18 Bluetooth Controllers Walk Into a Bar

Observability & Runtime Configuration with CNCF Tools

Simon Schrottner
Dynatrace

Manuel Timelthaler
Tractive

Cloud Native Linz, February 2026

What is JoustMania?

A motion-controlled party game for up to 18+ PlayStation Move controllers

Keep your controller still, jostle everyone else's.
No screens, just glowing controllers and chaos.

Photos by Brent Knepper, Sara Bobo, and Die Gute Fabrik;
showing original J.S. Joust game

How do we know what’s going on?

What if we added observability tools?

What if we added feature flags too?

Agenda

1. **The Challenge** – Why observability for a party game?
2. **The Journey** – 7 key learnings from instrumentation to subsecond metrics
3. **Live Demo** – Watch metrics respond in real-time as we change the game
4. **The Path Forward** – What's missing and how you can help

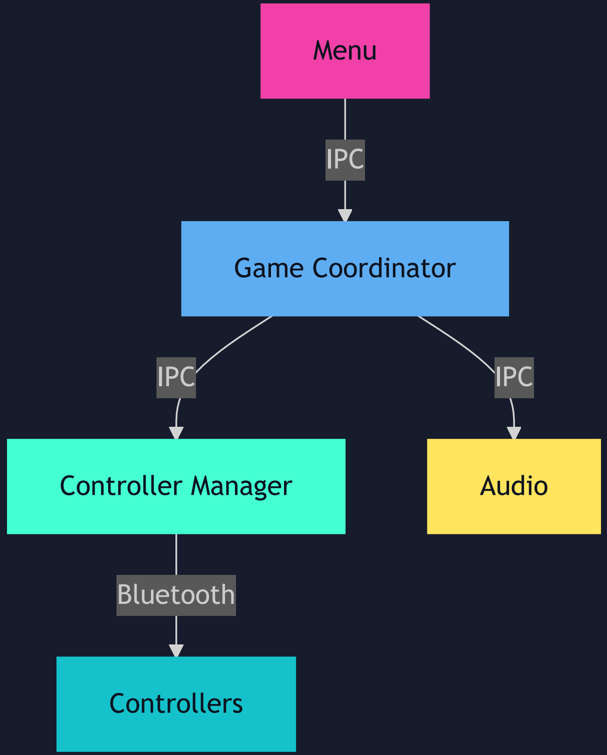
The Journey: 7 Learnings

What we discovered bringing CNCF tools to a real-time game

Learning 1: The Original Architecture

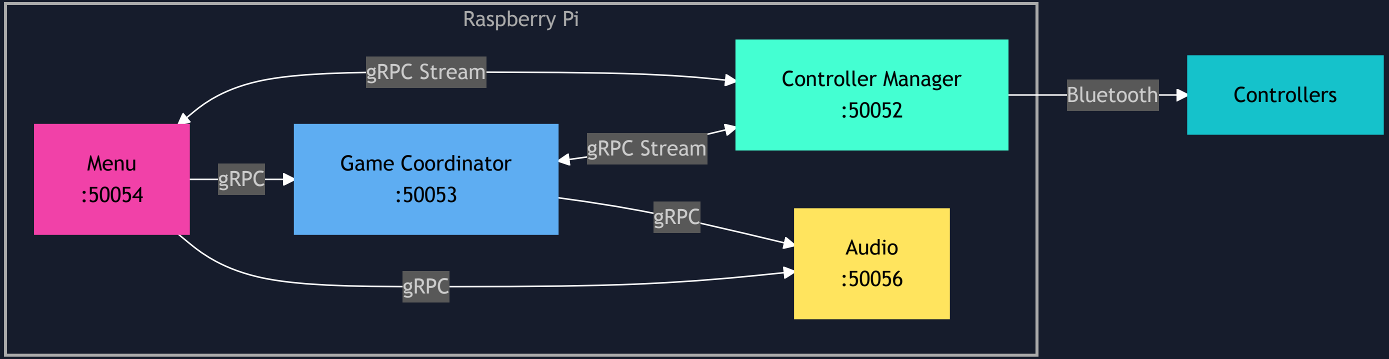
- Process-based (multiprocessing)
- 30 Hz game loop (33ms frames)
- IPC via queues/shared memory
- 4 Python processes

