Measure two designs so the difference is real

Key takeaways

1. A measurement answers a specific question. Throughput, tail latency, and scaling describe different parts of a system's behavior.

2. Repeated runs expose variation. Standard deviation measures spread; dividing it by the mean puts that spread on a relative scale.

3. A design comparison needs matched runs and an estimate of uncertainty. A useful improvement must also justify the cost of the change.

Common misunderstandings of performance measurement

Which claim would you defend right now? Choose before the evidence.

"If two implementations have the same average latency, users will experience them the same way."

"A speedup larger than the coefficient of variation proves that the new design is faster."

"Enough repetitions will remove a bias in how the experiment is run."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

Instapoll

Instapoll · Review · Oct 8 B

An agent runs design A ten times, then design B ten times. B’s mean runtime is 3% lower; each batch has a standard deviation of 1% of its mean. What does this establish about B?

- A. B is faster because its improvement exceeds the relative spread of either batch
- B. B is faster because ten repetitions make the comparison independent of run order
- C. B’s improvement is unresolved because design and execution order changed together
- D. B’s improvement is unresolved because performance claims always require 30 runs

Throughput, tail latency, and scaling

Throughput
Completed operations / elapsed time.

How much work finishes at a stated load and thread count?

p99 latency
99% of measured requests finish at or below this latency.

How long do slow requests wait?

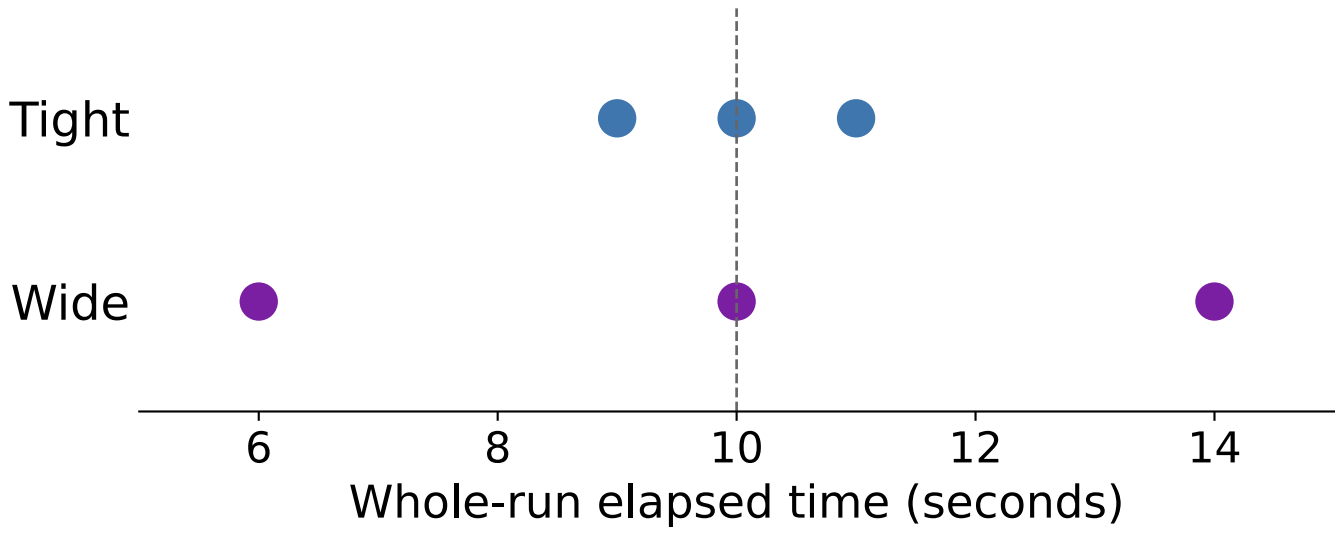
Scaling
Throughput T(k) as worker count k grows.

Report T(k) and T(k) / T(1); a slow baseline can flatter a ratio.

Illustration: a batcher can raise operations/second while requests wait longer for each batch. Its throughput improves while p99 worsens.

Request latencies form one distribution; whole-run results form another.

