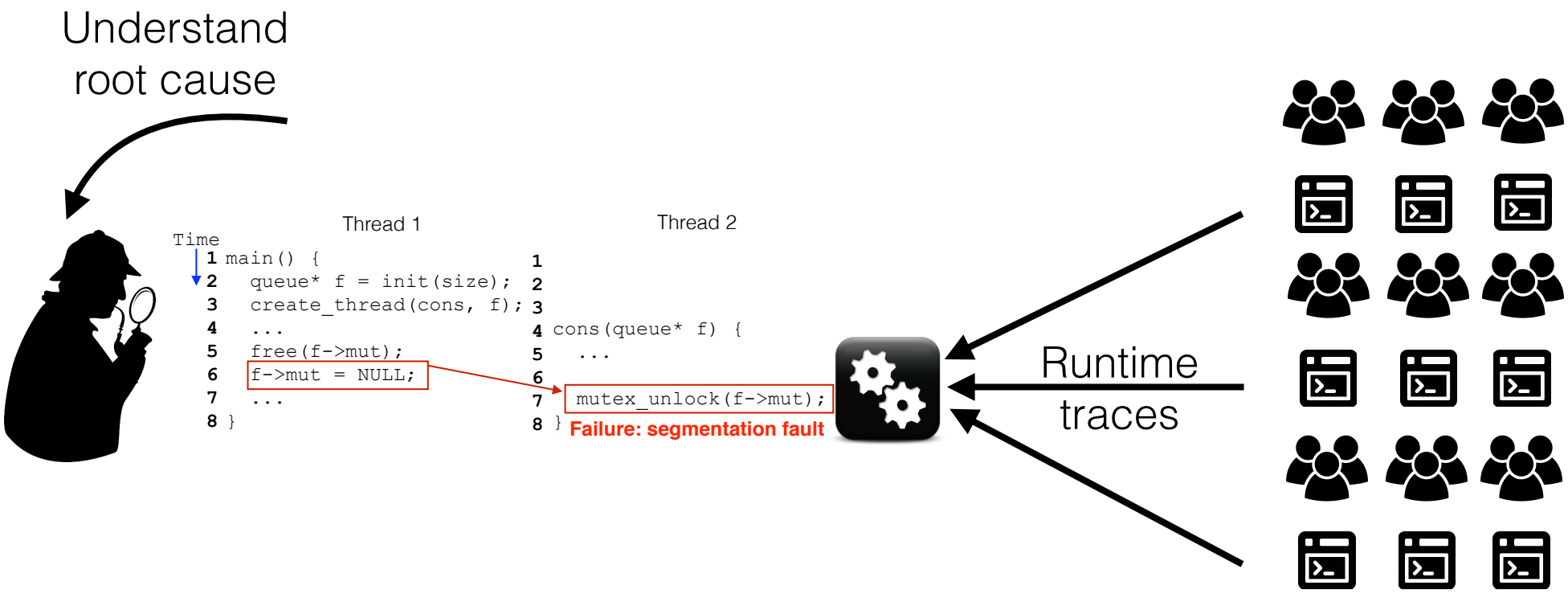
Failure Sketch Use Case



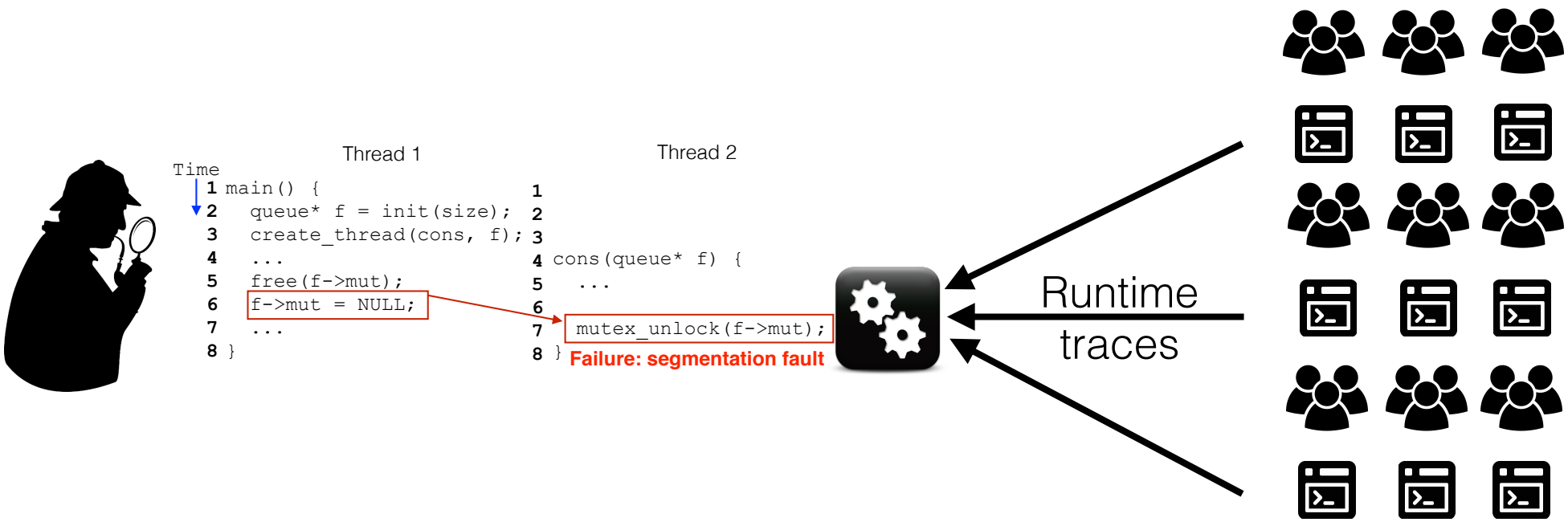
Failure Sketch Use Case



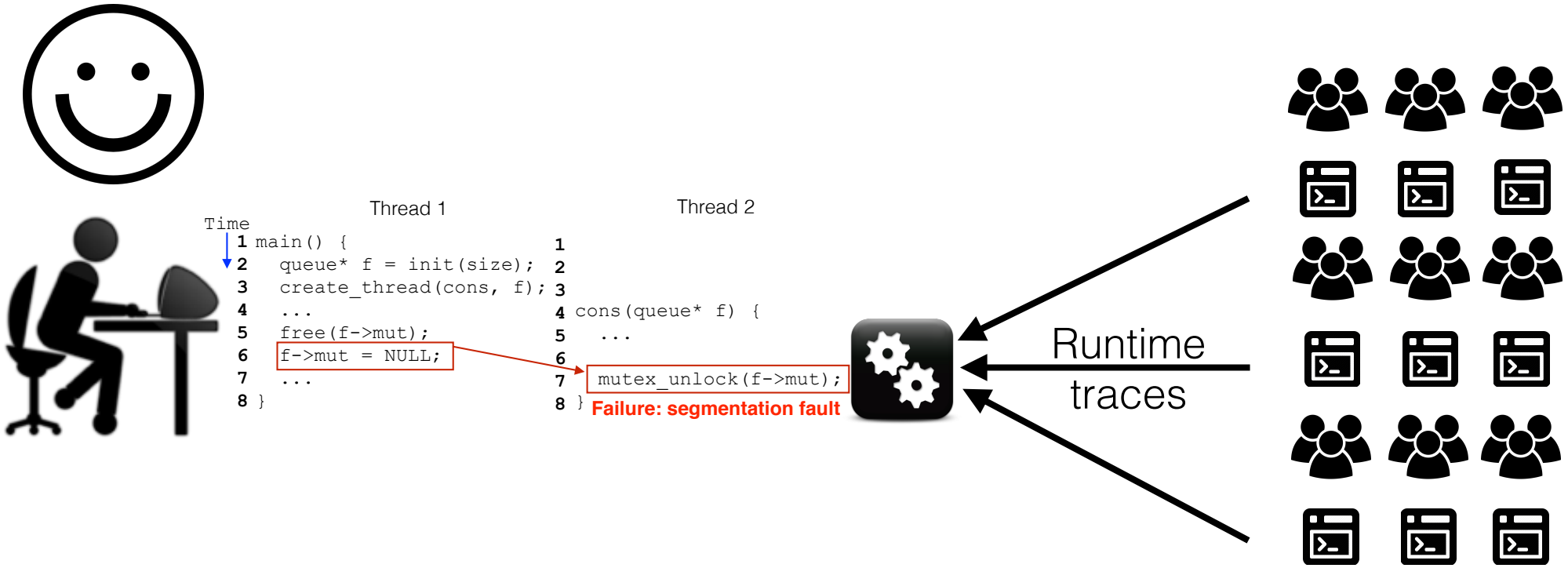
Failure Sketch Use Case