Problem: IPC (pipes/queues) needs manual instrumentation



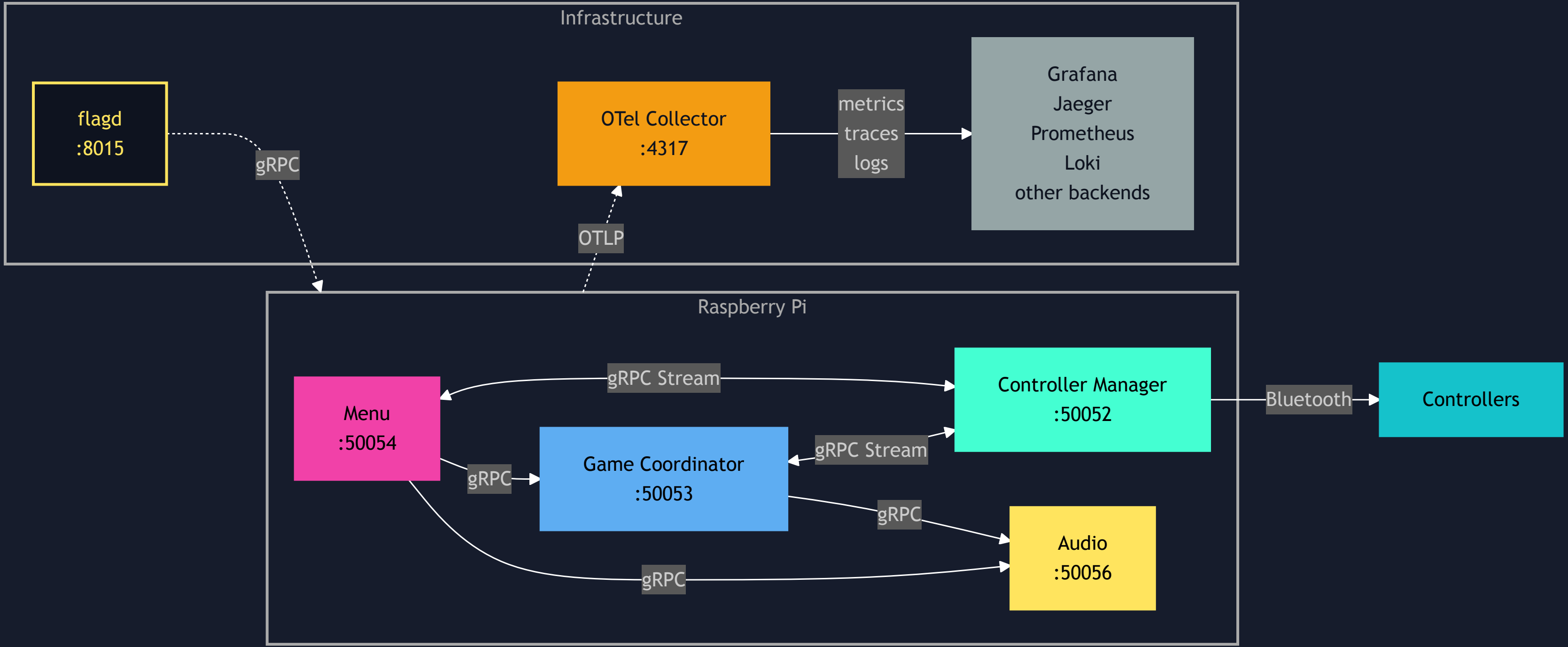
Learning 1: Microservices Unlocked the Stack

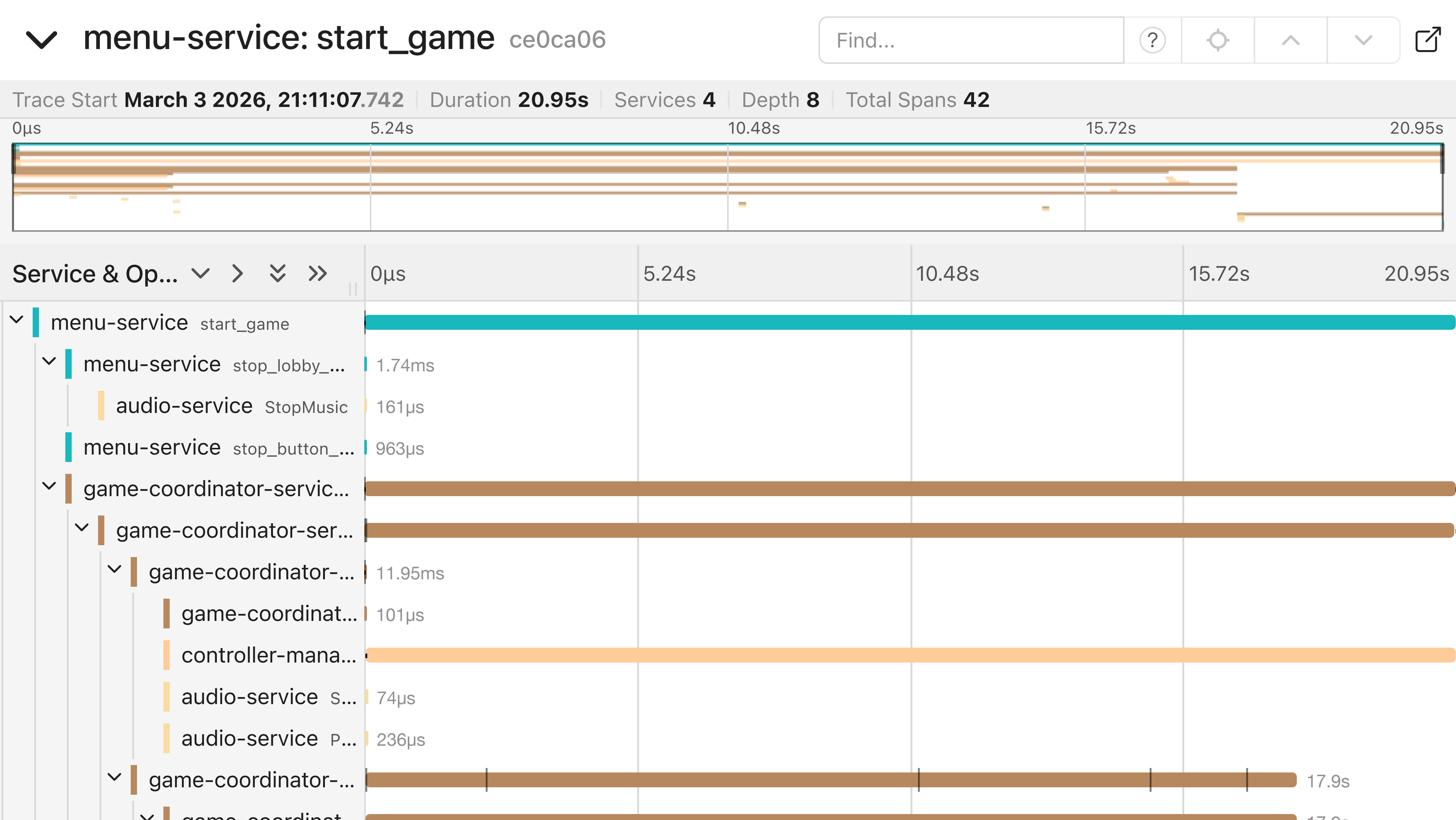
Auto-instrumentation came for free with gRPC



