

Mutex or channel in Go? Measured under contention

We measured sync.Mutex and channels in Go 1.27 across three scenarios and different goroutine and P counts: which is faster depends on G and P.

01 GO · CONCURRENCY · MEASUREMENT

Mutex or channel in Go?

Two ways to protect shared state, measured under contention in three scenarios. There is no short answer: the result changes with the number of goroutines (G) and P.

Muhammet Şafak · October 3, 2026



02 CONTEXT FIRST

Measurement environment and confidence level

- ENVIRONMENT**
Go 1.27.1 linux/arm64; Docker Engine 29.8.0, 12 vCPU, Apple M4 Pro. P values are vCPU groups separated with cpuset; the container has no network, is read-only, 4 GiB.
- LOW CONFIDENCE**
518 of 845 cells (61.3%) are flagged; the threshold is 15%. Speed and latency ratios come from unflagged cells; the one flagged cell cited in the visual is marked with its flag.
- LATENCY**
Latency values are the upper bounds of logarithmic buckets, not exact measurements.

03 SCENARIO 1 · G=64, P=4

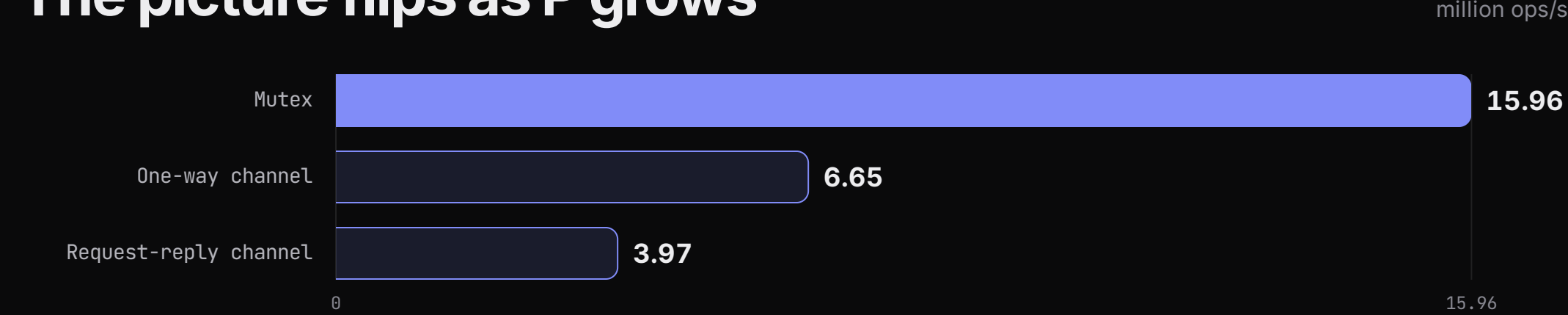
5x

mutex p99 latency, versus the one-way channel

Mutex p99 is 491,519 ns, one-way channel 98,303 ns. In throughput the gap is 3.2% and within noise: 7.69 M versus 7.94 M ops/s.

04 SCENARIO 1 · G=64, P=8

The picture flips as P grows



Mutex is 2.40× faster than the one-way channel and 4.02× faster than the request-reply channel. From P=4 to P=8, mutex sped up 2.08×, while the channel dropped to 0.84×; the cause was not measured.

05 SCENARIO 1 · G=10,000, P=8

Many goroutines: speed and fairness

METHOD	OPS/S	JAIN FAIRNESS INDEX	FLAG
Mutex	14,878,958	0.8237 (not compared)	fairness_unexplained
One-way channel	2,052,681 (7.25× slower)	0.9999	none
Request-reply channel	1,778,745 (8.36× slower)	1.0000	none

06 READING THE FAIRNESS RESULT

The mutex's unfairness could not be explained

- WARNING**
The correct reading is not “mutex is less fair” but “the unfairness in the mutex cell cannot be explained.” The cell carries the fairness_unexplained flag.
- NOTE**
By the source's design rule, cells carrying this flag do not enter the fairness comparison across models. The mutex cell carries only the fairness flag; its speed ratio stands.

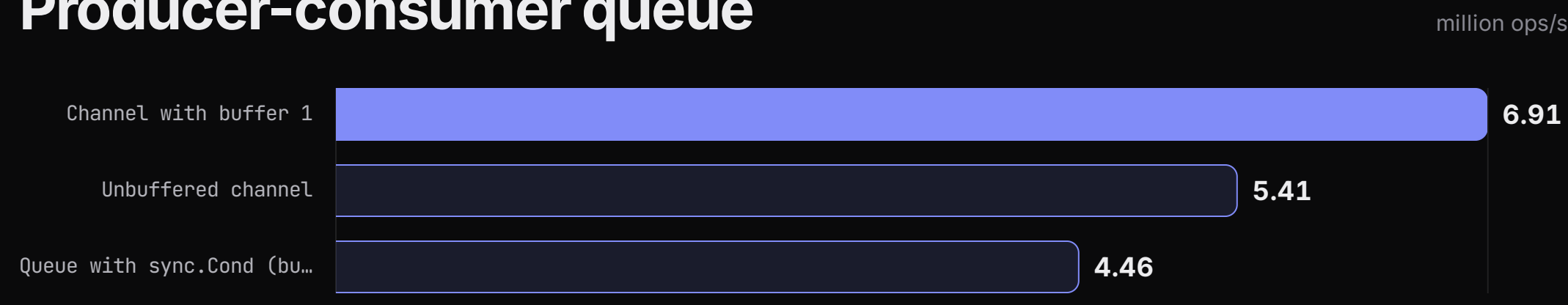
07 SCENARIO 2 · G=64, P=4

Read-heavy load

READ RATIO	MUTEX	OWNER-GOROUTINE CHANNEL	DIFFERENCE
50%	9.86 M ops/s	4.42 M ops/s	Mutex 2.23× faster
90%	12.23 M ops/s	4.58 M ops/s	Mutex 2.67× faster

08 SCENARIO 3 · G=64, P=4

Producer-consumer queue



The channel with buffer 1 is 1.28× faster than the unbuffered one and 1.55× faster than the sync.Cond queue. The channel cells with buffers 64 and 1,024 are flagged and left out.

09 FLAGGED CELLS

What we did not claim

- SCENARIO 1**
The P=1 and P=2 cells are flagged; no result is stated for those P values.
- SCENARIO 2**
The 99% read cell is flagged, so no ratio is given. The RWMutex cells carry the observer_effect flag.
- FLAGS**
fairness_unexplained 235, spread_p99 189, observer_effect 171, spread_throughput 125, calib_drift 72, spread_persistent 36.

10 LIMITS

What this measurement does not tell you

- WARNING**
The load is a synthetic counter, not a production workload.
- NOTE**
sync.Map was not measured.
- NOTE**
P values are vCPUs inside a virtual machine; they are not physical cores.
- NOTE**
The causes of the differences were not measured; the visual only shows what happened.

11 CONCLUSION

The answer to
“which is faster?”
depends on G and P.

This record does not say which is faster, but when each is faster. Don't assume a behavior in production when you don't know its cause; measure on your own P.