SILEXICA

SLX Analytics & Testing Platform

Benjamin Goldschmidt

goldschmidt@silexica.com

Stuttgart, 12.12.2019

Silexica Facts

RWTH AACHEN UNIVERSITY

Est. 2014 after a decade of research

Team of world leading software and hardware experts

60 people worldwide, engineering HQ in Germany

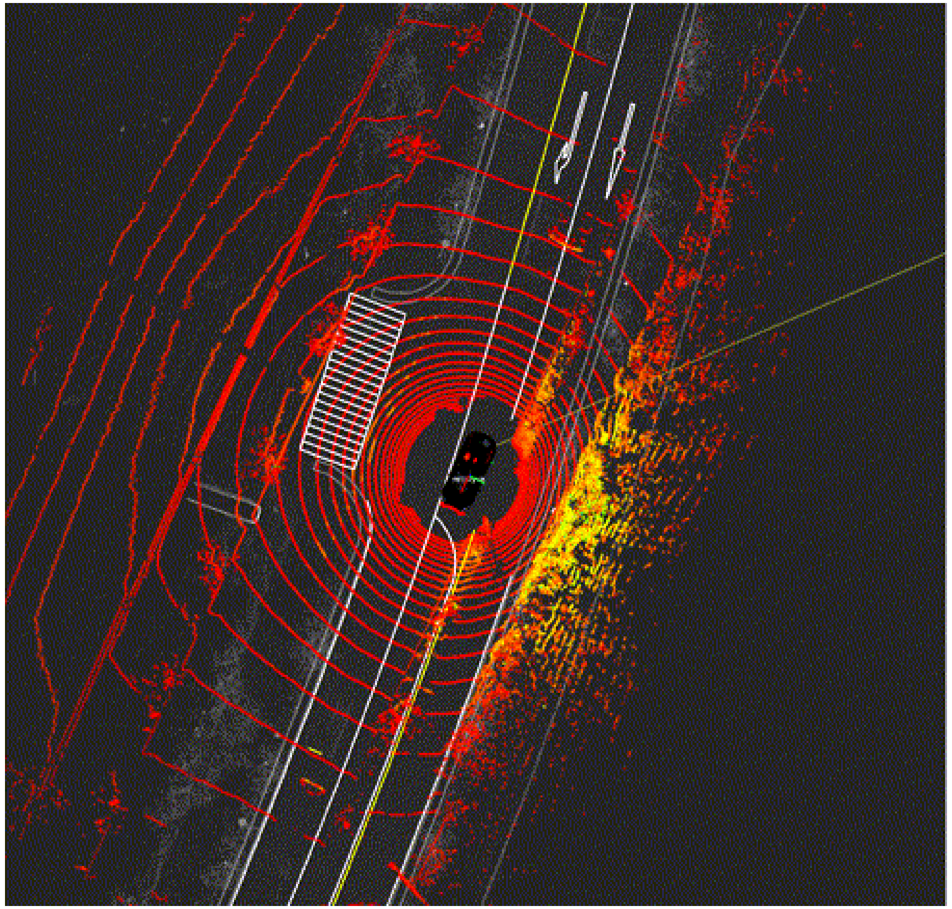
3 offices and worldwide local support engineers