Failure Sketch Use Case



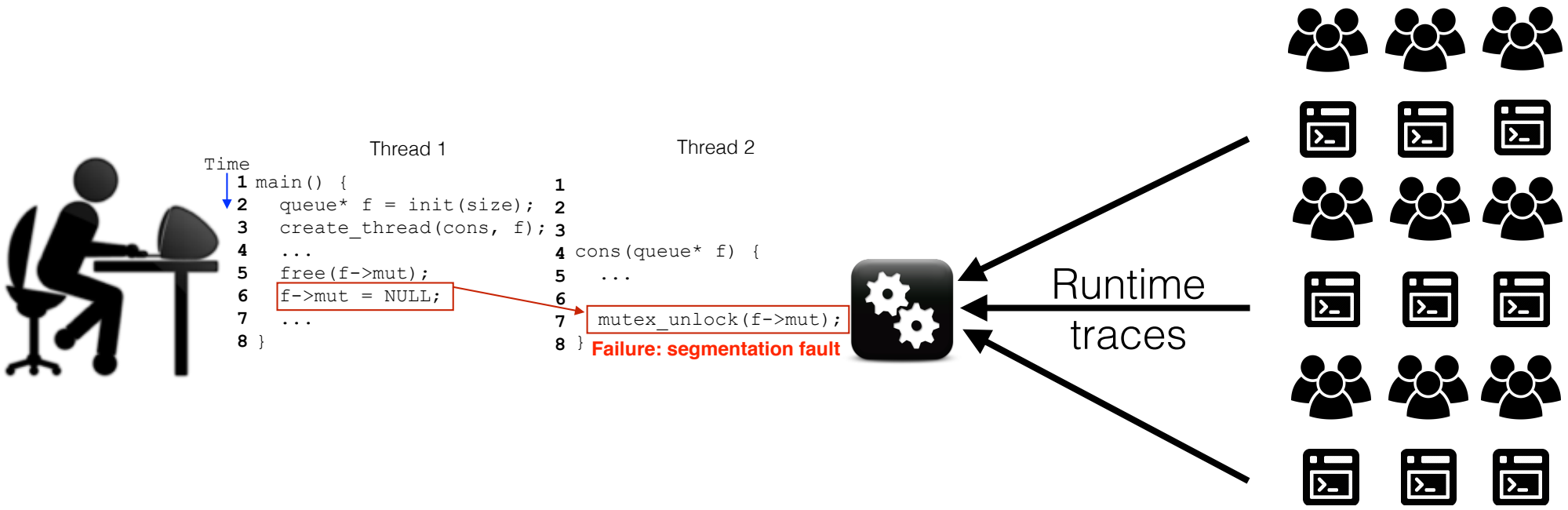
Failure Sketch Use Case



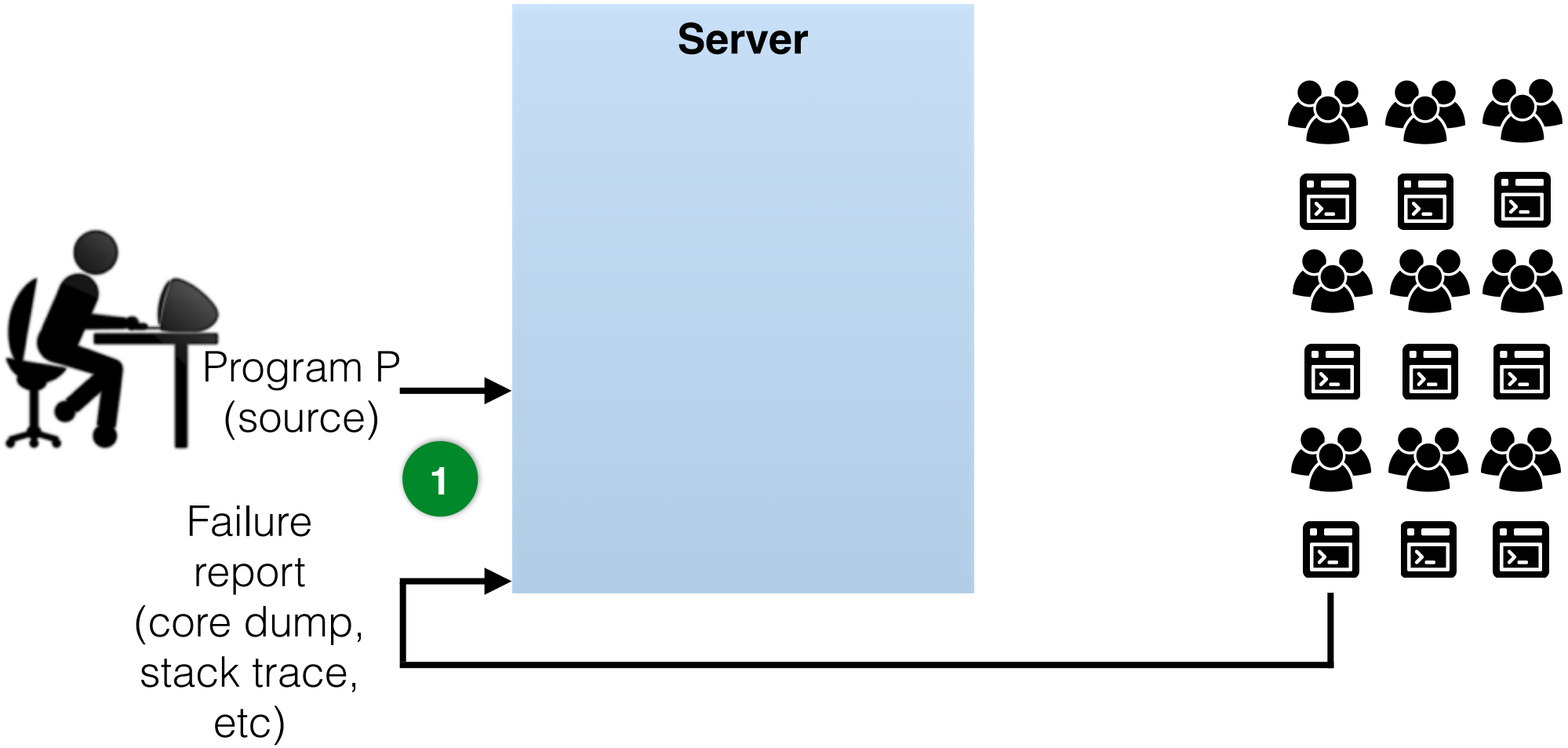
Research Challenges

- Hard-to-reproduce failures
  - Recur only a few times in production
- Accuracy of failure sketches
  - No extraneous elements in the failure sketch
- Latency of failure sketch computation
  - Developers can’t wait forever for failure sketches

