HOWTO

CONTINUOUS PERFORMANCE ENGINEERING

Henrik Ingo

Fosdem 2026

NYRKIÖ

HOWTO

CONTINUOUS BENCHMARKING

Henrik Ingo

Fosdem 2026

NYRKIÖ

POLL

Continuous Performance
Engineering
maturity levels

1. We don’t really do benchmarking
2. Benchmarks yes, continuous no
3. Some benchmarks run in CI

   …but we ignore the results
4. Benchmarks in CI,

   Daily or more often,

   Actionable results,

   Regressions fixed within weeks

CONTINUOUS INTEGRATION

CONTINUOUS TESTING

CONTINUOUS DEPLOYMENT

DEVOPS

LEFT SHIFTING

ITERATIONS

NYRKIÖ

CONTINUOUS PERFORMANCE ENGINEERING?

Somewhere in that evolution, they forgot to bring the performance engineer

Velociraptor, the fastest dinosaur (wikipedia)

NYRKIÖ

INTUITION
PERFORMANCE ENGINEERS
MATH

WHY WAS PERFORMANCE ENGINEERING LEFT IN 1999?

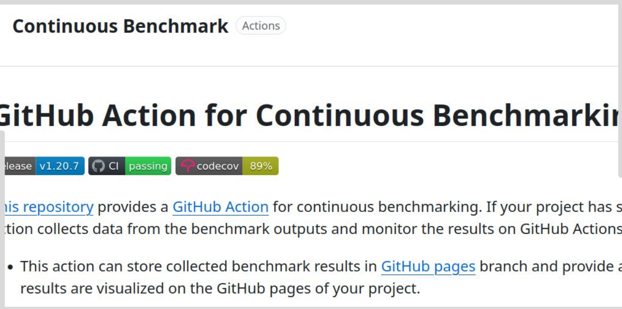
Deploying is necessary - benchmarking is optional

Performance Engineers busy fixing prod / customer problems
….and maybe enjoying it?

Math is hard. Unfortunately it is also mandatory.

Relevant tuning is unintuitive, not well known

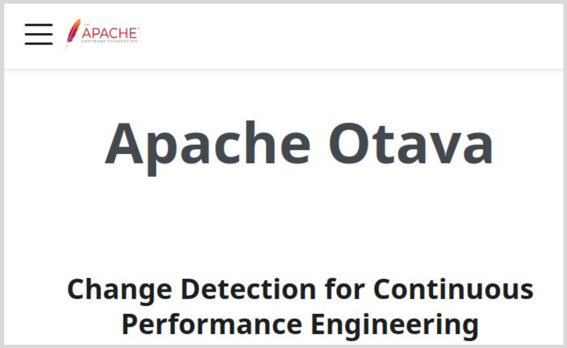
WHO ELSE TALKS ABOUT THIS?



GitHub Action for Continuous Benchmarkir

is repository provides a GitHub Action for continuous benchmarking. If your project has some b
tion collects data from the benchmark outputs and monitor the results on GitHub Actions workf

- This action can store collected benchmark results in GitHub pages branch and provide a chart results are visualized on the GitHub pages of your project.



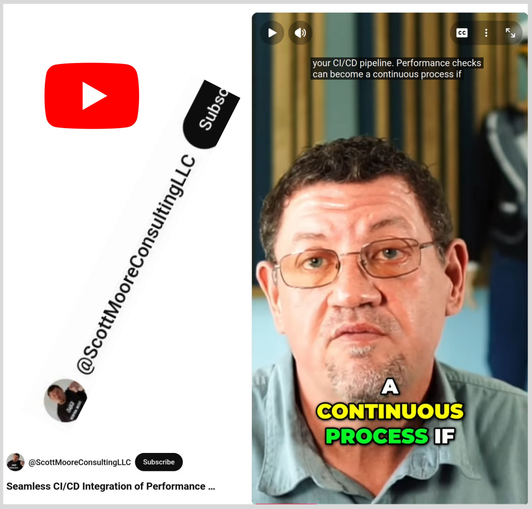
Apache Otava

Change Detection for Continuous Performance Engineering

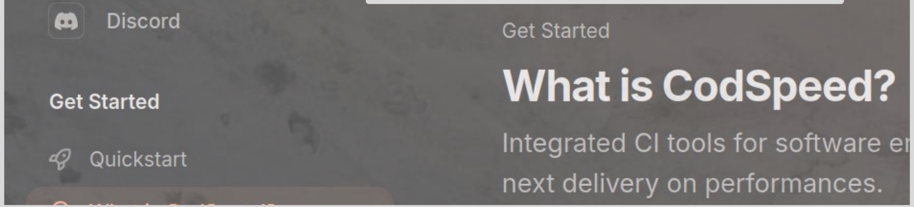
Netherlands

Business Description: Perfana delivers softwar deliver high-performing s

Based in: Amsterdam



Seamless CI/CD Integration of Performance ...