mazda HITACHI Inspire the Next HUAWEI FUJITSU MBDA DENSO THALES RICOH

EUROPE
USA
JAPAN

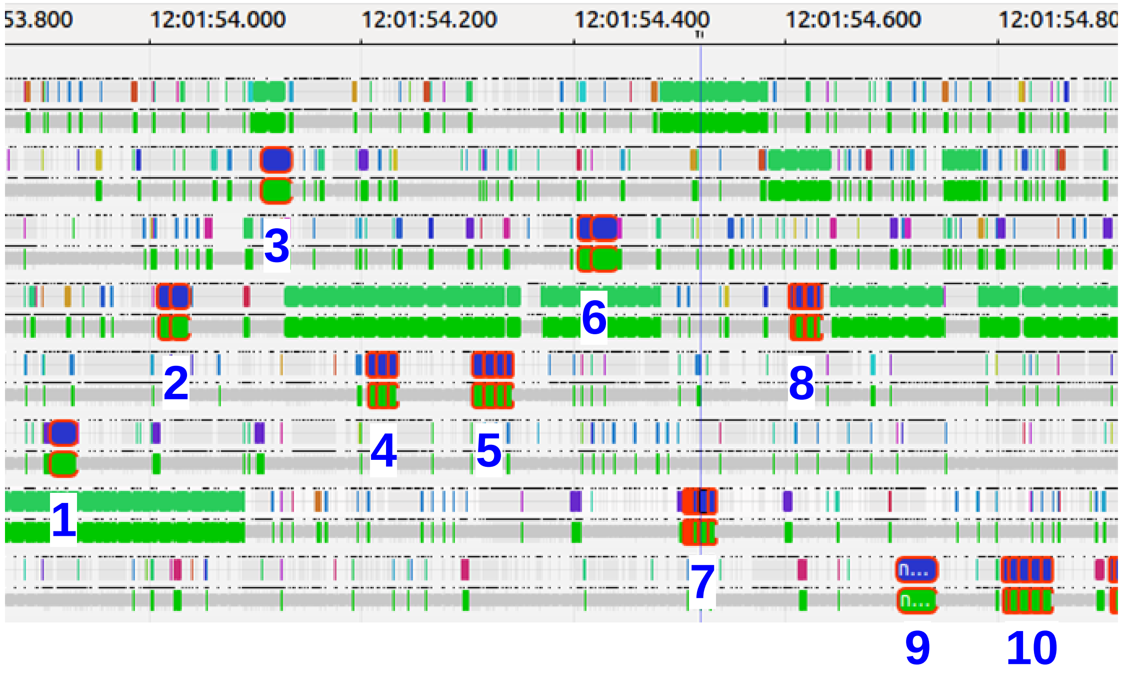
Motivation

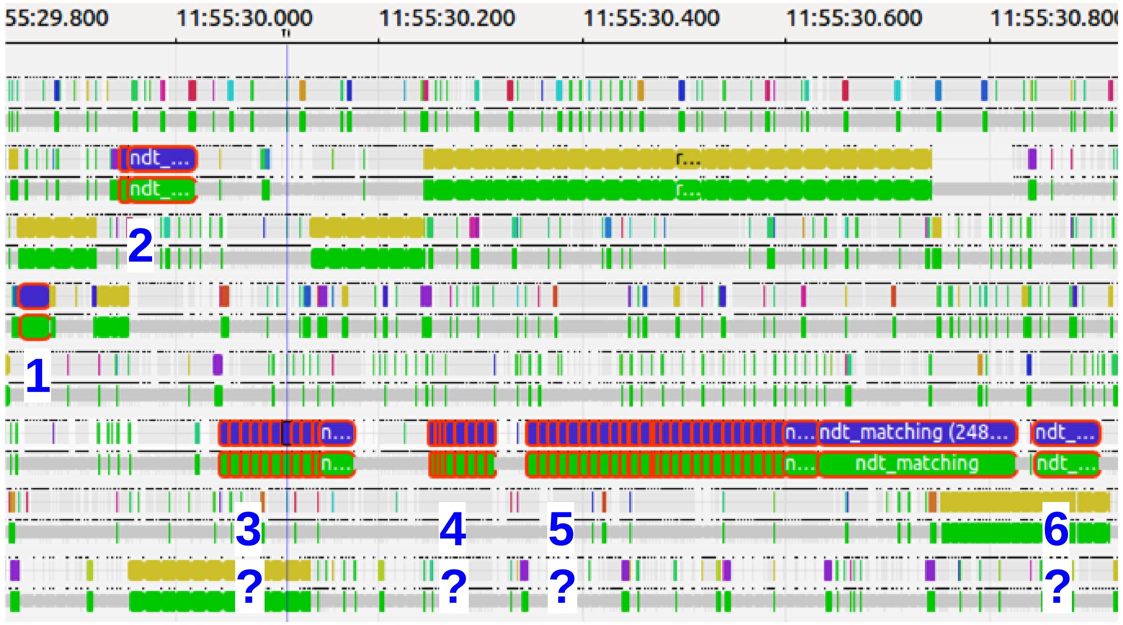
- Autoware ROS-based AD stack (Moriyama example, v1.8)
- Problem: Positioning of the car is sometimes unstable
- Many active modules in the example + large codebase:
  - Where to start?!
  - How to make behavior reproducible!?



Motivation

Two Runs of the Same Scenario





Successful run:

- Positioning module (marked red) runs in regular intervals with acceptable runtime variation

Unsuccessful run:

- Positioning module (marked red) does not run in regular time intervals and has a highly variable runtime

Problem Statement

Missing system overview → Engineers are fixing, not developing

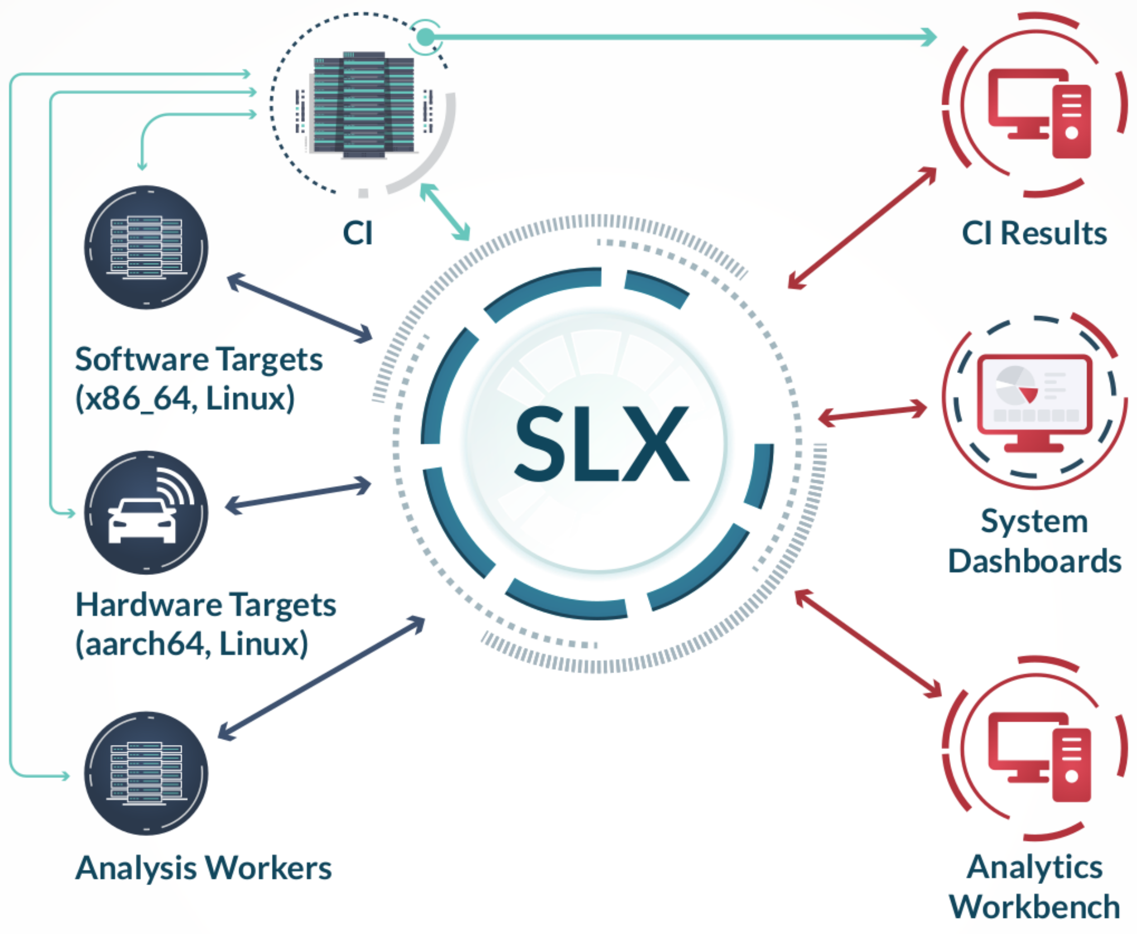
No statistical profiling of system → High risk of (creeping) system degradation

No automated testing of system metrics → Last-minute fire-fighting / delay of release

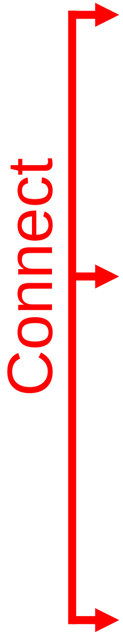
No scalability of testing solution → Limited test coverage to achieve high release rate

SLX Analytics & Testing Platform

- Automated outlier detection:
  - Multi-level/run tracing + analysis
  - CI integration
- Interactive root cause analysis
- Highly scalable:
  - Cloud, on-premise, desktop
  - Open API