W3C trace context propagates automatically, enabling distributed tracing out of the box

Learning 1: Microservices - the full picture





menu-service: start_game ce0ca06

Find...

Trace Start **March 3 2026, 21:11:07.742** | Duration **20.95s** | Services **4** | Depth **8** | Total Spans **42**

0µs 5.24s 10.48s 15.72s 20.95s

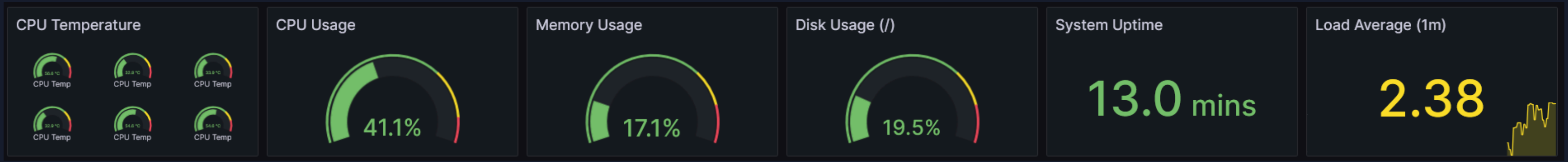
Service & Op...

Service & Operation	Duration
menu-service start_game	
menu-service stop_lobby_...	1.74ms
audio-service StopMusic	161µs
menu-service stop_button_...	963µs
game-coordinator-servic...	
game-coordinator-ser...	
game-coordinator-...	11.95ms
game-coordinat...	101µs
controller-mana...	
audio-service S...	74µs
audio-service P...	236µs
game-coordinator-...	17.9s

Learning 2: The Raspberry Pi Can Handle It

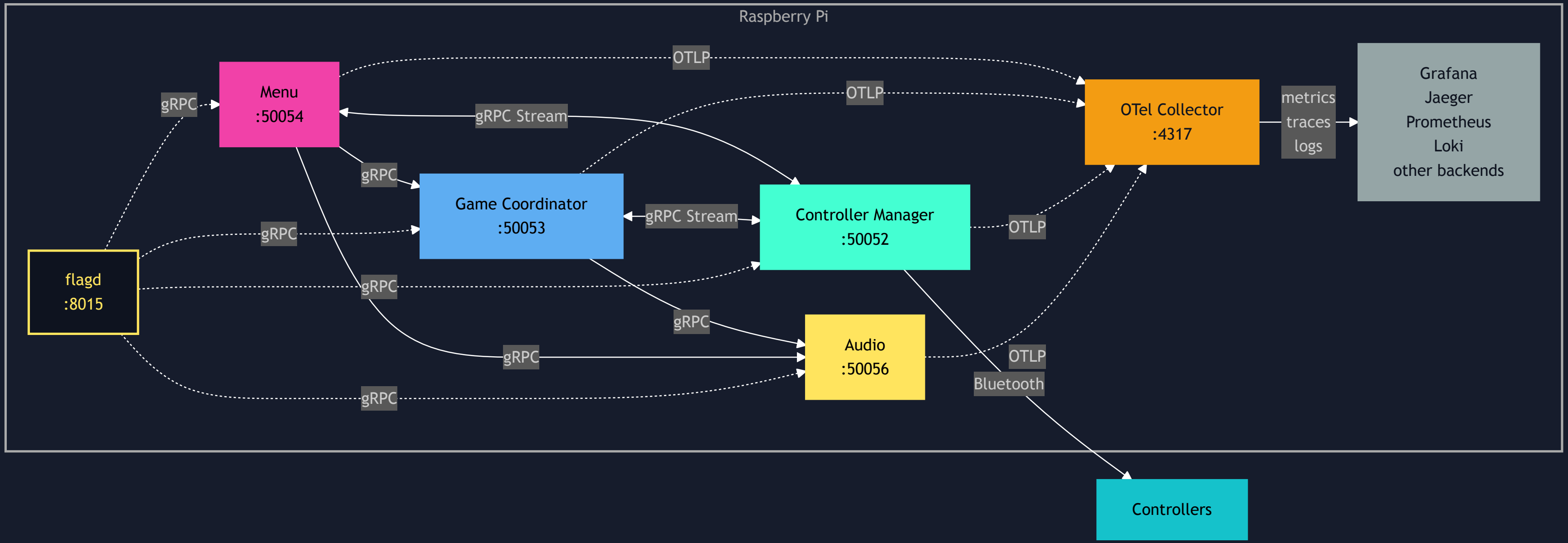
Raspberry Pi 5 Specs

- Quad-core ARM Cortex-A76 @ 2.4GHz
- 8GB LPDDR4X RAM
- ~$80 USD (before global memory shortage)



Learning 2: The Raspberry Pi Can Handle It

The Pi runs both the game AND the full observability stack.