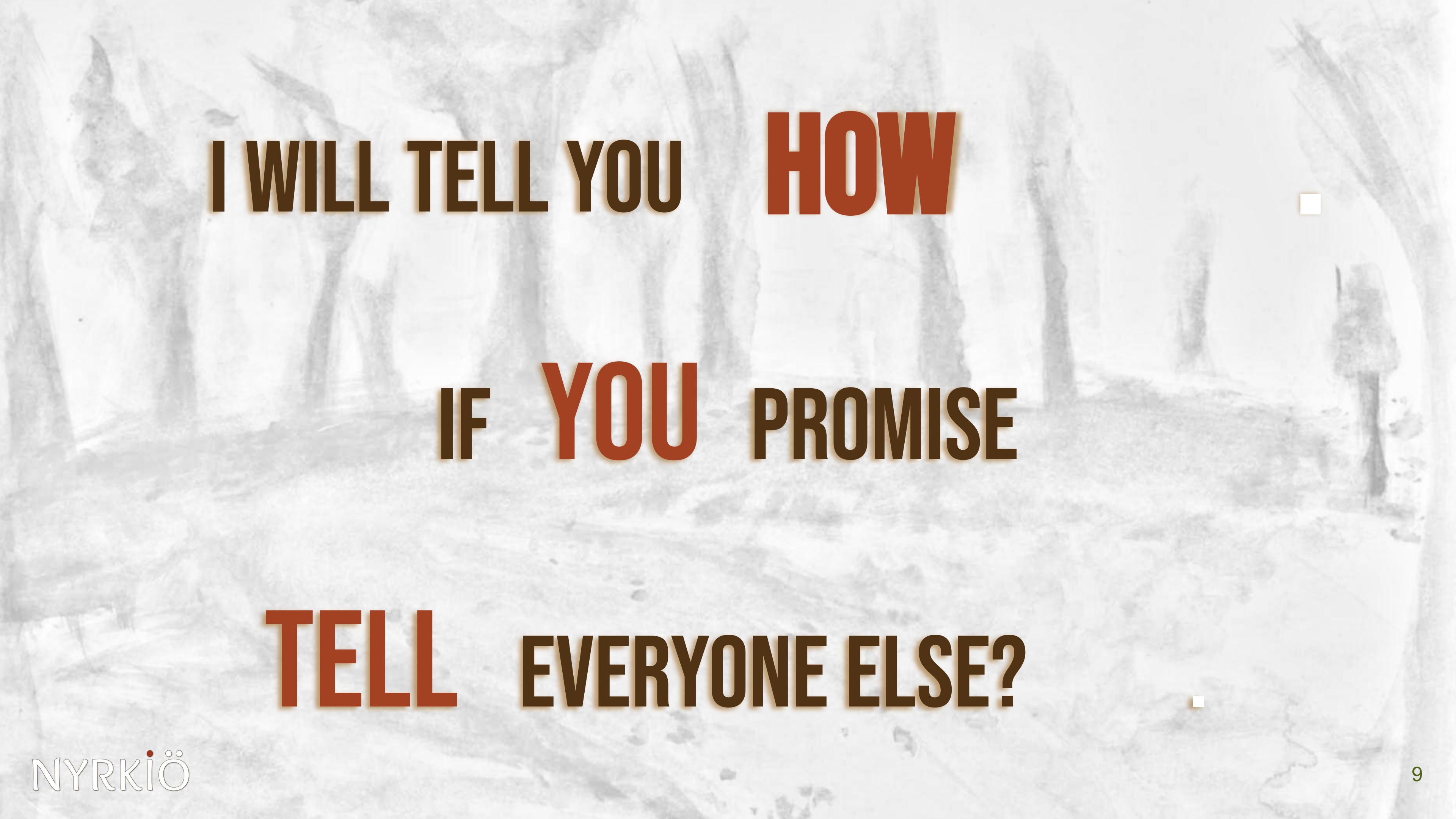


Get Started

What is CodSpeed?

Integrated CI tools for software er next delivery on performances.

ACM/SPEC International Conference on Performance Engineering



I WILL TELL YOU **HOW**

IF **YOU** PROMISE

TELL EVERYONE ELSE?

AGENDA

1. Benchmarking
   - Make it Continuous
2. Change Point Detection
   - Math & Science
3. Minimizing noise in the Benchmarks
   - Assume nothing.
   - Measure everything.

BENCHMARKING PROCESS AND TOOLS

ON BENCHMARK TOOLS & DESIGN

Previous talk by Kemal Akkoyun & Augusto de Oliveira.

But in general…

- at this point each language has its own standard framework:
    - Java & JMH
    - Python & pytest-benchmark
    - etc…

Frameworks to run fully, end to end automated distributed benchmarks exist. Personally I believe we will see new innovation coming here. (k8s)

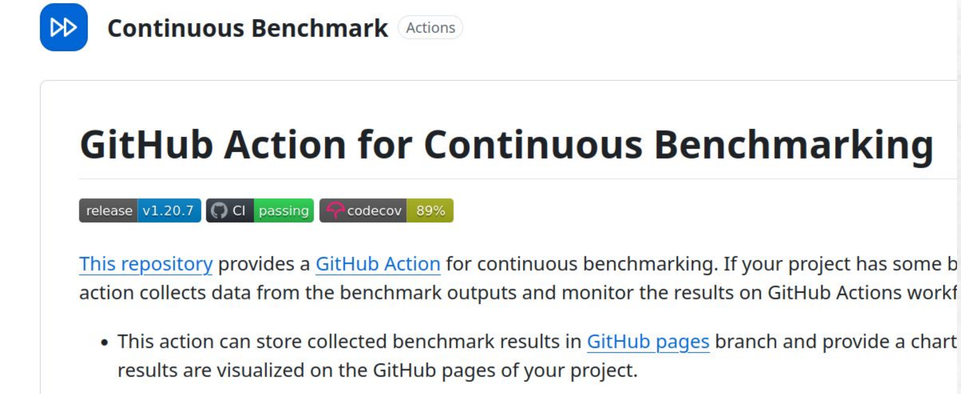
GITHUB-ACTION-BENCHMARK

Use in your workflows immediately after the benchmark step

Supports output from all major frameworks

Stores result history in your github repo.

Threshold based alerts.
Default threshold = 100% (2x)



GitHub Action for Continuous Benchmarking

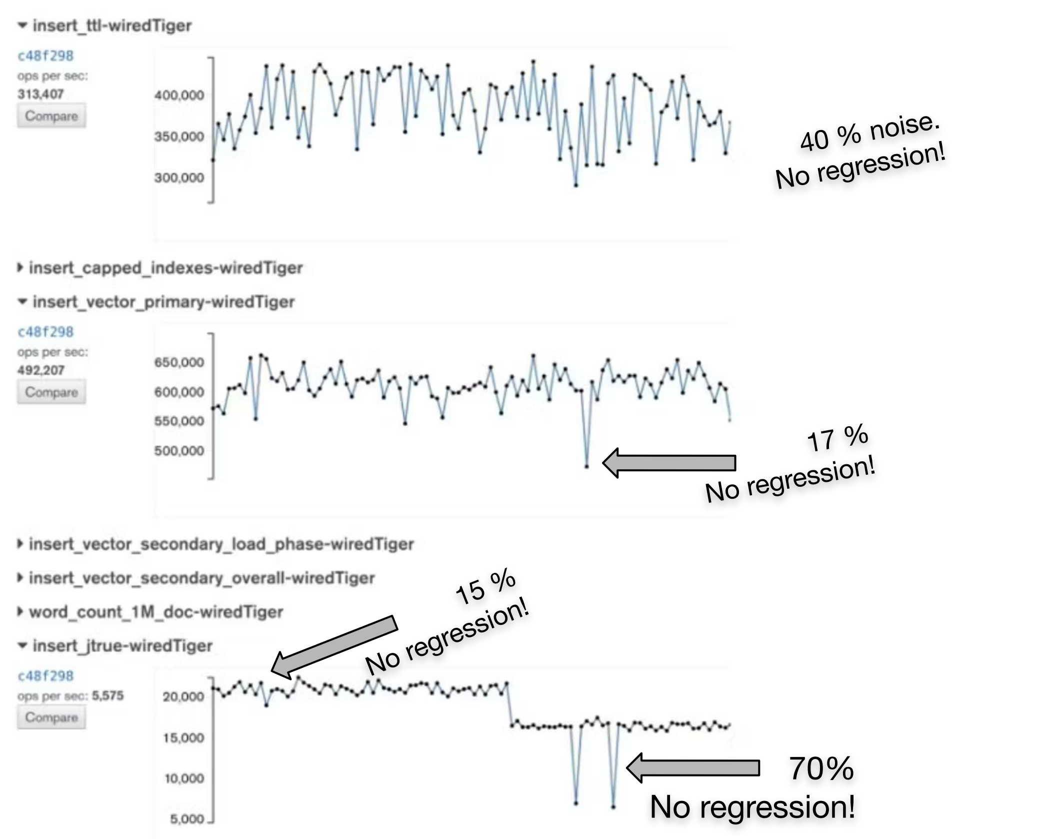
This repository provides a GitHub Action for continuous benchmarking. If your project has some b
action collects data from the benchmark outputs and monitor the results on GitHub Actions workf

- This action can store collected benchmark results in GitHub pages branch and provide a chart
results are visualized on the GitHub pages of your project.

NYRKIÖ

CHANGE POINT DETECTION

A DAY IN THE LIFE OF A MONGODB PERF ENGINEER, 2015.