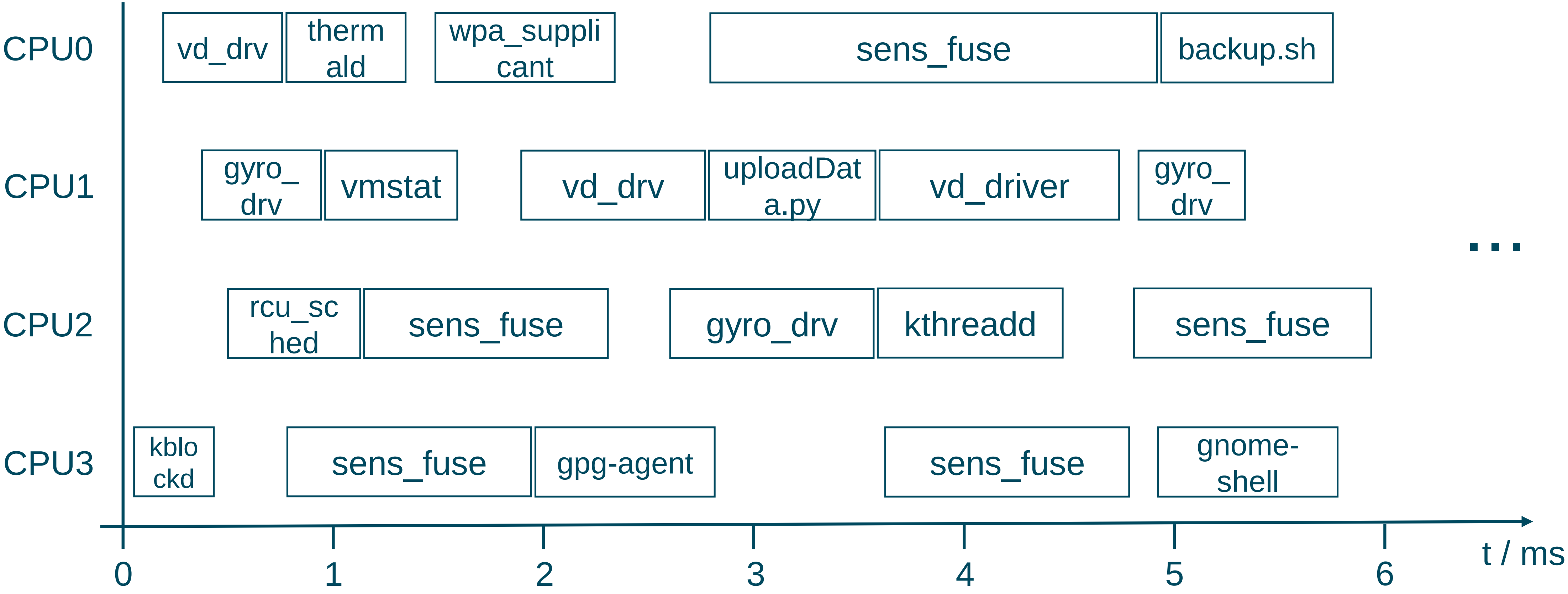


Multi-Level Analysis



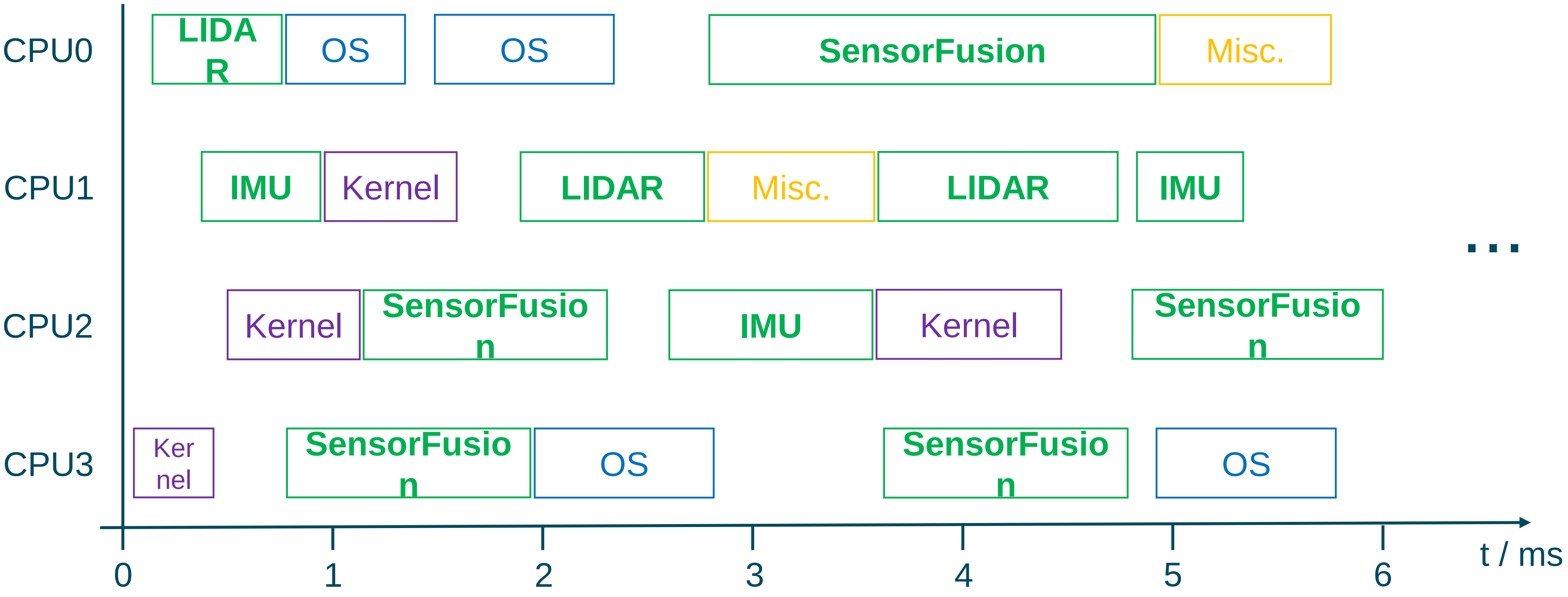
- System-level:
  - Acquire kernel/system events → global timeline of the system
- Middleware-level (e.g. ROS{1, 2}, Adaptive AUTOSAR, etc.):
  - Acquire middleware-specific data (e.g. from the communication stack) to provide semantic context
- Module-level:
  - Acquire module-specific data (static/dynamic) for in-depth analysis

Multi-Level Analysis



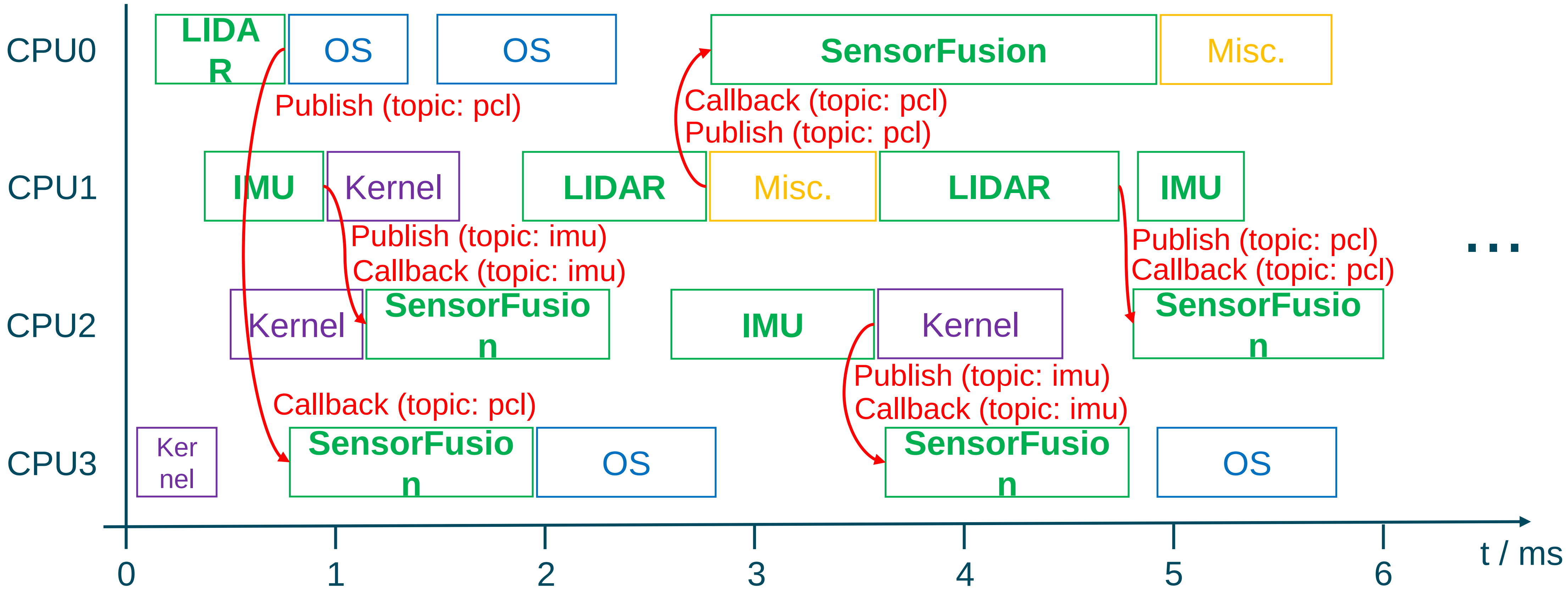
System-level data: Global CPU timeline (proccesses)