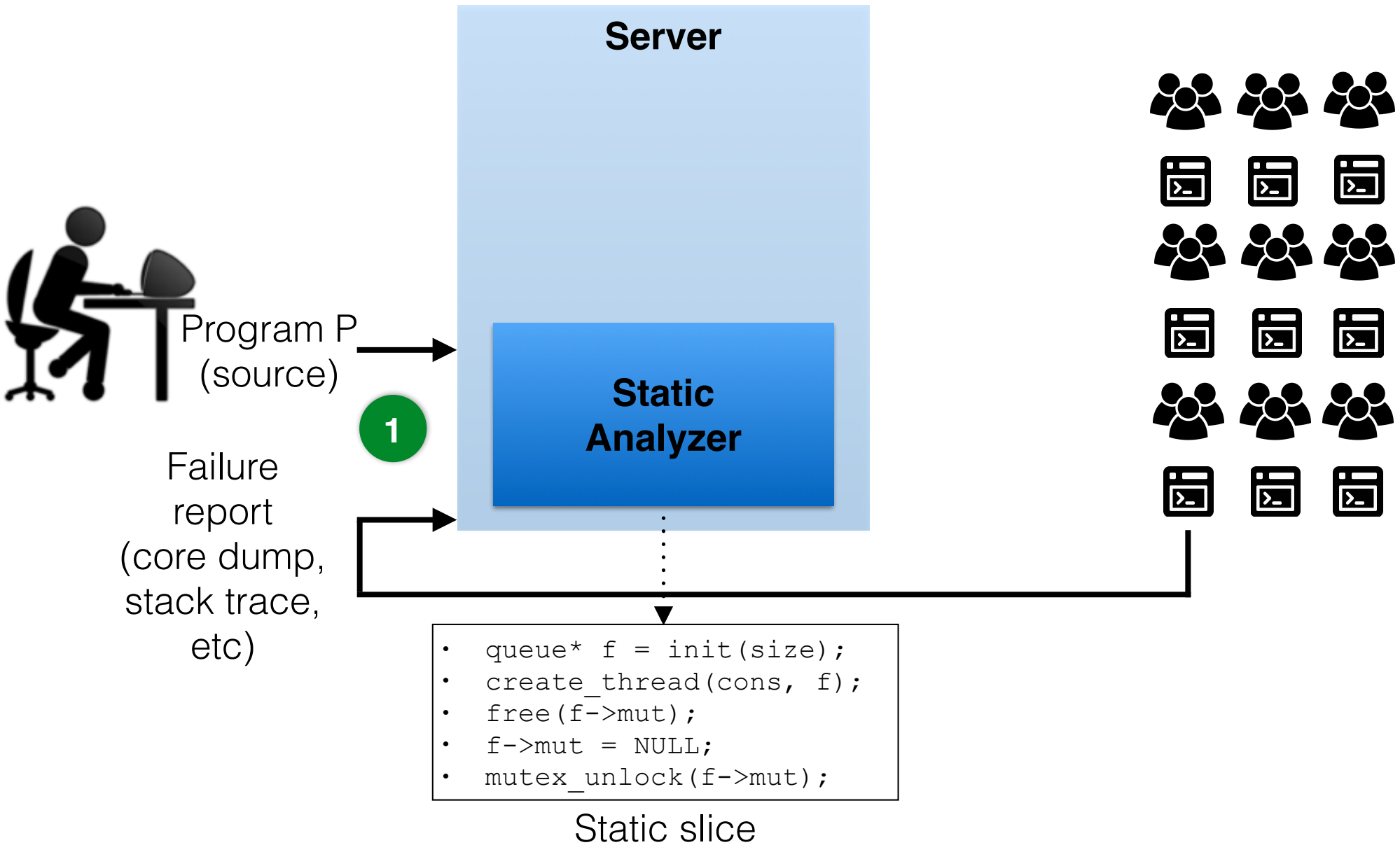
System Architecture



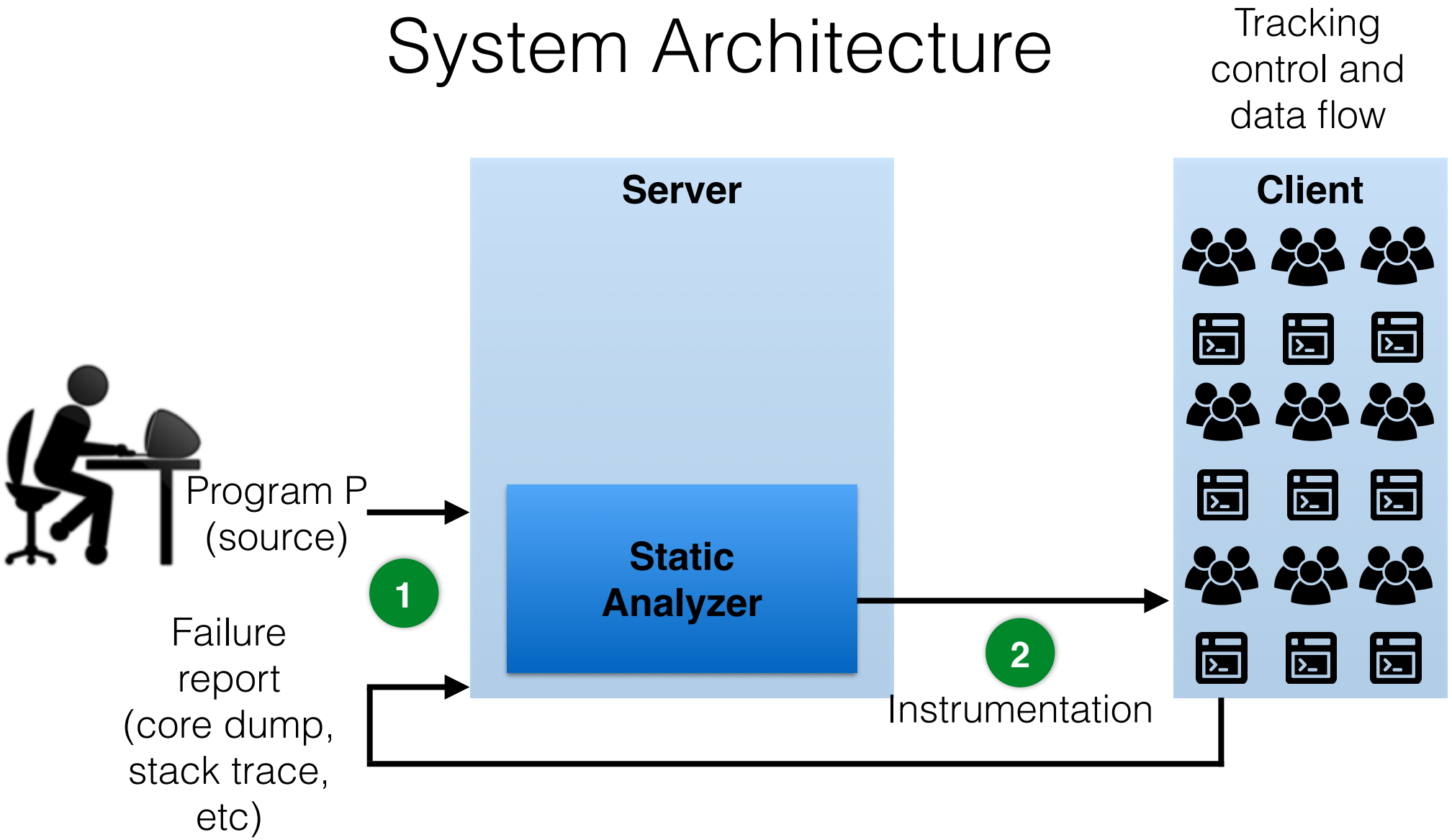
System Architecture



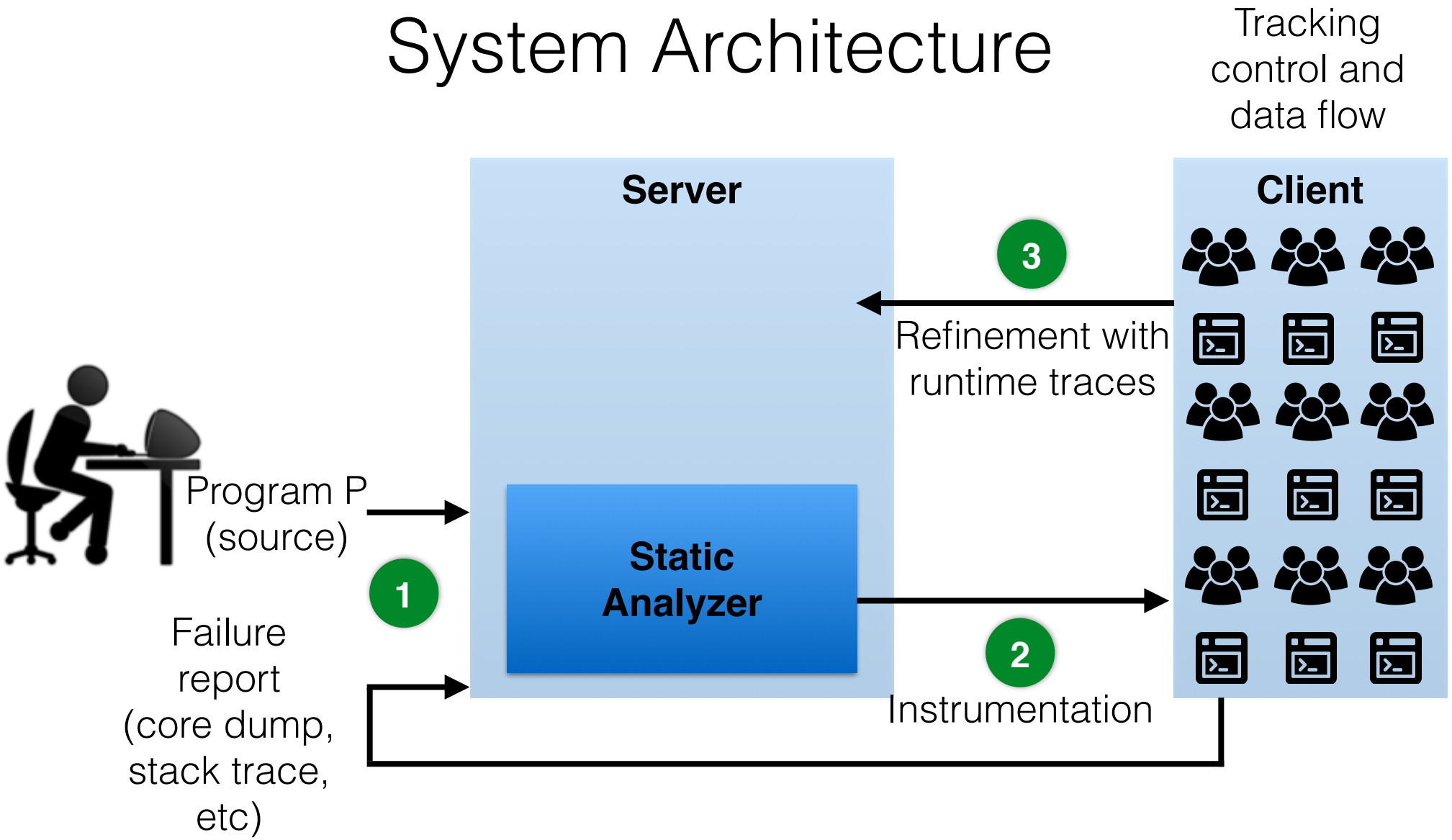
System Architecture



System Architecture



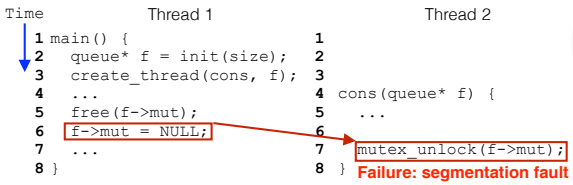
System Architecture



System Architecture

Tracking control and data flow

Failure Sketch



Program P (source)