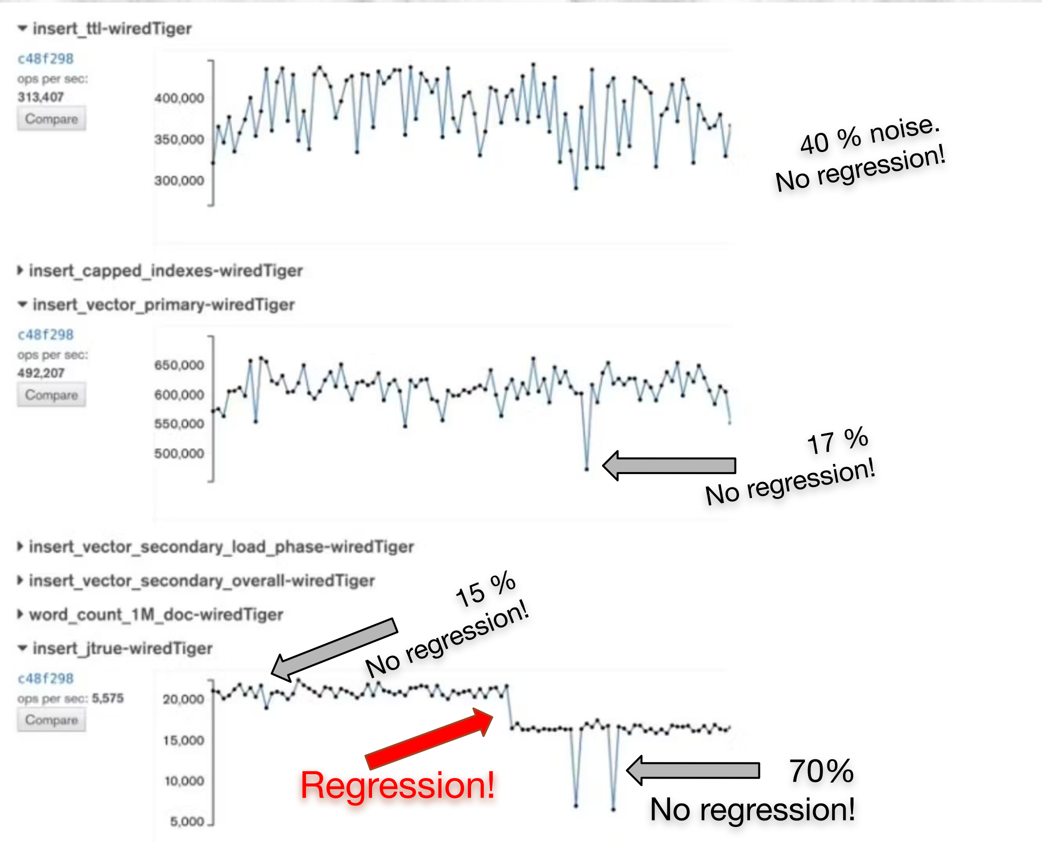
40 % noise.
No regression!

17 %
No regression!

15 %
No regression!

70%
No regression!

A DAY IN THE LIFE OF A MONGODB PERF ENGINEER, 2015.



▾ insert_ttl-wiredTiger

c48f298
ops per sec:
313,407

40 % noise.
No regression!

▸ insert_capped_indexes-wiredTiger

▾ insert_vector_primary-wiredTiger

c48f298
ops per sec:
492,207

17 %
No regression!

▸ insert_vector_secondary_load_phase-wiredTiger

▸ insert_vector_secondary_overall-wiredTiger

▸ word_count_1M_doc-wiredTiger

▾ insert_jtrue-wiredTiger

c48f298
ops per sec: 5,575

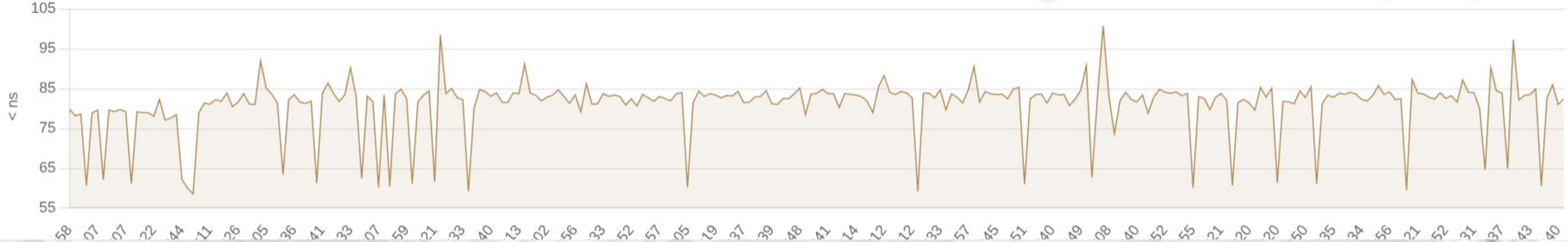
15 %
No regression!

Regression!

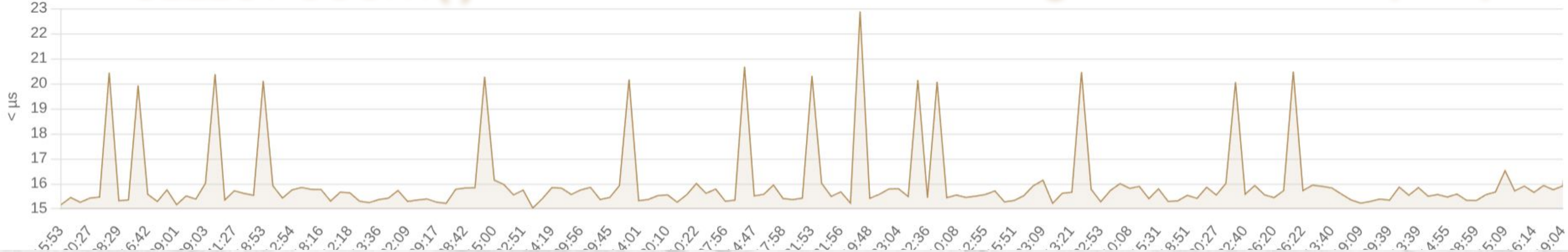
70%
No regression!

EVERYONE ON GITHUB, 2025:

SELECT 1 time ⇩ Range: 60-100 ns (40%)



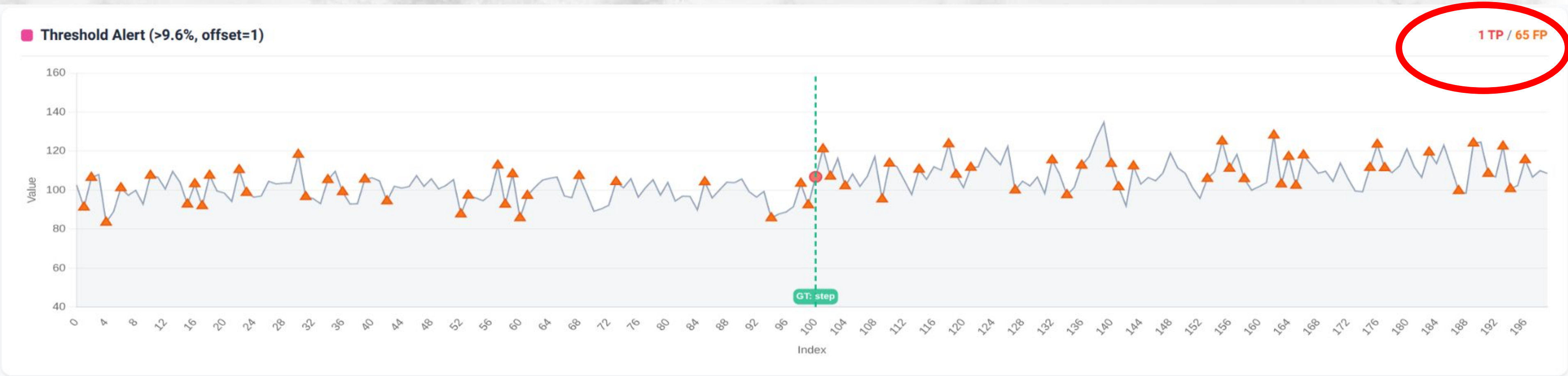
SELECT COUNT(*) time ⇩ Range: 15-23 ns (50%)