Multi-Level Analysis



Use middleware-level data to highlight relevant processes

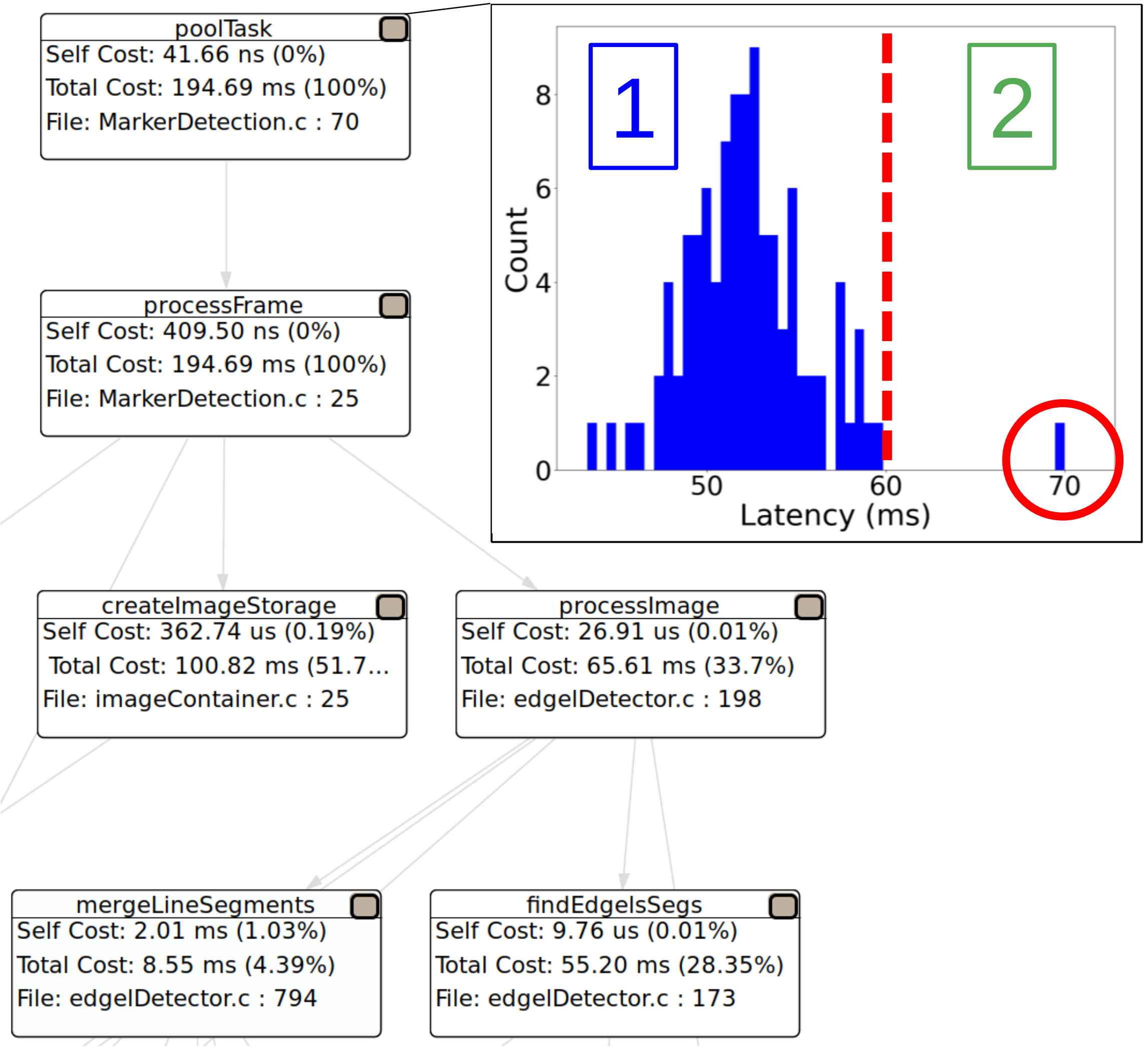
Multi-Level Analysis



Add middleware communication events to global timeline

Multi-Run Analysis

- Run test scenarios multiple times to detect outliers
- Configurable constraints for automated tests
- Full results history:
  - Compare code revisions
  - Compare same/different inputs



poolTask
Self Cost: 41.66 ns (0%)
Total Cost: 194.69 ms (100%)
File: MarkerDetection.c : 70

processFrame
Self Cost: 409.50 ns (0%)
Total Cost: 194.69 ms (100%)
File: MarkerDetection.c : 25

createImageStorage
Self Cost: 362.74 us (0.19%)
Total Cost: 100.82 ms (51.7...
File: imageContainer.c : 25