Learning 3: Cardinality Low, Volume High

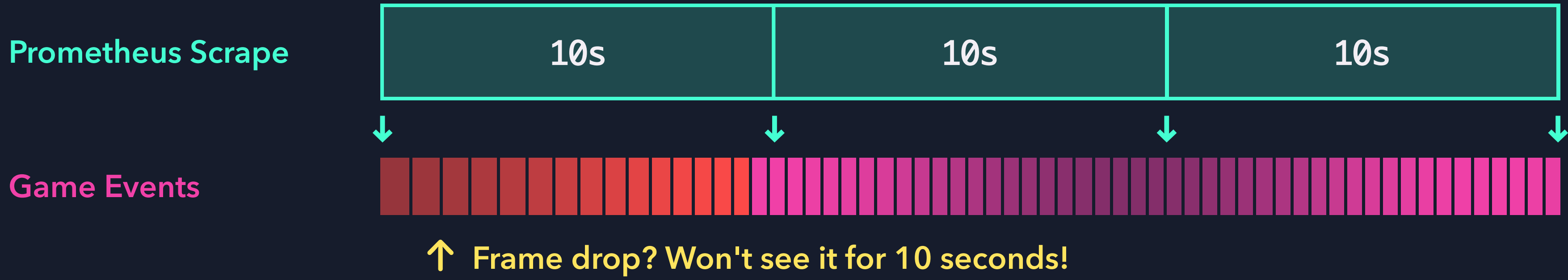
Export rate is the real challenge

- **Cardinality:** Hundreds of time series, not millions – manageable
- **Volume:** 18 controllers @ 60Hz ≈ 1,080 messages/second