Turso database, Github runners, on nyrkio.com/public

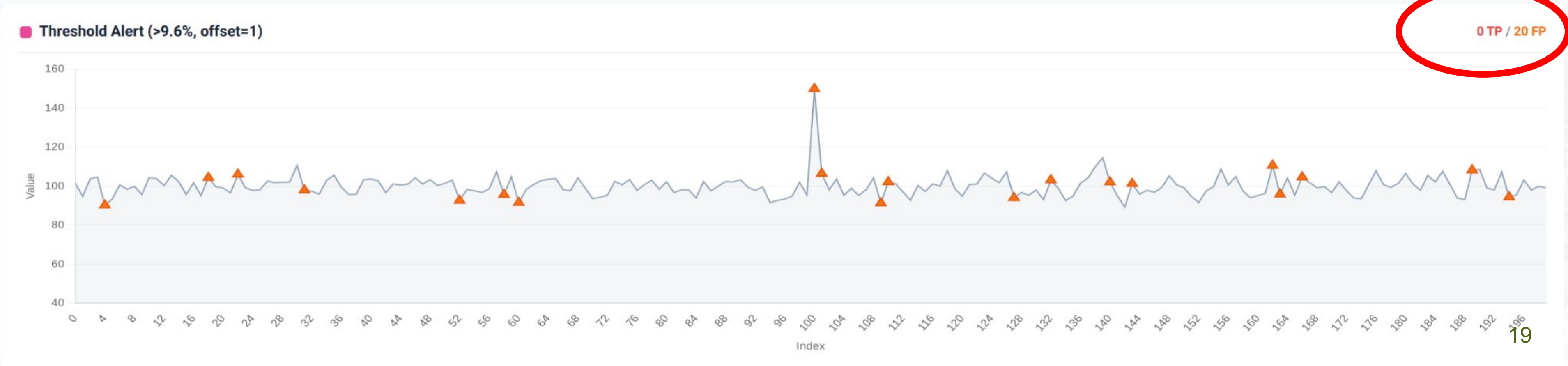
AUTOMATION IS ONE OF THE CORE PRINCIPLES…

Threshold based alerting…



Threshold Alert (>9.6%, offset=1)

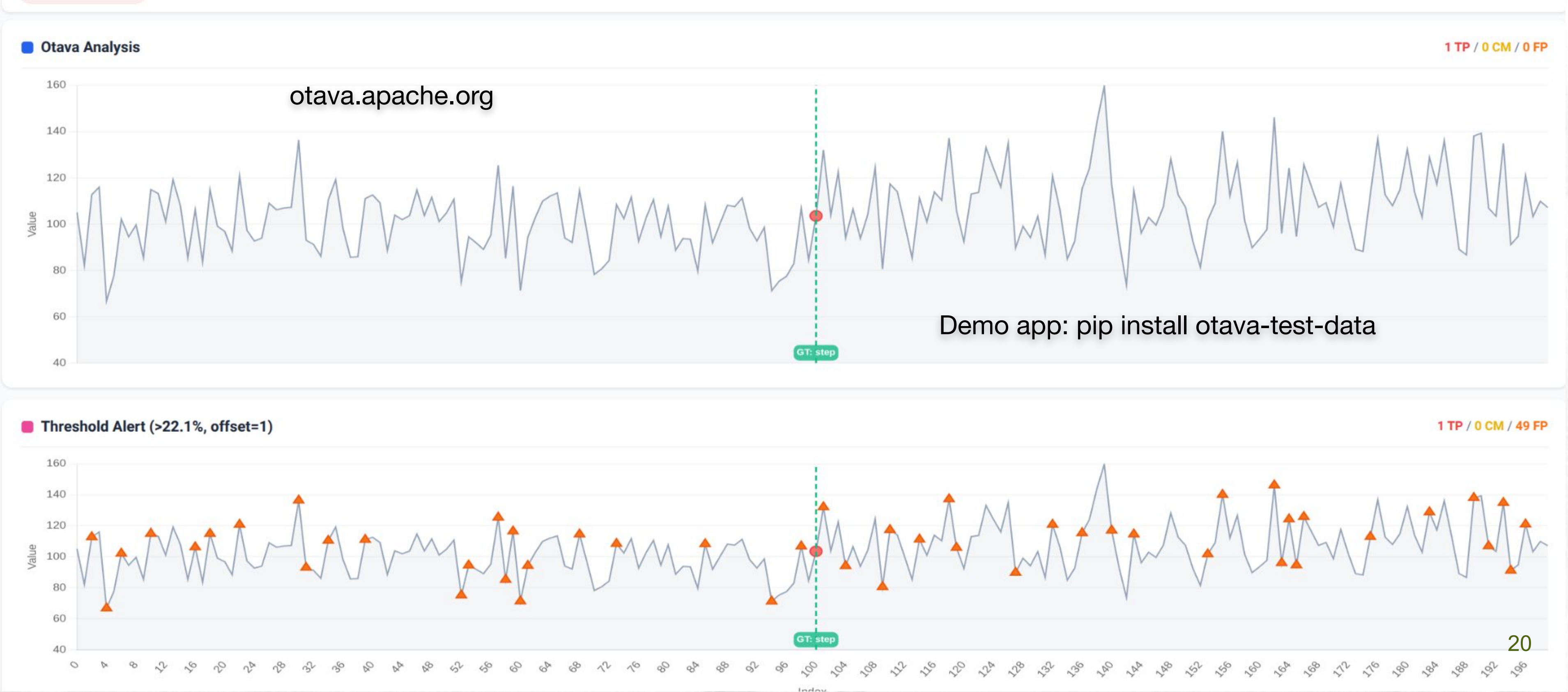
1 TP / 65 FP



Threshold Alert (>9.6%, offset=1)