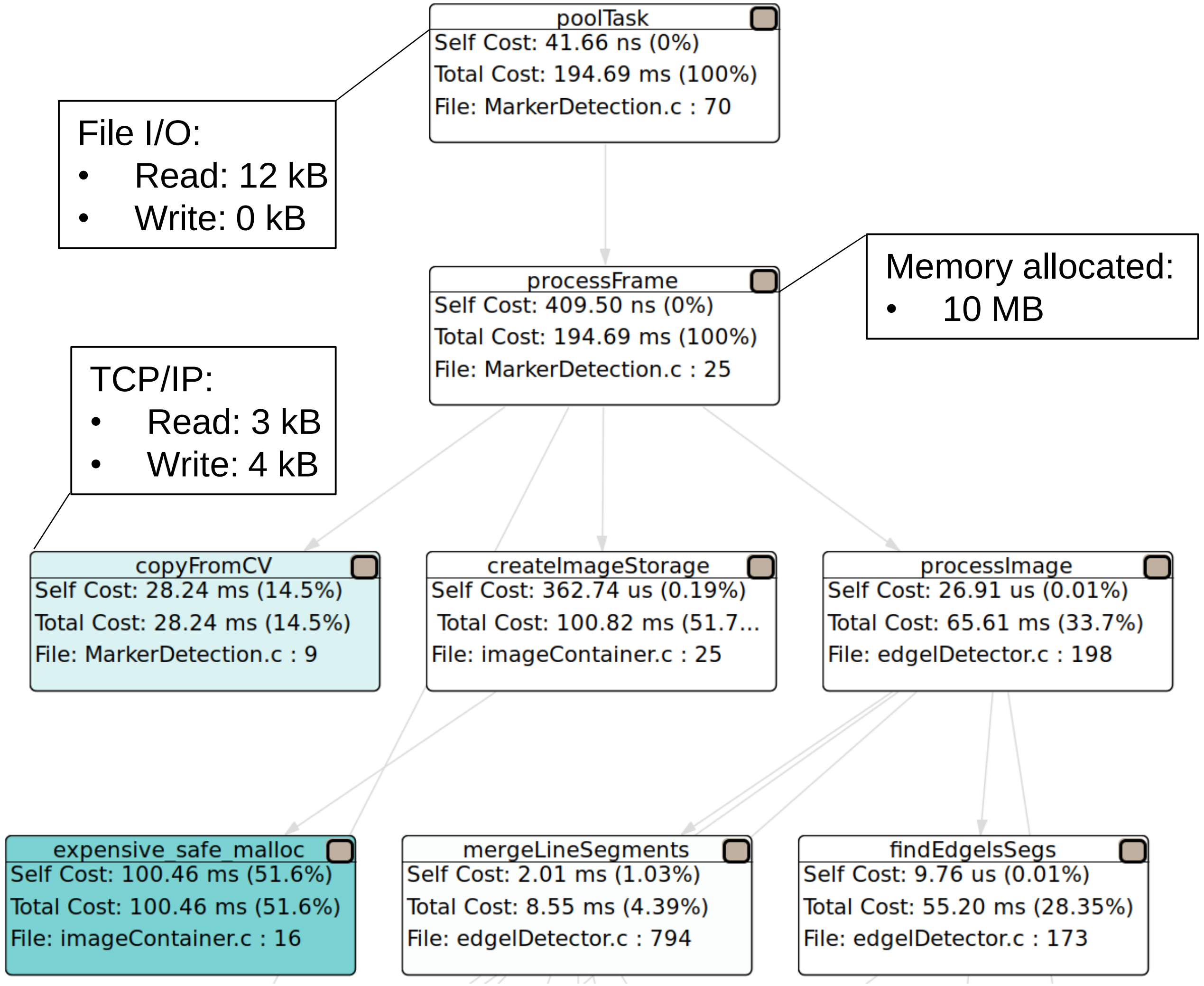
processImage
Self Cost: 26.91 us (0.01%)
Total Cost: 65.61 ms (33.7%)
File: edgelDetector.c : 198

mergeLineSegments
Self Cost: 2.01 ms (1.03%)
Total Cost: 8.55 ms (4.39%)
File: edgelDetector.c : 794

findEdgelsSegs
Self Cost: 9.76 us (0.01%)
Total Cost: 55.20 ms (28.35%)
File: edgelDetector.c : 173

Module-Level Analysis

- Detect memory allocations after startup phase
- Detect system calls with difficult to predict runtime behavior
- Detect TCP, UDP, and middleware communication



poolTask
Self Cost: 41.66 ns (0%)
Total Cost: 194.69 ms (100%)
File: MarkerDetection.c : 70

File I/O:
- Read: 12 kB
- Write: 0 kB

processFrame
Self Cost: 409.50 ns (0%)
Total Cost: 194.69 ms (100%)
File: MarkerDetection.c : 25

Memory allocated:
- 10 MB

TCP/IP:
- Read: 3 kB
- Write: 4 kB

copyFromCV
Self Cost: 28.24 ms (14.5%)
Total Cost: 28.24 ms (14.5%)
File: MarkerDetection.c : 9

createImageStorage
Self Cost: 362.74 us (0.19%)
Total Cost: 100.82 ms (51.7...
File: imageContainer.c : 25

processImage
Self Cost: 26.91 us (0.01%)
Total Cost: 65.61 ms (33.7%)
File: edgeIDetector.c : 198

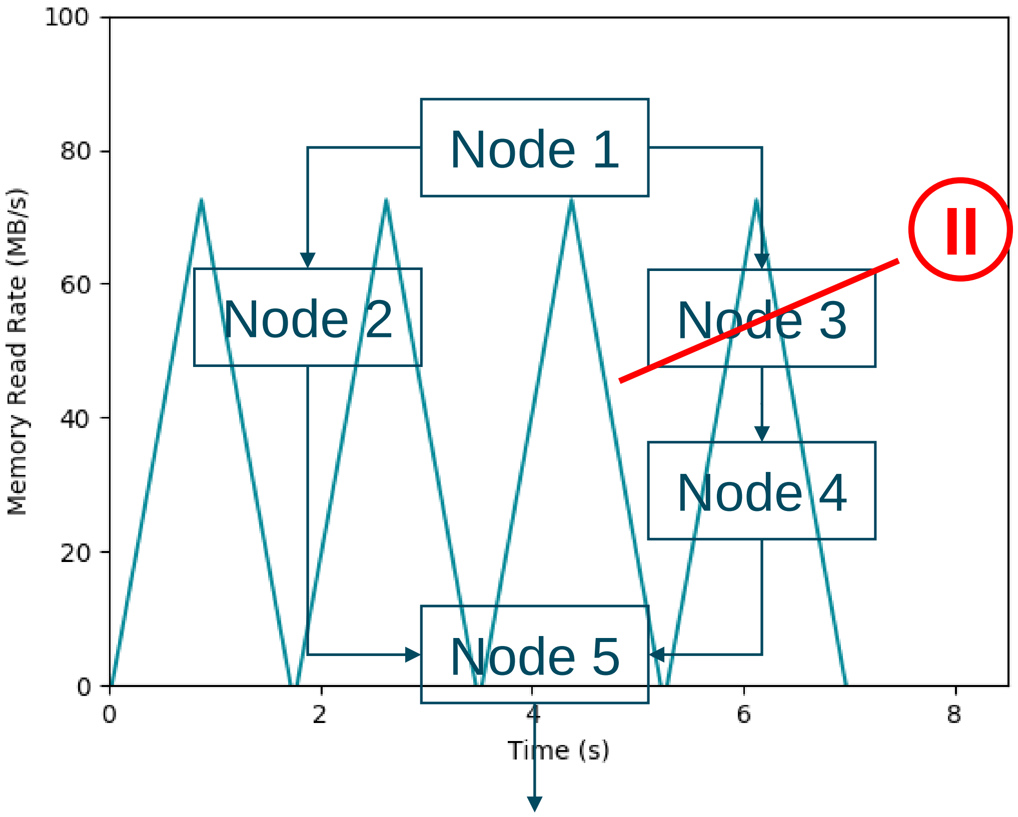
expensive_safe_malloc
Self Cost: 100.46 ms (51.6%)
Total Cost: 100.46 ms (51.6%)
File: imageContainer.c : 16

mergeLineSegments
Self Cost: 2.01 ms (1.03%)
Total Cost: 8.55 ms (4.39%)
File: edgeIDetector.c : 794

findEdgelsSegs
Self Cost: 9.76 us (0.01%)
Total Cost: 55.20 ms (28.35%)
File: edgeIDetector.c : 173