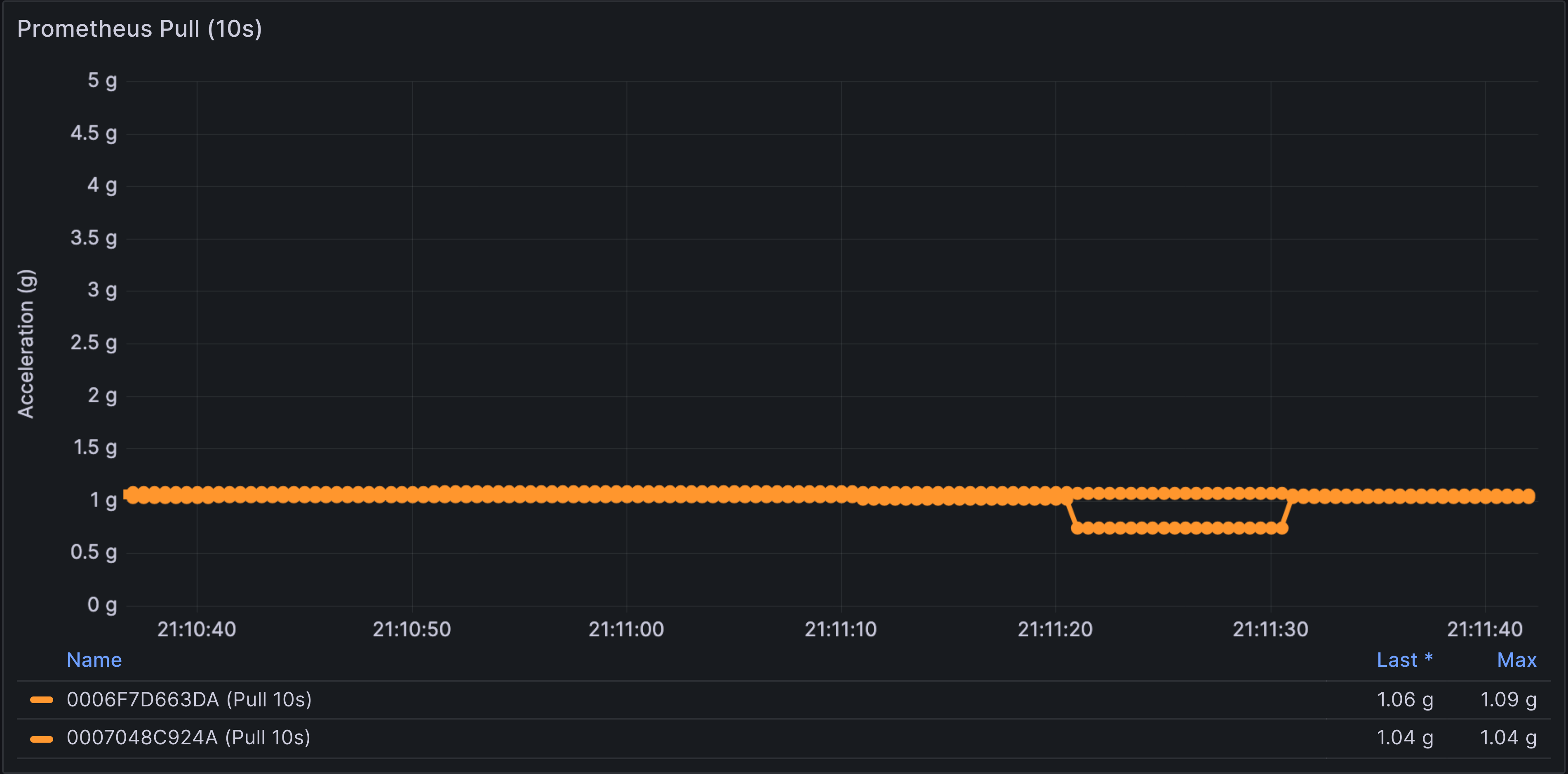
Solution

- **Two levels of batching**
  - SDK buffers and exports on a 100ms schedule (not every event)
  - Collector batches again before forwarding to backends
- **Result:** backends see smooth, regular pushes – not a firehose

Learning 4: Pull Scraping Is Too Slow



- **Pull interval:** 10 seconds (we tuned from 60s default)
- **Game loop:** 60Hz (16ms per frame)
- **Result:** 600 frames between each data point

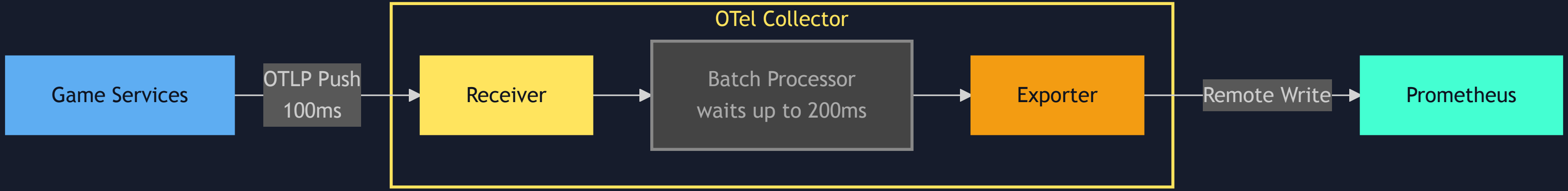


Name	Last *	Max
0006F7D663DA (Pull 10s)	1.06 g	1.09 g
0007048C924A (Pull 10s)	1.04 g	1.04 g

Learning 5: Push Metrics with Prometheus

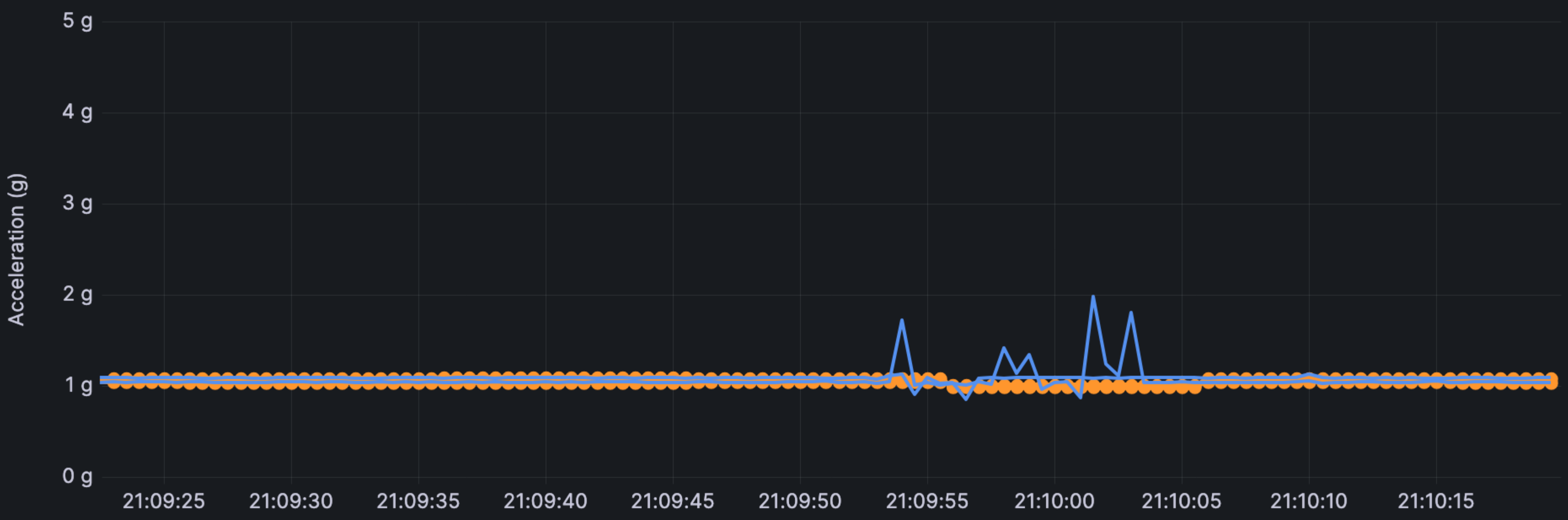
OTLP Push via PeriodicExportingMetricReader

```
# OpenTelemetry SDK Configuration
metric_reader = PeriodicExportingMetricReader(
    exporter=OTLPMetricExporter(endpoint="http://otel-collector:4318"),
    export_interval_millis=flagd.get_int("metrics_export_interval_ms")
    # controller-manager: 100ms (realtime) | other services: 1000ms
)
```