0 TP / 20 FP

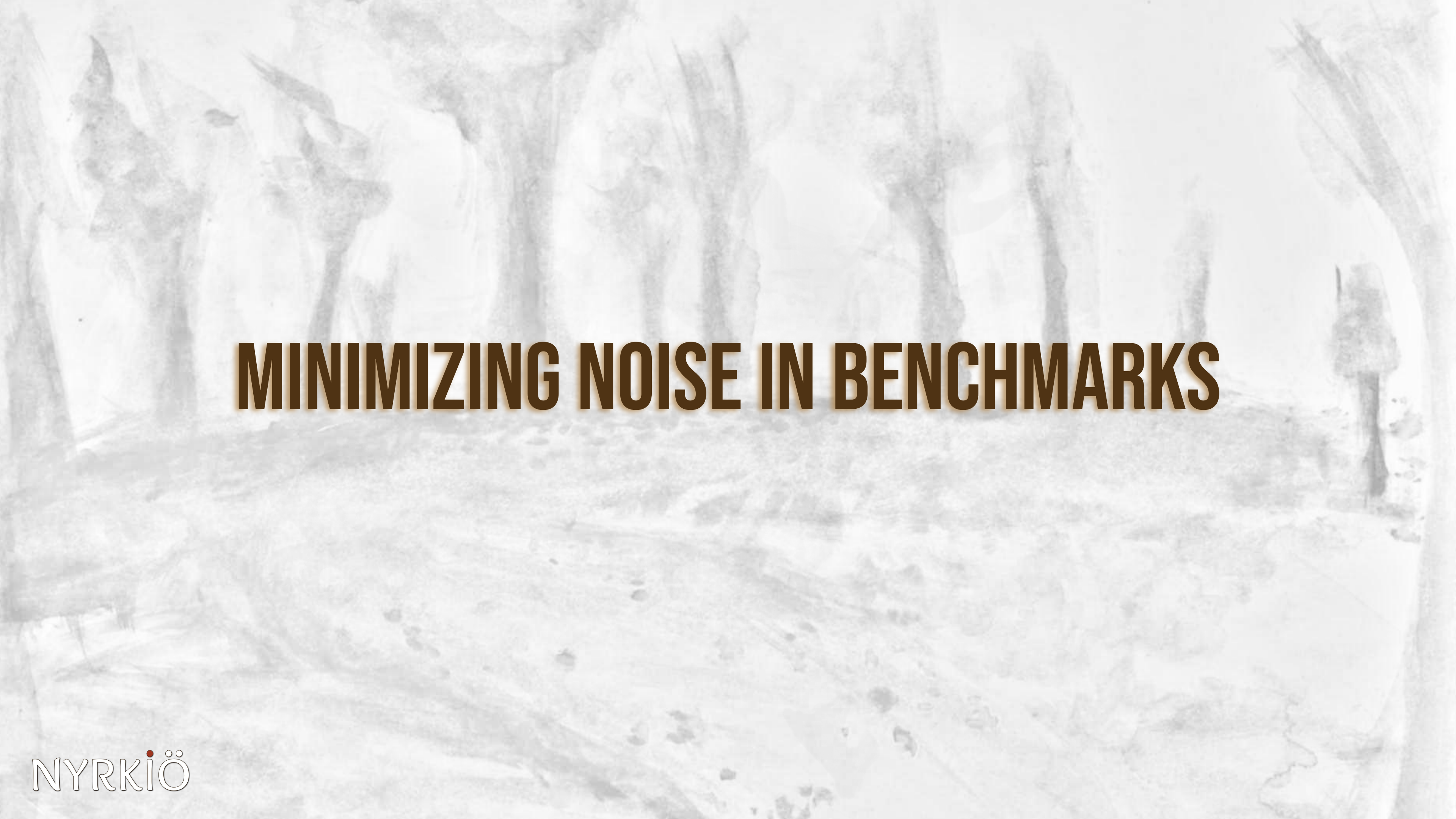
E-DIVISIVE MEANS (MATTESON & JAMES, 2014)



otava.apache.org

Demo app: pip install otava-test-data

otava.apache.org



MINIMIZING NOISE IN BENCHMARKS

Reasons why benchmark results are so noisy?

CLOUD

?

CLOUD

NOISY NEIGHBOR

SOVEREIGN
CLOUD

BAD PROGRAMMERS

SUPERSTITION
PERFORMANCE ENGINEERS
SCIENCE

FUN EXERCISE: RETROACTIVELY LIST ASSUMPTIONS BUILT INTO YOUR CURRENT ARCHITECTURE

Dedicated instance = more stable performance

Placement groups minimize network latency & variance

Different availability zones have different hardware

For write heavy tests, noise comes from disk

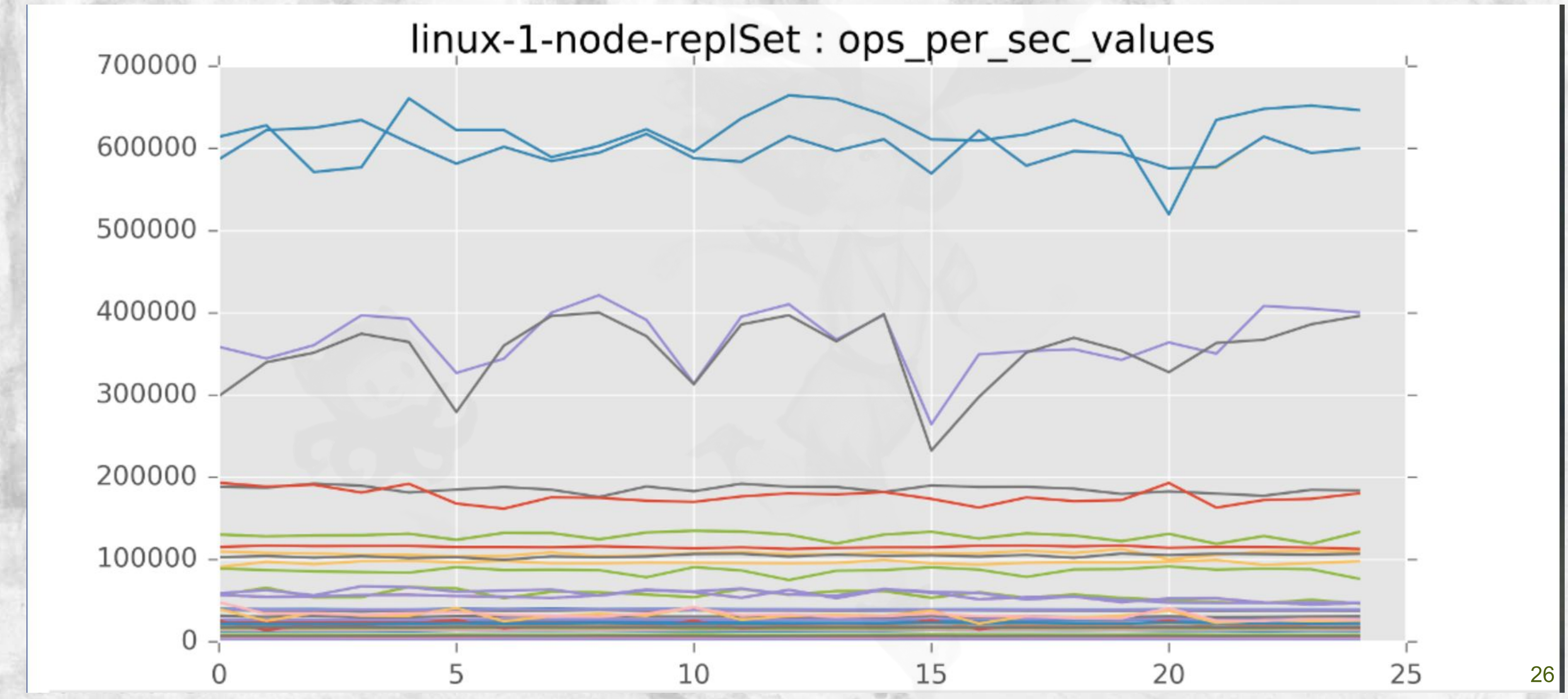
Ephemeral (SSD) disks have least variance

There are good and bad EC2 instances

Just use i2 instances (better SSD)