Stress Tests

- Highly configurable stress tests: CPU, memory, cache, etc.
- Loadable/storable stress profiles (stress over time)
- Chaos engineering: Randomly stop processes during execution

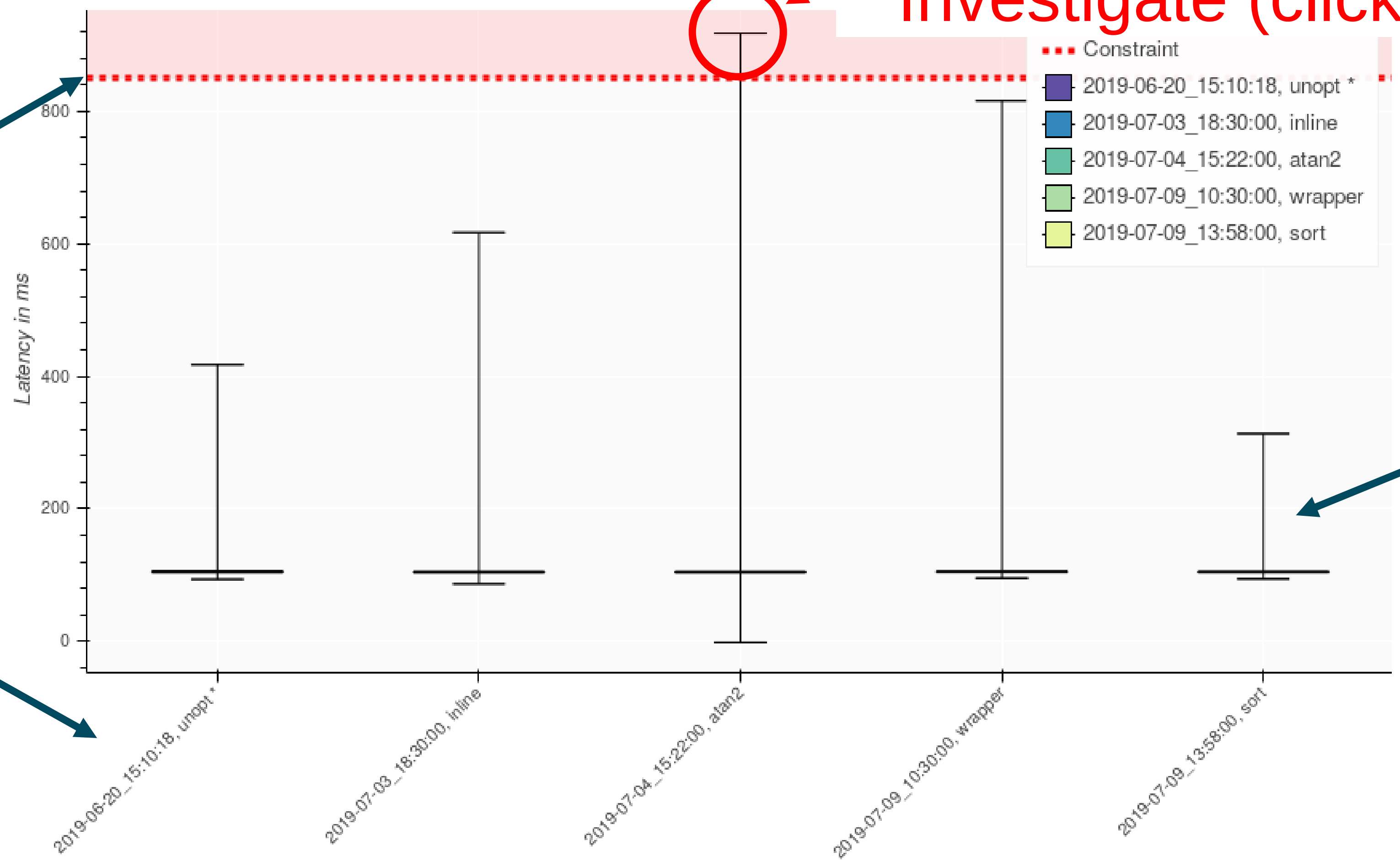


Interactive Root Cause Analysis

Latency Test History

Constraint violated!
Investigate (click!)