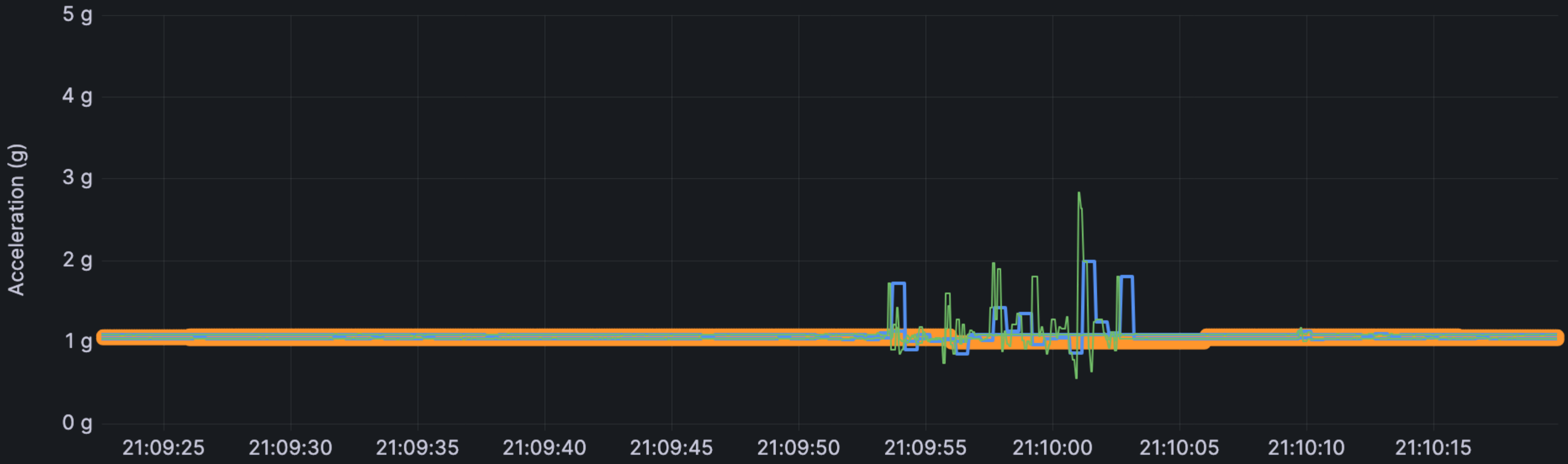
500ms end-to-end reliably – one config change, 30× faster than pull.
For true sub-100ms, the TSDB itself becomes the next bottleneck.

Pull + Push (Prometheus)



Name	Last *	Max
🟧 0006F7D663DA (Pull 10s)	1.08 g	1.09 g
🟧 0007048C924A (Pull 10s)	1.04 g	1.05 g
🟦 0006F7D663DA (OTEL→Prom 500ms)	1.09 g	1.13 g
🟦 0007048C924A (OTEL→Prom 500ms)	1.04 g	1.98 g

All Three Pipelines



Name	Last *	Max
🟧 0006F7D663DA (Pull 10s)	1.08 g	1.09 g
🟧 0007048C924A (Pull 10s)	1.04 g	1.05 g
🟦 0006F7D663DA (OTEL→Prom 500ms)	1.09 g	1.13 g
🟦 0007048C924A (OTEL→Prom 500ms)	1.04 g	1.98 g
🟩 0006F7D663DA (OTEL→VM 100ms)	1.09 g	1.44 g

Learning 6: Labels Are Not Free

Add a `game_id` label

🏷️

1,216 series → 18,000 series

Prometheus concurrent p50

87ms → 389ms

VictoriaMetrics: 47ms → 46ms

Every label dimension multiplies your series count – cardinality is where Prometheus hurts

Benchmark: 36 controllers at 100ms push intervals – 2× our setup, so real-world numbers are better