You can't use cloud for performance testing

1 MongoDB Binary, 5 Servers, Repeat Tests 5x (2017)



FUN EXERCISE: RETROACTIVELY LIST ASSUMPTIONS BUILT INTO YOUR CURRENT ARCHITECTURE

Dedicated instance = more stable performance

Placement groups minimize network latency & variance

Different availability zones have different hardware

For write heavy tests, noise comes from disk

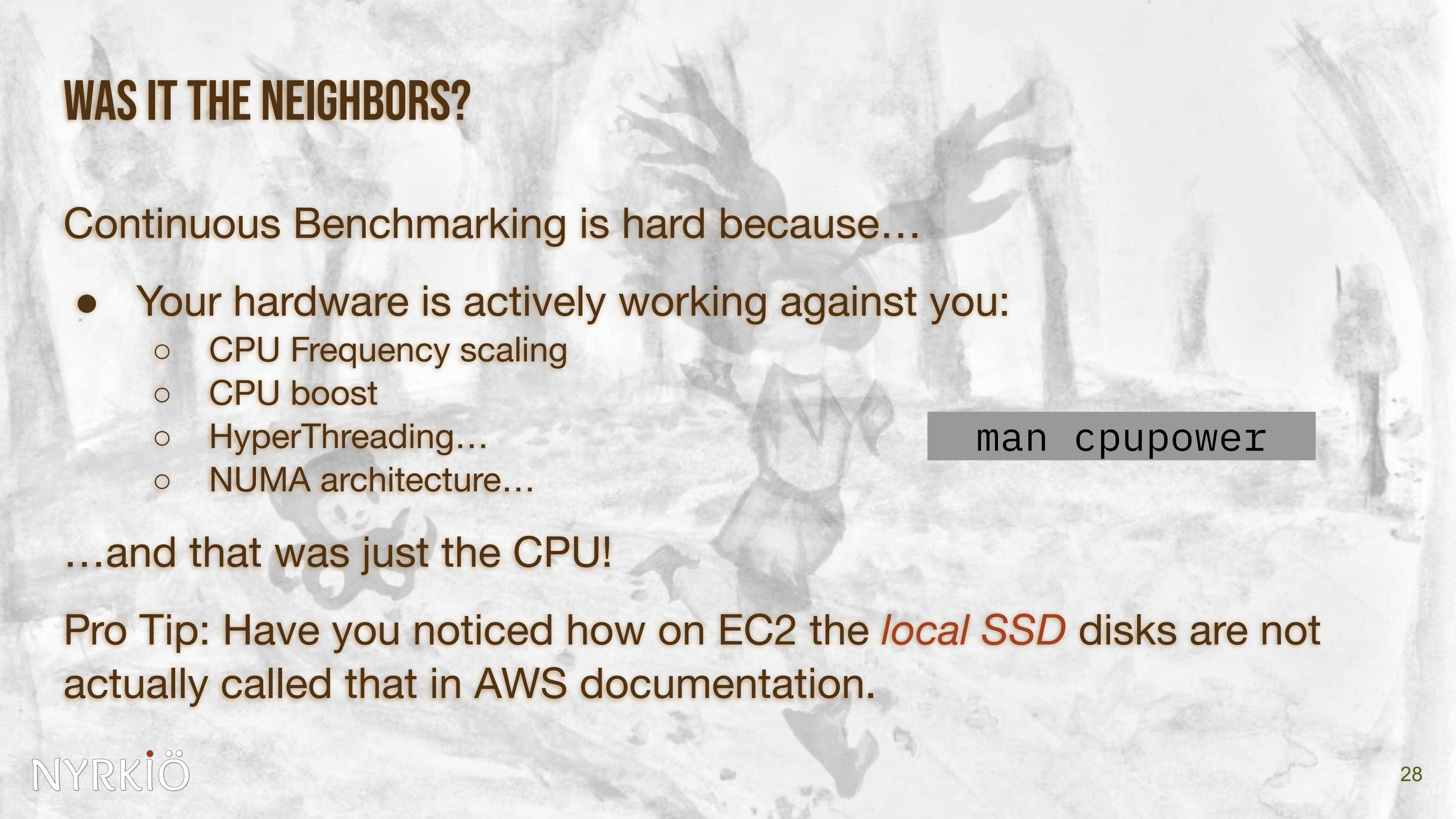
Ephemeral (SSD) disks have least variance

There are good and bad EC2 instances **False**

Just use i2 instances (better SSD)

You can't use cloud for performance testing

NYRKIÖ



WAS IT THE NEIGHBORS?

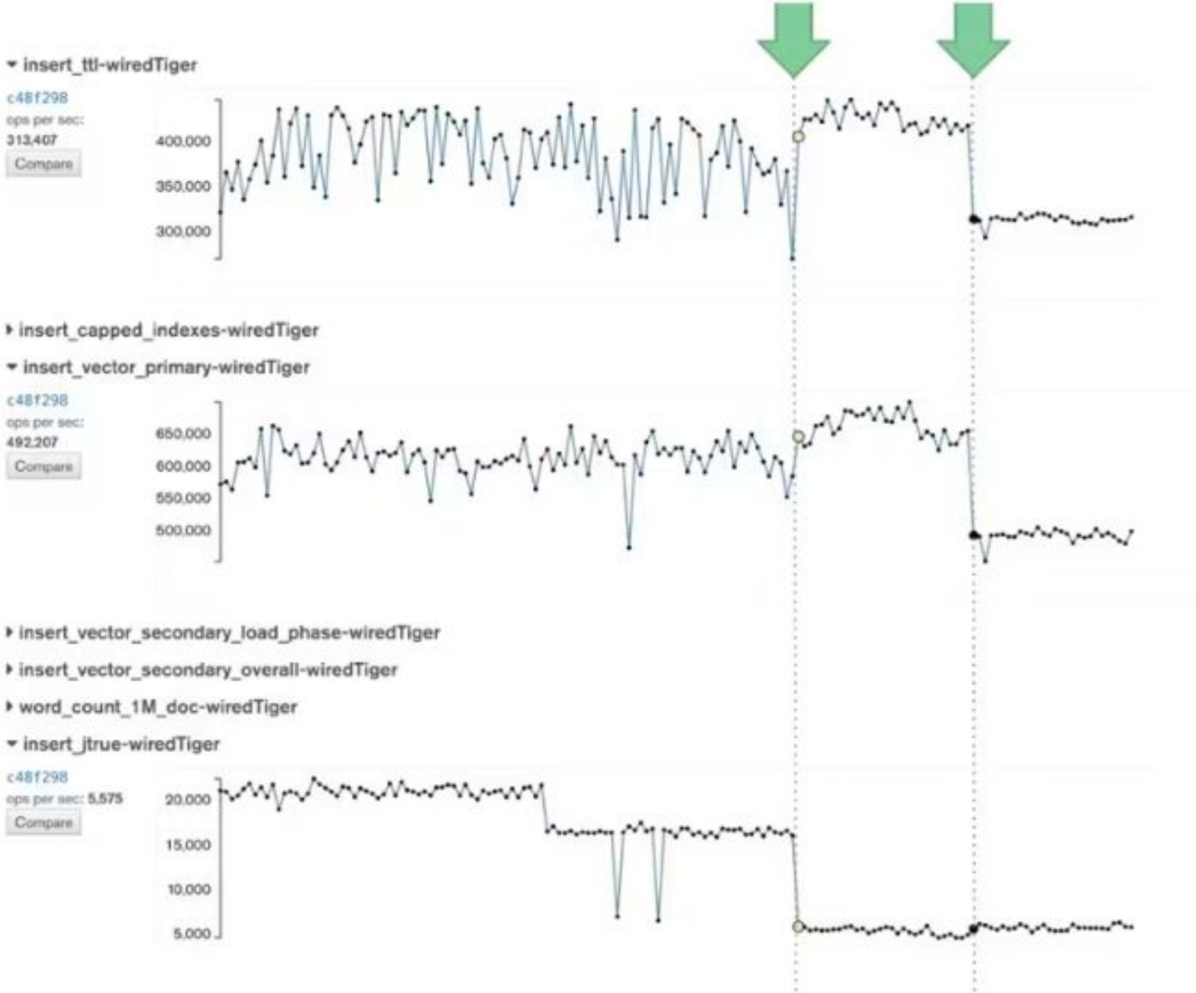
Continuous Benchmarking is hard because…

- Your hardware is actively working against you:
  - CPU Frequency scaling
  - CPU boost
  - HyperThreading…
  - NUMA architecture…

```
man cpupower
```

…and that was just the CPU!

Pro Tip: Have you noticed how on EC2 the *local SSD* disks are not actually called that in AWS documentation.