Variance and standard deviation



Illustrative samples: tight = [9, 10, 11]; wide = [6, 10, 14].

Both means are 10 s. The wide sample varies much more.

Variance averages squared deviations using n − 1. Standard deviation is its square root.

$\bar{x} = \text{sum}(x) / n$

$s^2 = \Sigma(x - \bar{x})^2 / (n - 1)$

$s = \sqrt{s^2}$

Tight: variance 1 s^2, std dev 1 s. Wide: variance 16 s^2, std dev 4 s.

source: NIST, paired-observation statistics: https://www.itl.nist.gov/div898/handbook/prc/section3/prc311.htm

Coefficient of variation

CV = standard deviation / mean expresses relative spread.

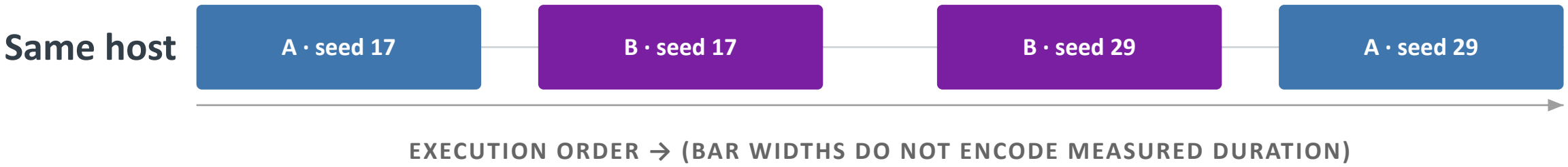
Illustrative run times	Mean	Std dev	CV
Short, variable	10 ms	1 ms	10%
Long, stable	100 ms	1 ms	1%
Long, variable	100 ms	10 ms	10%

```
from statistics import mean, stdev
runs_ms = [9, 10, 11]
cv = stdev(runs_ms) / mean(runs_ms)  # 0.10 = 10%
```

CV needs a meaningful zero and is unstable near zero. It does not measure confidence.

source: NIST, Coefficient of Variation: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/coefvari.htm

Paired runs isolate the design change



Within each pair	Same host, CPU affinity, workload, seed, build flags, and starting state. Change only the design.
Across pairs	Reverse A/B order to reduce warm-up or time-of-day bias. Vary seeds between pairs when the claim covers different inputs.
Compare	Compute each pair’s difference or ratio first. Keep the paired results and all run metadata.

Run A and B sequentially. Concurrent benchmarks compete for the same hardware.

The state at the start is part of the experiment

1. **Freeze the comparison**
   Record commits, build flags, thread counts and CPU pinning. Check correctness first.

2. **Prepare equivalent state**
   Use separate clean CARGO_TARGET_DIRs. Build before timing; reset run outputs.

3. **Warm up deliberately**
   Give both designs the same untimed warm-up. Include startup when measuring cold starts.

4. **Keep the planned repetitions**
   Retain raw results and failures. A fixed seed does not fix thread scheduling.

A clean output directory does not clear the OS page cache. Name the cache state.

source: pyperf benchmarking guide: https://pyperf.readthedocs.io/en/latest/run_benchmark.html

A median describes a typical run

Five whole runs

```
from statistics import mean, median
runs_s = [10, 11, 9, 10, 30]

mean(runs_s)    # 14 seconds
median(runs_s)  # 10 seconds
```

Illustrative data. Keep the 30-second run and investigate whether it is a slow path or interference.