Learning 7: These Tools Actually Work

Live Demo: Real-time observability in action

Controlled Chaos with Feature Flags

Status: **none**

INJECT FAULT

Poll Drop | Acceleration Spike

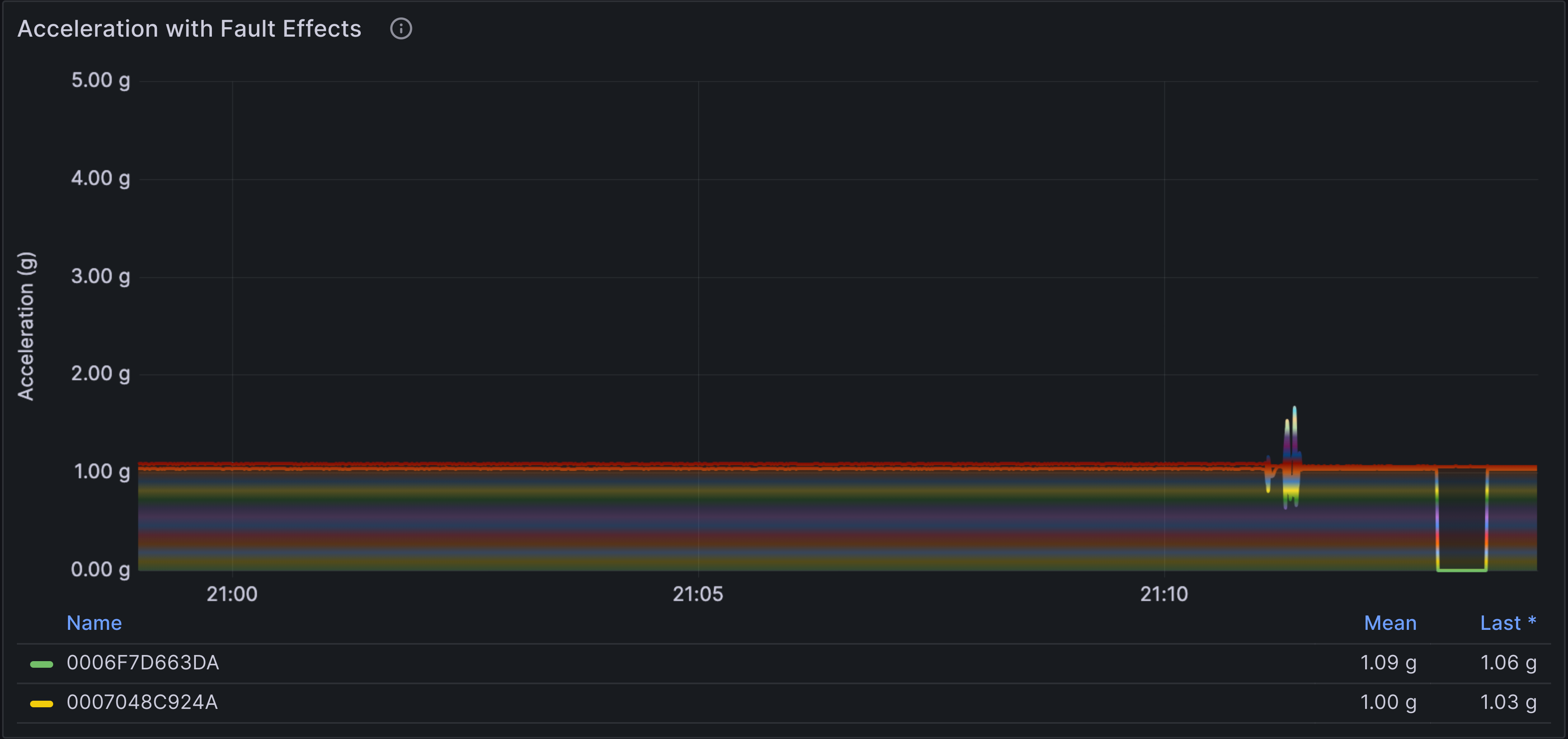
LED Flicker | Disconnect

FRACTION: **50%**

Reset All Faults

RAW FLAGD STATUS

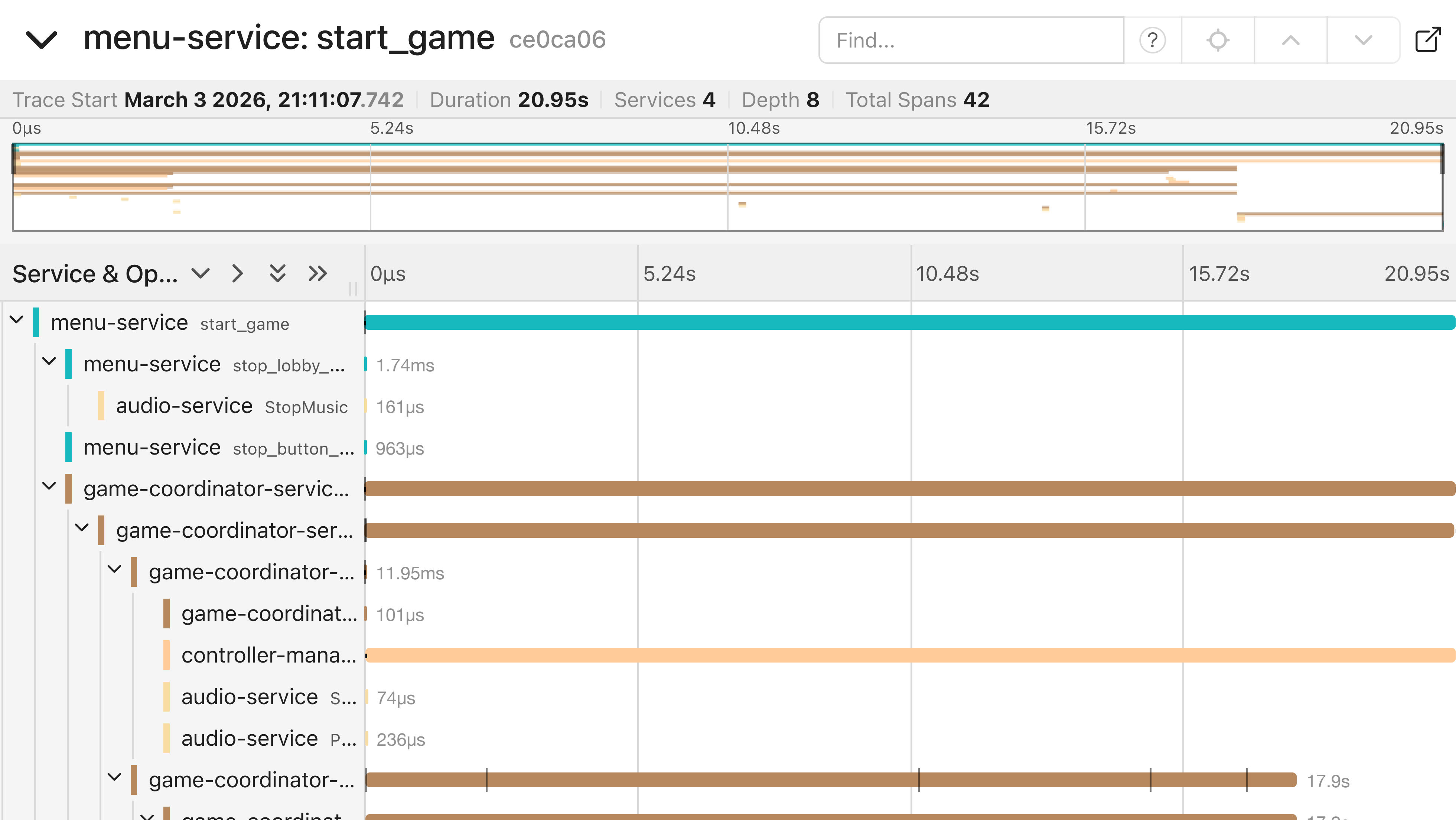
```
{
  "state": "ENABLED",
  "variants": {
    "accel_spike": "accel_spike",
    "disconnect": "disconnect",
    "led_failure": "led_failure",
    "none": "none",
    "poll_drop": "poll_drop"
  },
  "defaultVariant": "none"
}
```