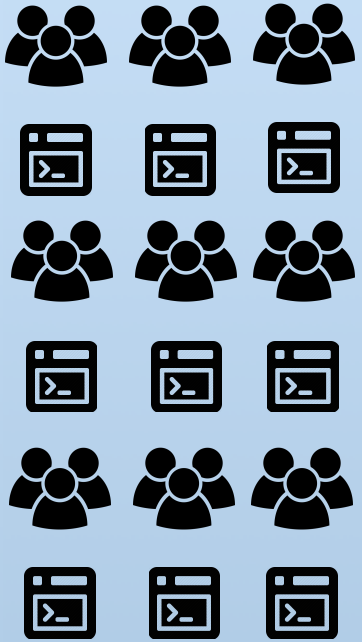
Failure report (core dump, stack trace, etc)

Server

Failure Sketch Computation Engine

Static Analyzer

Client

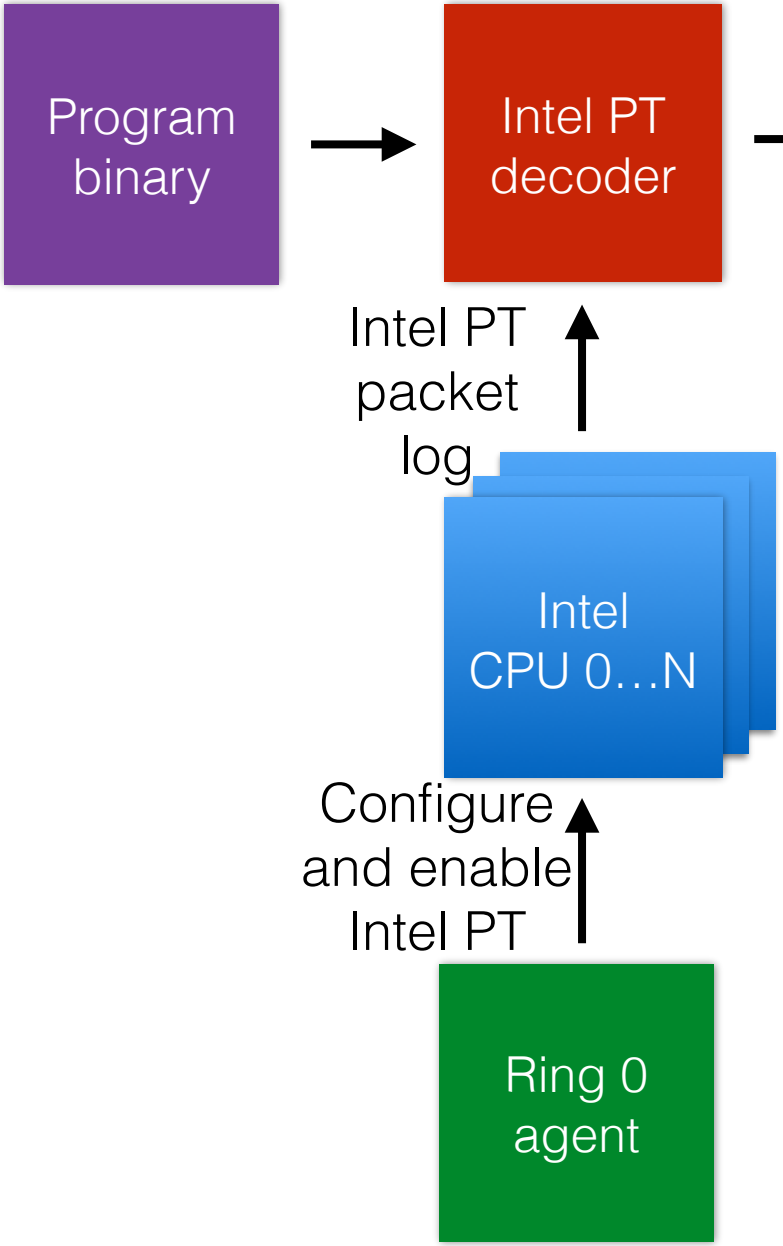


Refinement with runtime traces

Instrumentation

1 2 3 4

Intel Processor Trace (Intel PT)