Constraint:
850 ms



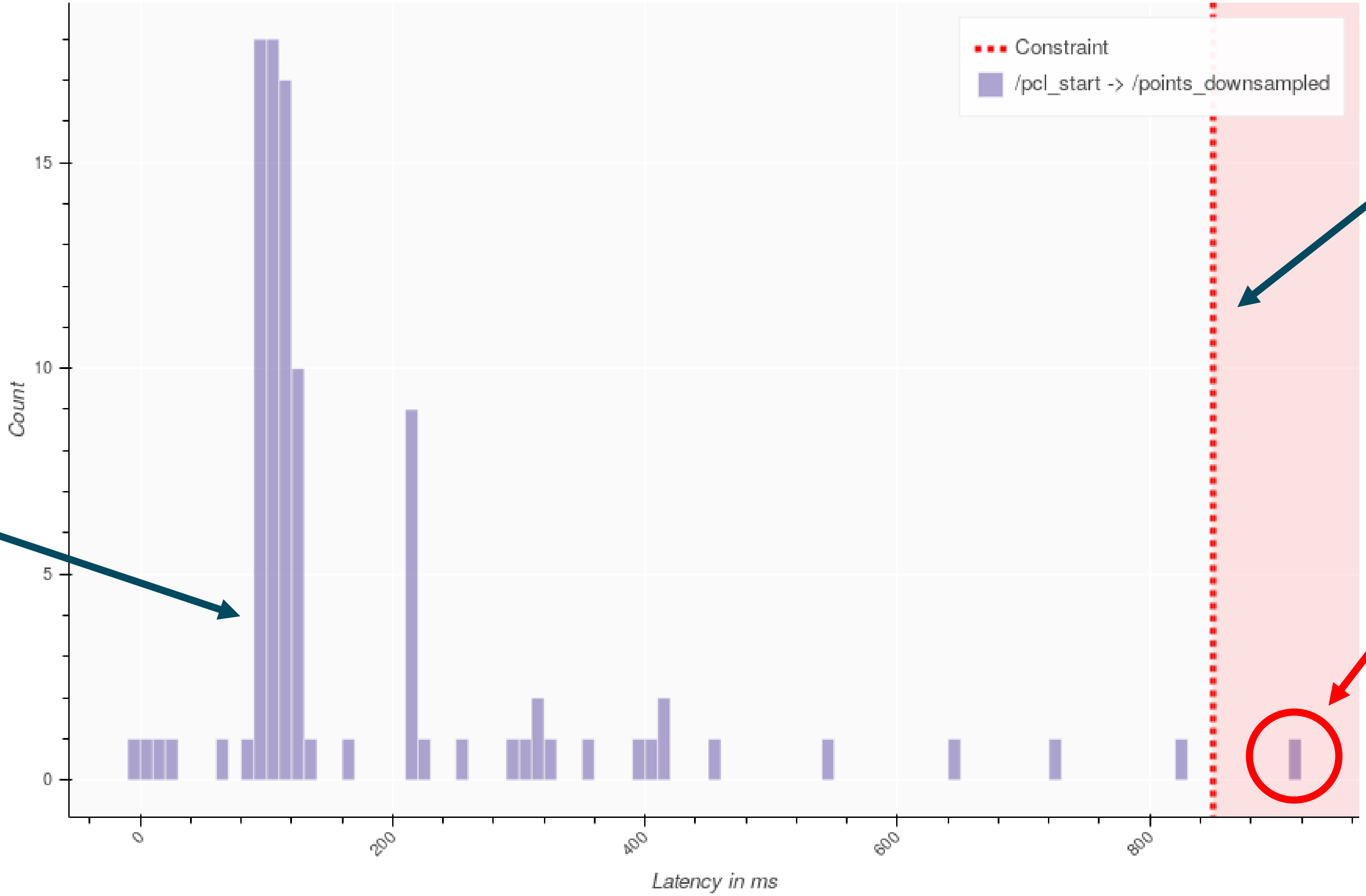
Measured
latency
distributions

Commits

Level 1

Interactive Root Cause Analysis

Latency Histogram



Measured latencies

Constraint: 850 ms

Investigate further (click!)

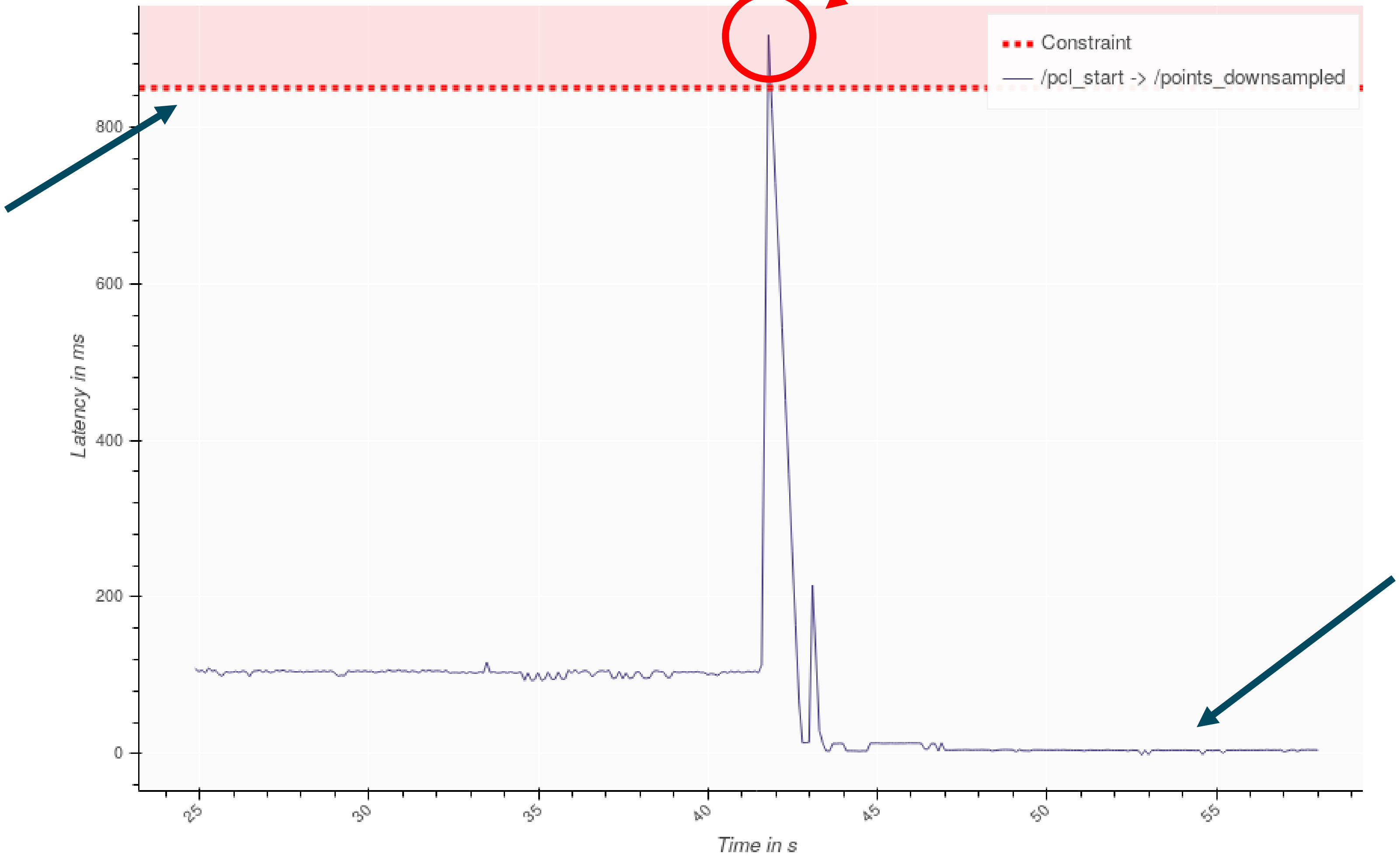
Level 2

Interactive Root Cause Analysis

Latency Over Time

Investigate further (click!)