Acceleration with Fault Effects

	Name	Mean	Last *
🟩	0006F7D663DA	1.09 g	1.06 g
🟨	0007048C924A	1.00 g	1.03 g

Fault Injection Rate



menu-service: start_game ce0ca06

Find...

Trace Start **March 3 2026, 21:11:07**.742 | Duration **20.95s** | Services **4** | Depth **8** | Total Spans **42**

0µs 5.24s 10.48s 15.72s 20.95s

Service & Op...

Service & Operation	Duration
menu-service start_game	
menu-service stop_lobby_...	1.74ms
audio-service StopMusic	161µs
menu-service stop_button_...	963µs
game-coordinator-servic...	
game-coordinator-ser...	
game-coordinator-...	11.95ms
game-coordinat...	101µs
controller-mana...	
audio-service S...	74µs
audio-service P...	236µs
game-coordinator-...	17.9s

4 Key Takeaways

1 **These tools work for real-time.**

Games, IoT, embedded systems–anything real-time. Not just web apps.

2 **But they're optimized for web apps.**

Default configs assume 15-second scrapes, not 60Hz game loops.

3 **With tuning, you can get subsecond observability on an $80 computer.**

Intervals, push vs pull, storage backends.

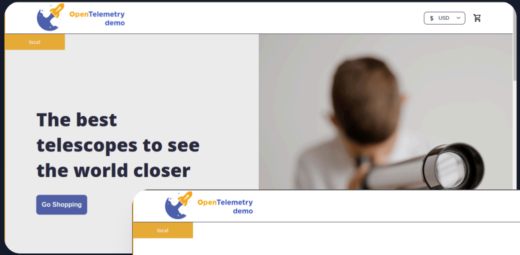
4 **The tools exist. The patterns exist.**

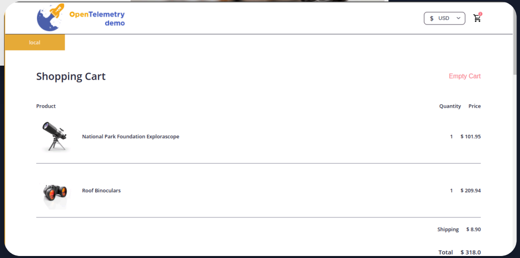
What's missing is the documented path.

Where's the Real-Time Systems Demo?

Great for microservices—but where's the robotics? The game engines? The industrial IoT?

Try these tools on real-time systems.
Document what works. Share tuning tricks. Contribute examples.



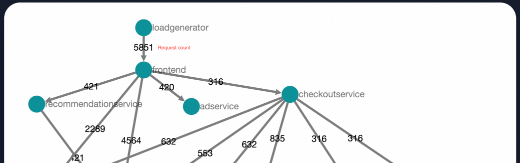


Shopping Cart

Product	Quantity	Price
National Park Foundation Explorascope	1	$ 101.95
Roof Binoculars	1	$ 209.94

Shipping $ 8.90

Total $ 318.0



JoustMania is Open Source

Fork it. Break it. Make it better.

Shoutouts

J. S. Joust · JoustMania · OTel / CNCF community

github.com/WatchMeJoustMyFlags/JoustMania

Real-Time Observability with CNCF Tools

Thank You!

Simon Schrottner

Dynatrace

linkedin.com/in/aepfli

@aepfli · github.com/aepfli

Manuel Timelthaler

Tractive

linkedin.com/in/manuel-timelthaler

github.com/Lorti

or find us on CNCF's Slack Channel

Questions? 🎮

We need your feedback 🙏

Scan to leave feedback – helps us improve for KubeCon