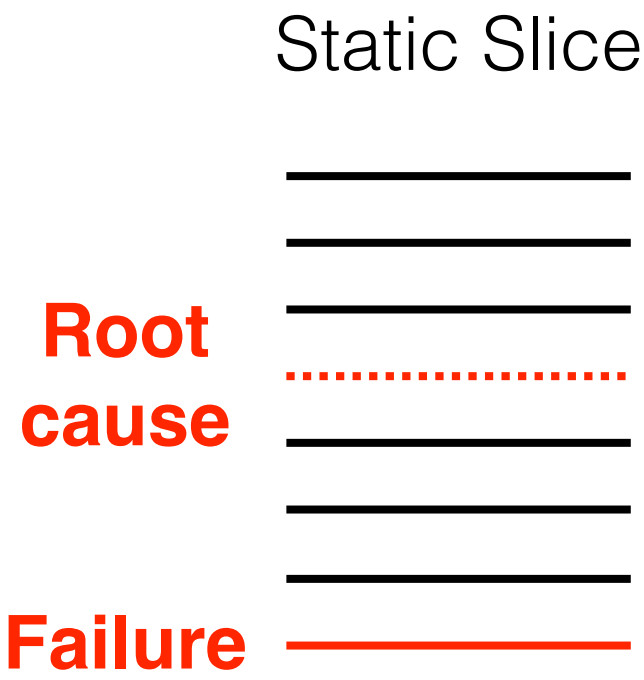


Control flow information

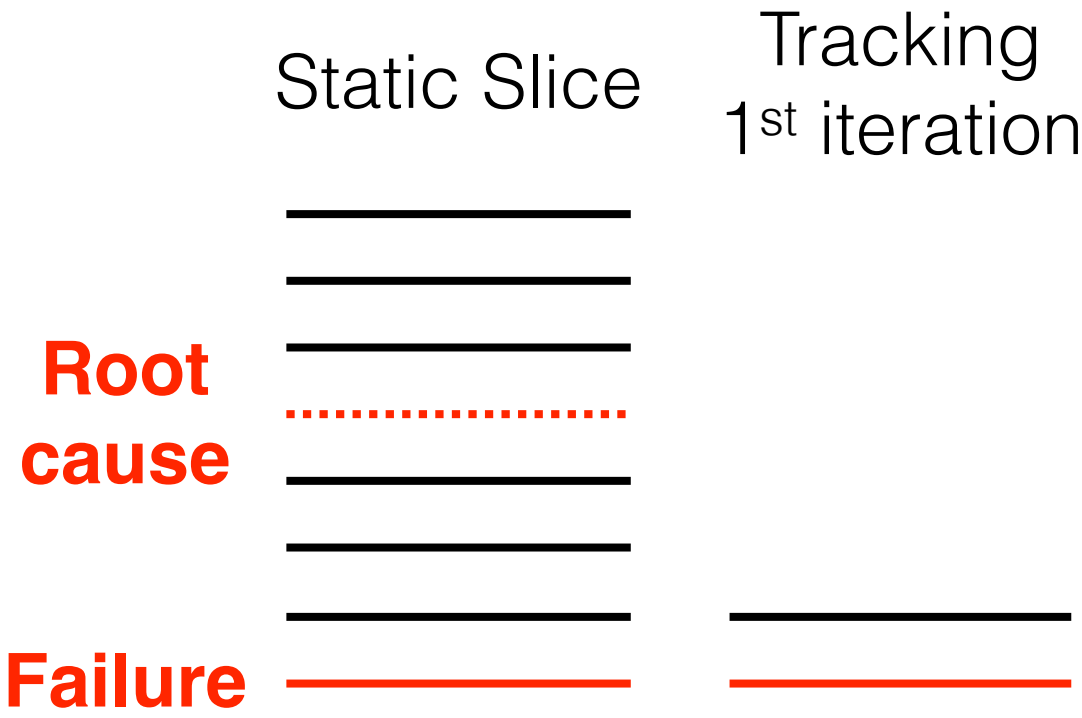
- Compressed trace of branches taken (~1 bit per instruction)
- Low overhead (~40% **full** tracing overhead)