Constraint:
850 ms

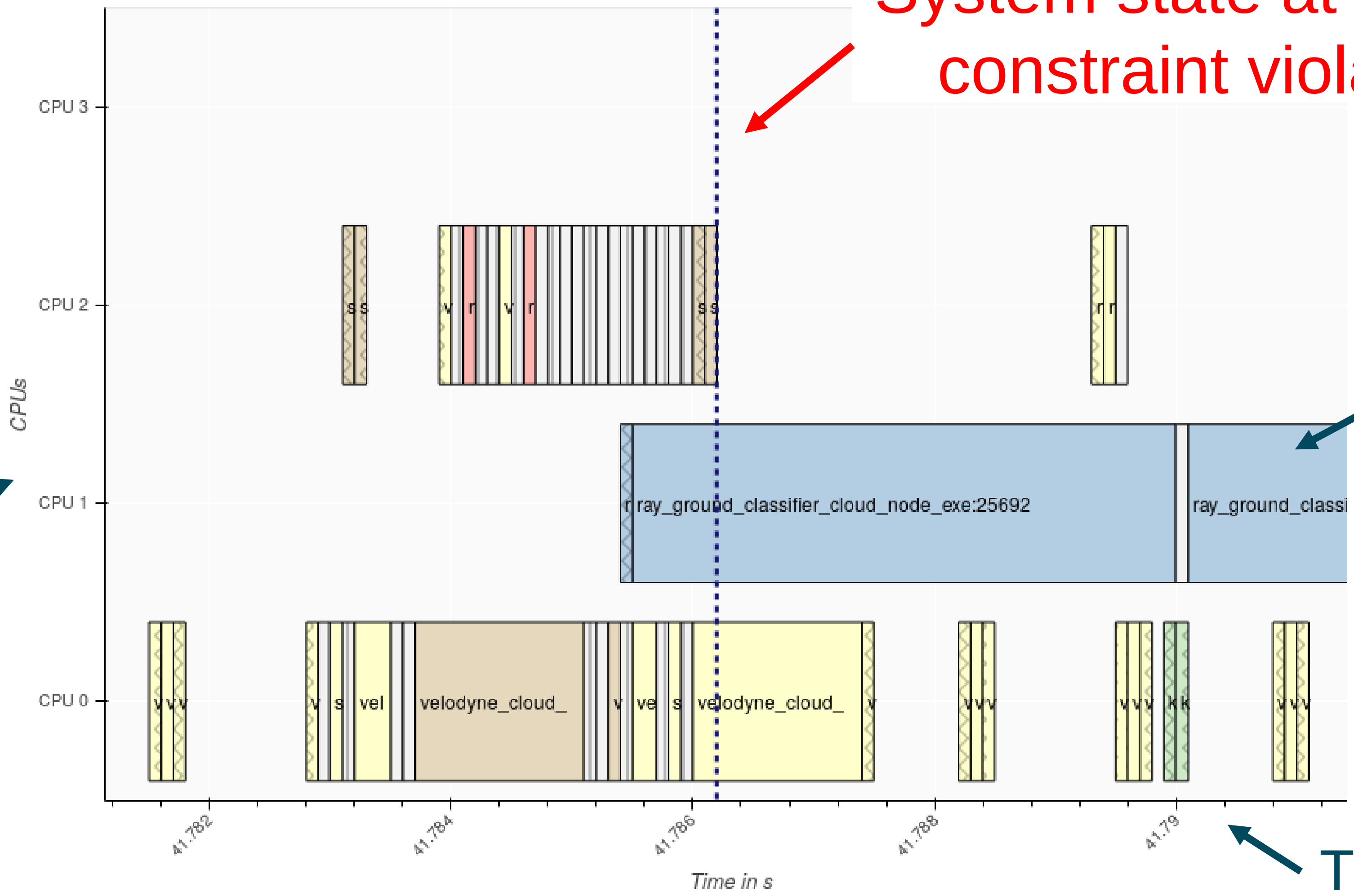


Measured
latency
over time

Level 3

Interactive Root Cause Analysis

CPU Gantt Chart



System state at time of constraint violation

Processes

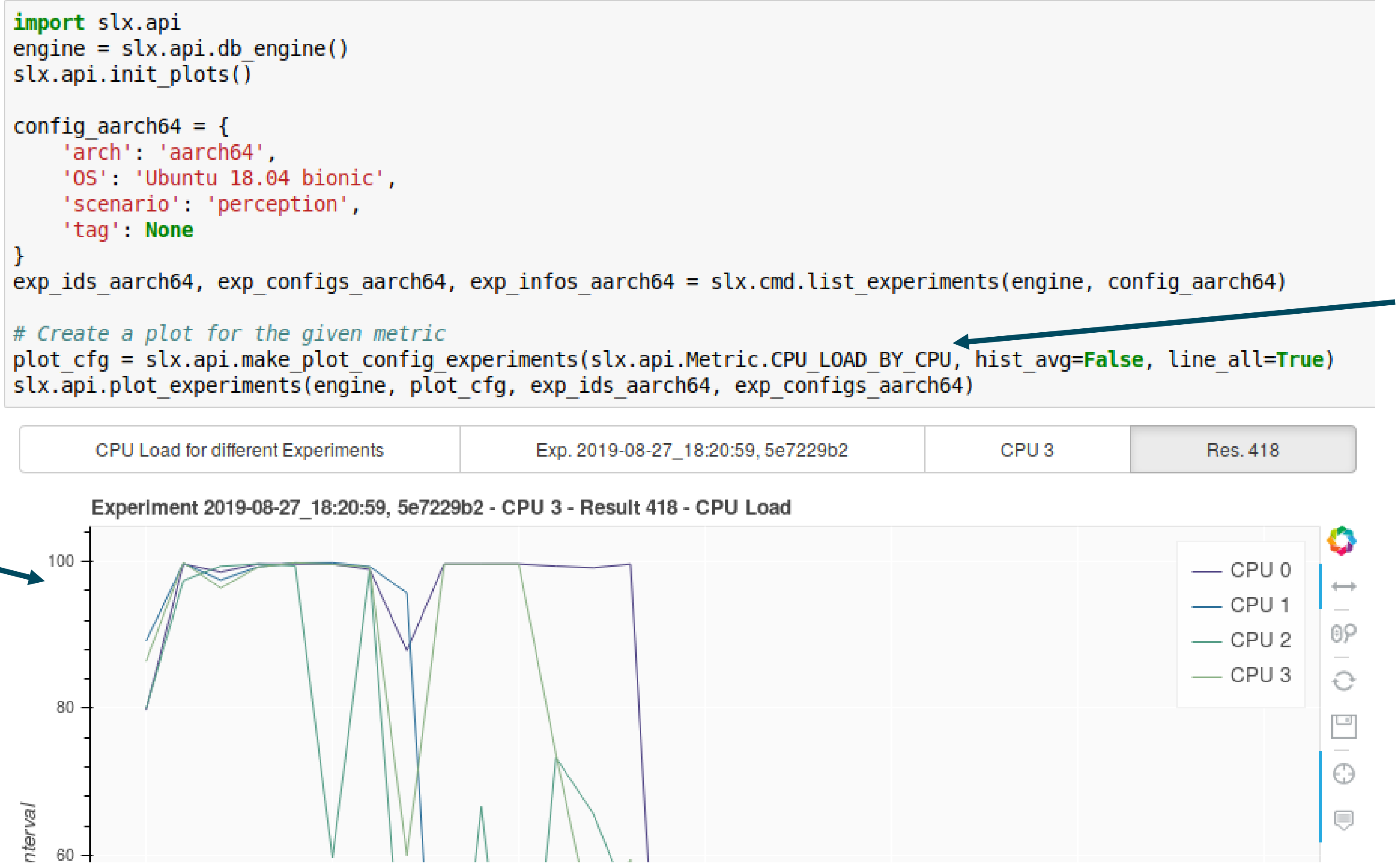
CPU
s

Time / s

Level 4

Open API / Jupyter Notebooks

Select Results



```
import slx.api
engine = slx.api.db_engine()
slx.api.init_plots()

config_aarch64 = {
    'arch': 'aarch64',
    'OS': 'Ubuntu 18.04 bionic',
    'scenario': 'perception',
    'tag': None
}
exp_ids_aarch64, exp_configs_aarch64, exp_infos_aarch64 = slx.cmd.list_experiments(engine, config_aarch64)

# Create a plot for the given metric
plot_cfg = slx.api.make_plot_config_experiments(slx.api.Metric.CPU_LOAD_BY_CPU, hist_avg=False, line_all=True)
slx.api.plot_experiments(engine, plot_cfg, exp_ids_aarch64, exp_configs_aarch64)
```

CPU Load for different Experiments
Exp. 2019-08-27_18:20:59, 5e7229b2
CPU 3
Res. 418
Experiment 2019-08-27_18:20:59, 5e7229b2 - CPU 3 - Result 418 - CPU Load
100
80
60
nterval
CPU 0
CPU 1
CPU 2
CPU 3

Select Metric

Visualize

Thanks

1st product release in January 2020

silexica