What the summaries say

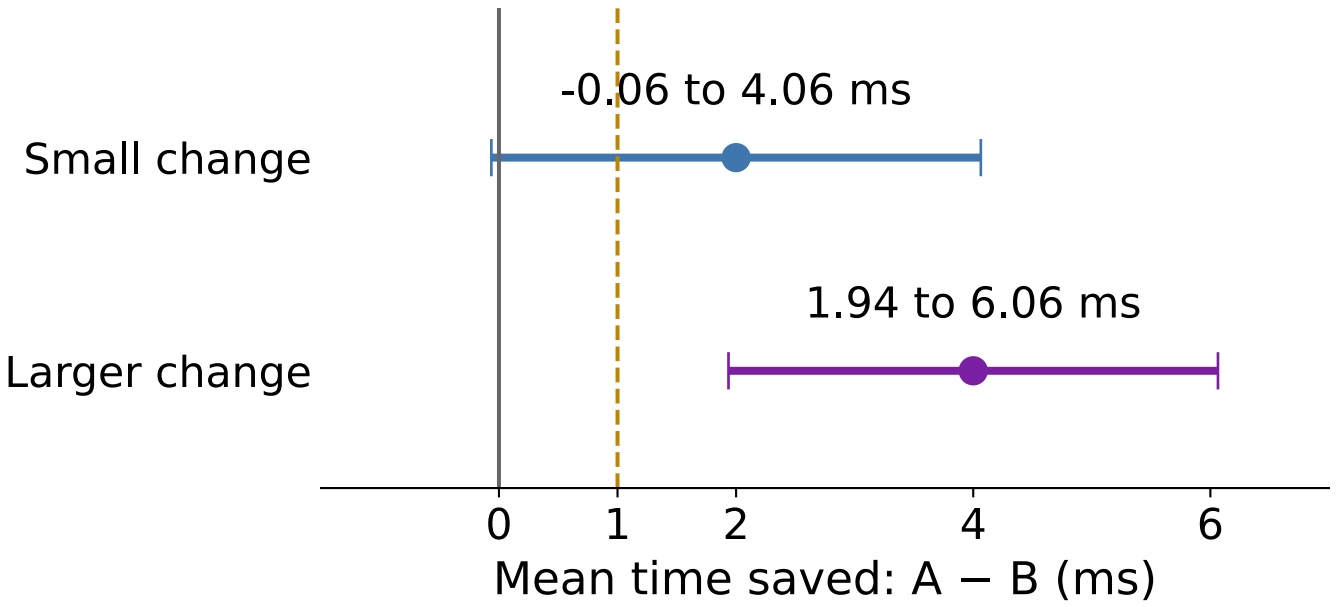
The mean includes every cost. The median is the middle sorted run and resists an extreme observation.

Near a timing boundary, start with at least three whole runs per condition. Close results need more.

Request p99 needs enough requests in each run. Three run summaries cannot estimate it.

Choose the repetition count and summary before seeing which design wins.

A finding must survive uncertainty and matter



95% confidence intervals for mean savings.
Illustrative: 25 pairs; s_d = 5 ms.

d = A time − B time; positive means B is faster.

SE = s_d / √n
95% CI = mean(d) ± t × SE

SE measures uncertainty in the mean; s_d measures run spread.

Small change: direction unresolved.
Larger change: clears the prechosen 1 ms threshold (dashed).

This t interval assumes independent, roughly symmetric paired differences; n counts pairs.

source: NIST, paired-mean confidence interval (t = 2.064 for 24 df): https://www.itl.nist.gov/div898/handbook/prc/section3/prc312.htm

V2 compares two prototypes against one floor

The V2 comparison

- Two structurally distinct prototypes share a delete-correct base and pass the full public suite.
- Pair each with the floor on both supplied curves and one changed workload. Change keys or operations; a new seed does not count.
- V2_DESIGN.md records both commits, raw logs, the choice and what would change it.

Local runs and grades

- Four alternating pairs per point. Keep the same worker sweep and repeat policy.
- At least four logical CPUs; six preferred. No oversubscription.
- Grading uses its own inputs and machine. Self-reported local numbers do not enter the grade.

V2 carries forward the PR round’s requirement for reproducible evidence.

source: Your concurrent-index kit: SPEC_v2.md and V2_DESIGN.template.md; your tm-compute kit: PR_ROUND.md

Key Takeaways

1. Name the metric and workload before measuring. Throughput, p99 latency and scaling answer different questions. Keep the absolute values when reporting ratios.

2. Keep repeated runs and their spread. Standard deviation has the metric's units; CV divides it by the mean. A median describes a typical run and can hide expensive tails.

3. Compare matched pairs under equivalent conditions. Estimate uncertainty in the difference and require a useful benefit. Extend the claim only when you measure the new conditions.