▾ insert_ttl-wiredTiger

c48f298

ops per sec: 313,407

Compare

▸ insert_capped_indexes-wiredTiger

▾ insert_vector_primary-wiredTiger

c48f298

ops per sec: 492,207

Compare

▸ insert_vector_secondary_load_phase-wiredTiger

▸ insert_vector_secondary_overall-wiredTiger

▸ word_count_1M_doc-wiredTiger

▾ insert_jtrue-wiredTiger

c48f298

ops per sec: 5,575

Compare

○ SSD -> EBS

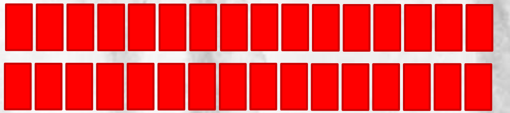
● CPU: No HT, single socket, scheduling

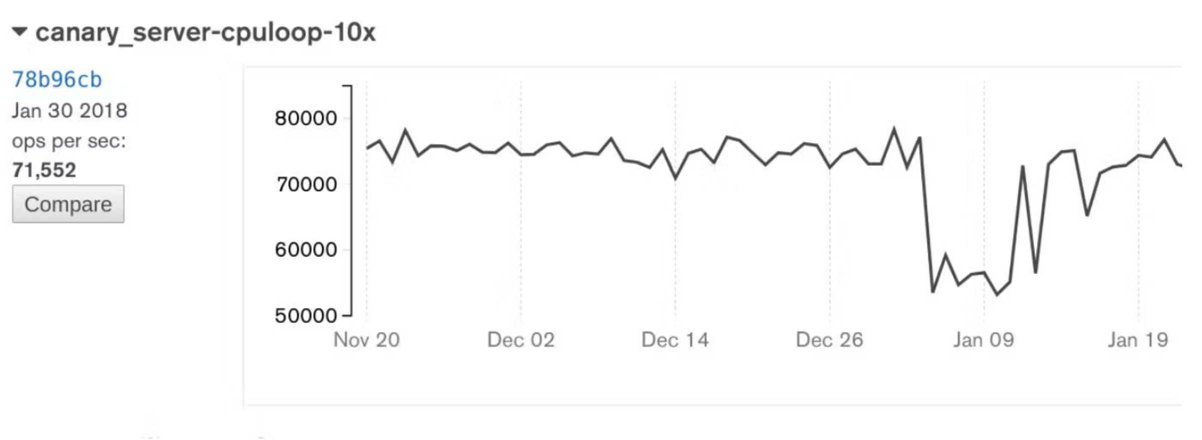
Noise range for all tests: 5%

CANARIES

Add tests that measure your infrastructure.

- CPU
- Disk
- Network





canary_server-cpuloop-10x

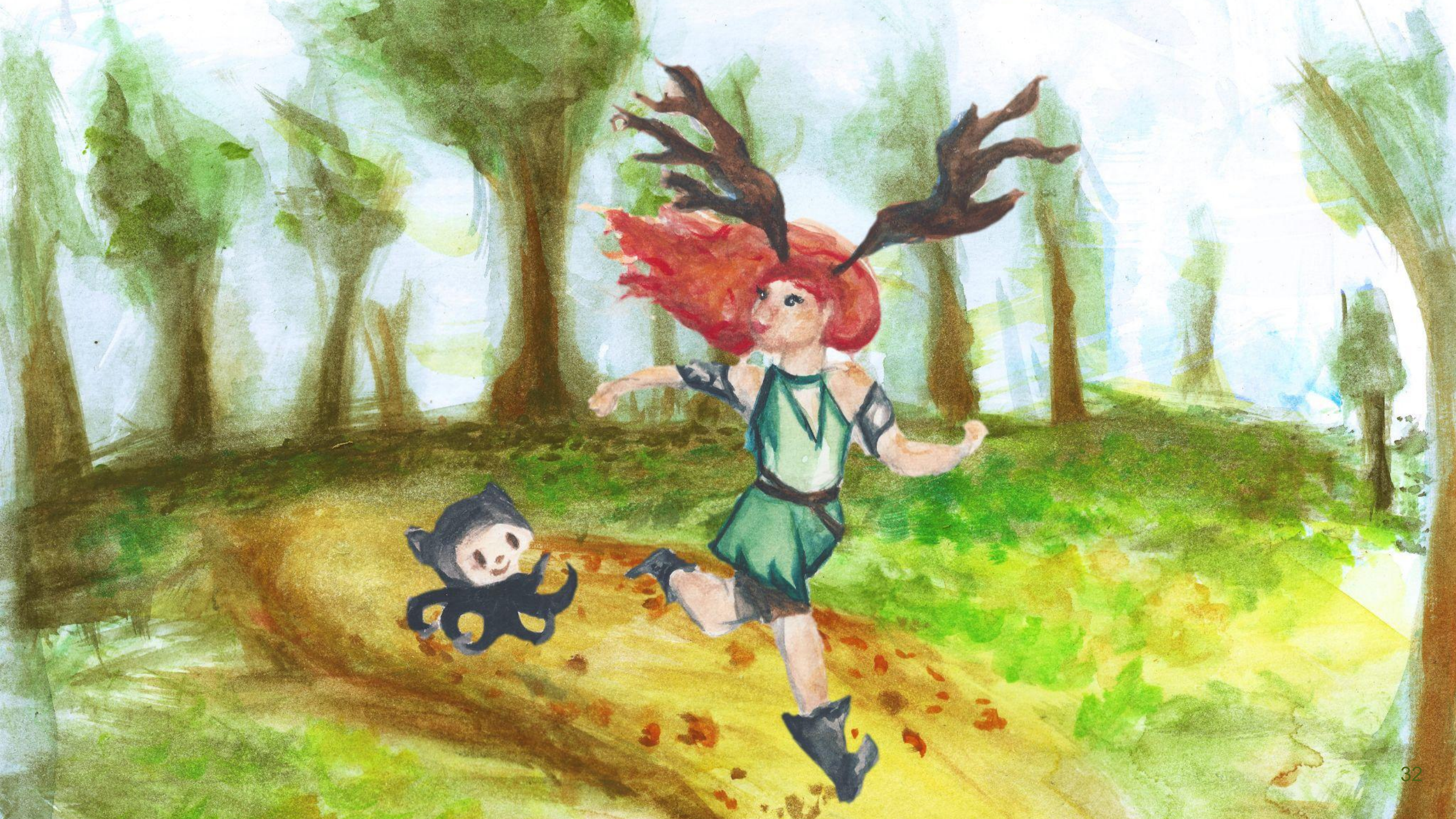
78b96cb

Jan 30 2018

ops per sec:

71,552

10 years later…



NYRKIÖ

GitHub Runners

NYRKIÖ

GitHub Runners

1. Install as GitHub app: nyrkio.com
2. Pick a subscription (20 c/h)
3. In workflow.yml:

```
runs-on : nyrkio_2
```

C7a instances
Hand picked &
Carefully tuned

NOT best performance
NOT for price/performance

But

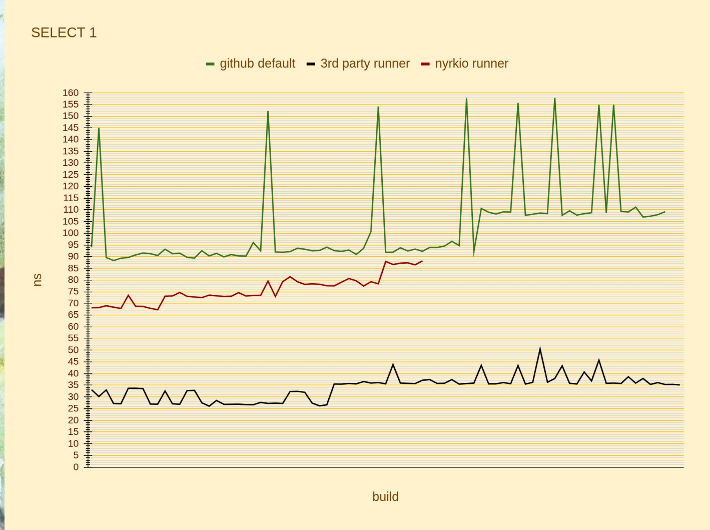
REPEATABLE
performance

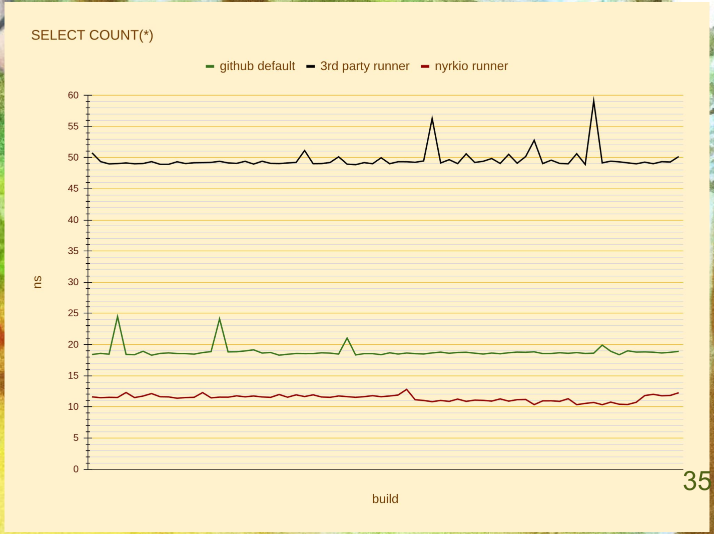
NYRKIÖ

GitHub Runners

min-max ranges (ns)		
github default	3rd party runner	nyrkio runner
88	35	73
154	51	75
66 ns	15 ns	**2 ns**
75%	44%	3%

min-max ranges (ns)		
github default	3rd party runner	nyrkio runner
18	49	11
25	59	13
6 ns	10 ns	**1 ns**
30%	21%	12%





ANY DIFFERENCE IN 10 YEARS?

C7a family offers high fidelity performance for 100% of the price

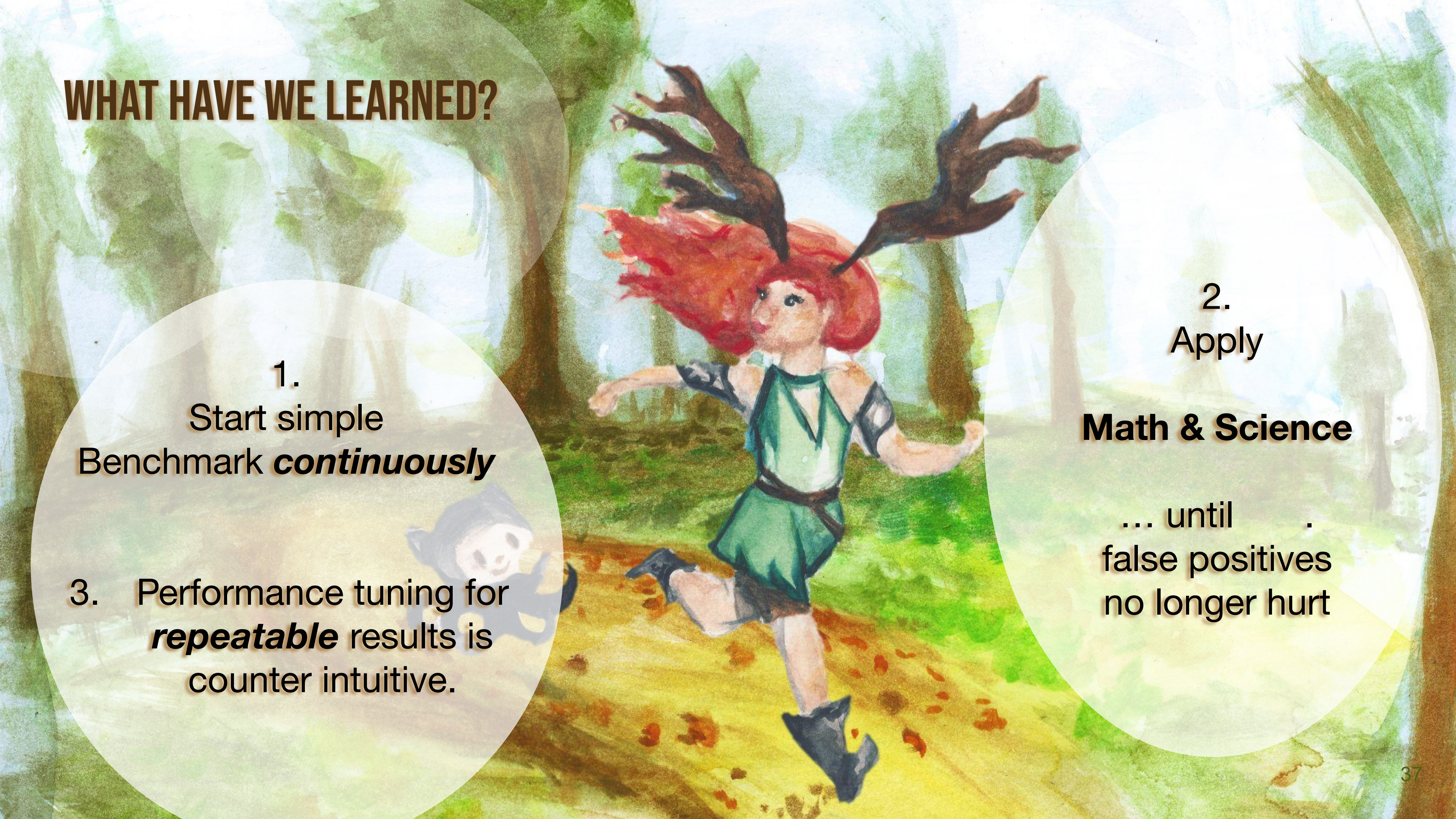
- No hyperthreading
- Single CPU socket
- AMD?

> 4x cheaper

Future opportunities:

- Sub nano-second precision
- Dist-sys: Run K8s cluster, on 128 core server

WHAT HAVE WE LEARNED?



1.
Start simple
Benchmark ***continuously***

2.
Apply

Math & Science

… until .
false positives
no longer hurt

3. Performance tuning for ***repeatable*** results is counter intuitive.

CREDITS AND REFERENCES

Nyyrikki and cat running, watercolor: Ebba Ingo

Velociraptor: Wikimedia commons

Otava test data graphs:
Joe Drumgoole & Claude

Everyone who contributed to Change Detection, now known as Apache Otava (incubating)

Ingo,Daly: Reducing Variability in Performance Tests on EC2
www.youtube.com/watch?v=3kHGZ7niHl4

David Daly et.al.: The Use of Change Point Detection to Identify Software Performance Regressions in a Continuous Integration System, 2020.

Fleming & Kołaczkowski: Hunter: Using Change Point Detection to Hunt for Performance Regressions, 2023.

Fixing Performance Regressions Before they Happen (Netflix)

8 Years of Optimizing Apache Otava: How disconnected open source developers took an algorithm from n^3 to constant time

Processor state control for Amazon EC2 Linux instances