Tracking Control Flow Using Intel PT

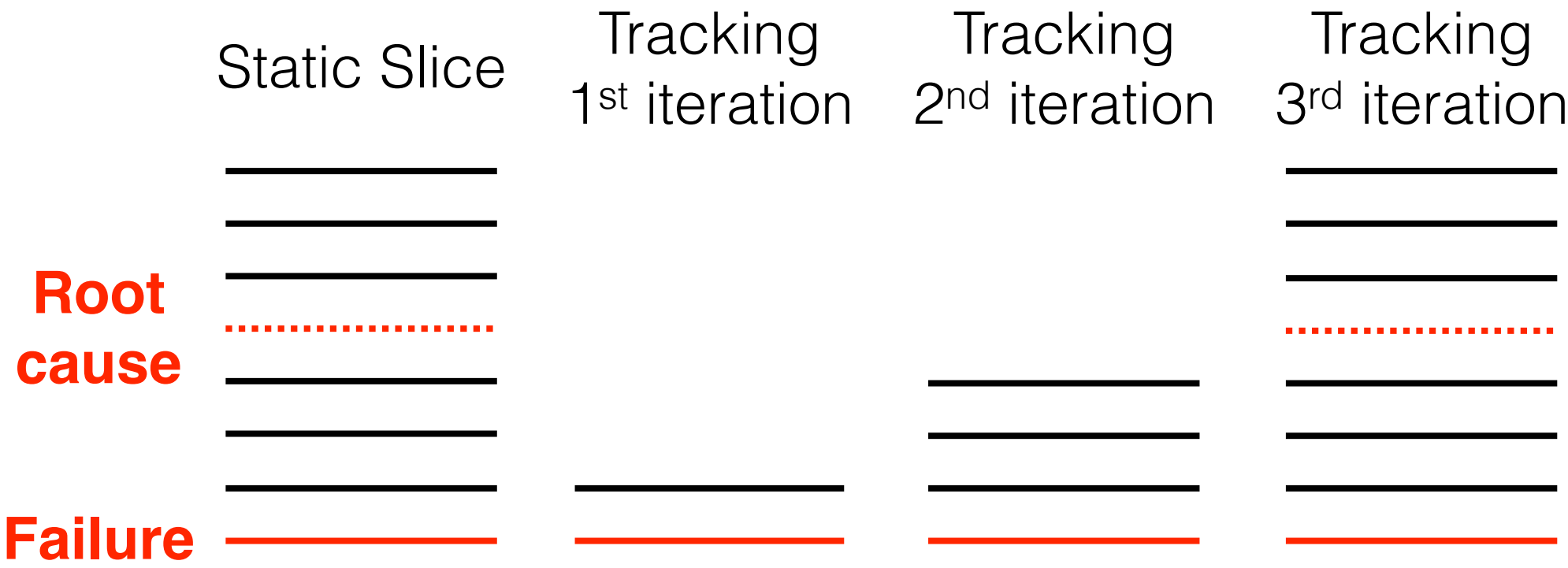
Tracking Control Flow Using Intel PT



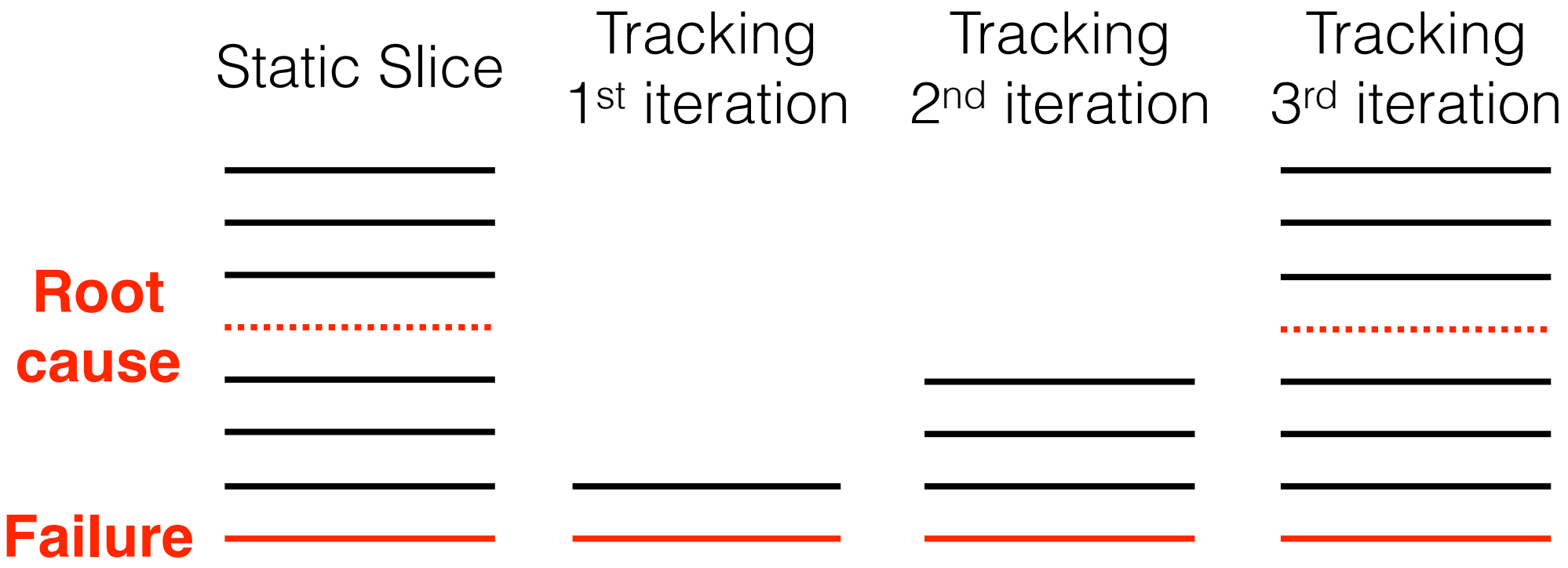
Tracking Control Flow Using Intel PT



Tracking Control Flow Using Intel PT



Tracking Control Flow Using Intel PT



Monitoring small portions of a slice works well because most failures have nearby root causes[1,2]

[1]W. Zhang et al., ConSeq: Detecting concurrency bugs through sequential errors. ASPLOS 2011

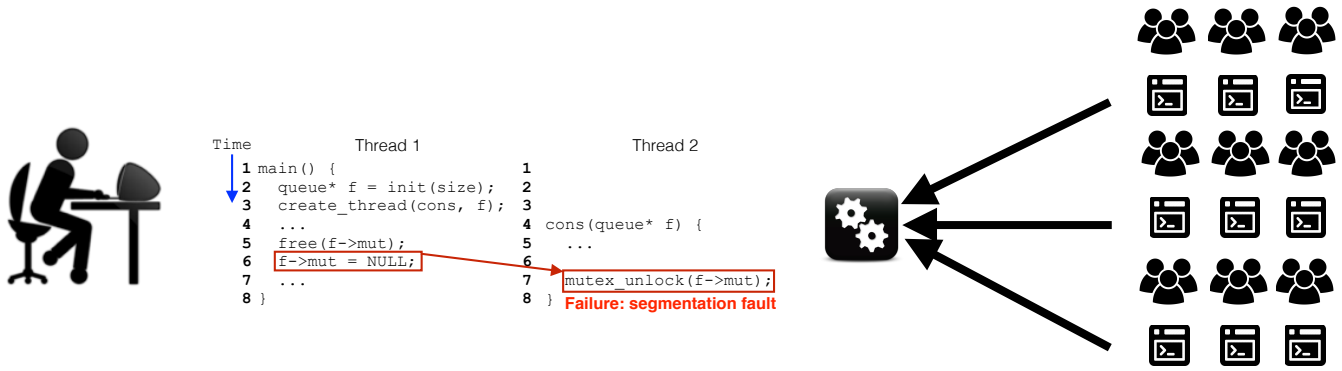
[2]F. Qin et al., Rx: Treating bugs as allergies a safe method to survive software failures. SOSP 2005

Discussion

- Intrusiveness
  - Currently, we do static instrumentation
  - Dynamic instrumentation is less intrusive
- Privacy
  - Use anonymization
  - Forgo data monitoring when privacy requirements are very strict

Future Work

- Diagnosing performance problems
  - Correlating control flow with slowdowns
- Speeding up program analysis
  - Use control flow information to tackle path explosion
- Using failure sketches for test case generation



- Failure sketches
  - Summary explaining failure root causes
- Application of hardware-based monitoring
  - Enabler for building failure sketches
  - Many potential use cases