BENCHMARK · ARM64 VS X86_64

Imaxe Monitor Solution - Zabbix 7.0 1.2

Measured comparison between the Graviton image we publish and an identical x86_64 build, on equivalent instance types.

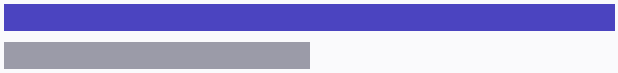
Ref. PERF-zabbix70-1.2-20261107 Generated 2026-11-07 us-east-1

HEADLINE

Graviton delivers +14,1 % in median performance across the measured tests.

Colour convention throughout this document: **ARM64 (Graviton)** · **x86_64**. Deltas also carry a symbol (▲ / ▼) so the report reads in black and white.

1 Results

TEST	ARM64 · C7G.LARGE	X86_64 · C6I.LARGE	Δ ARM64	COMPARISON
sysbench — CPU (1 hilo)	**1113 events/s**	1074 events/s	▲ +3,6 %	
sysbench — CPU (todos los hilos)	**2221 events/s**	1113 events/s	▲ +99,5 %	
sysbench — memoria	**40914 MiB/s**	19077 MiB/s	▲ +114,5 %	
openssl — AES-256-GCM	**4147 MB/s**	8772 MB/s	▼ -52,7 %	
openssl — SHA-256	**1600 MB/s**	1284 MB/s	▲ +24,6 %	
gzip — compresión de un corpus fijo	**129,6 MB/s**	155,6 MB/s	▼ -16,7 %	
fio — lectura aleatoria 4k	**3043 IOPS**	3020 IOPS	▲ +0,8 %	
fio — escritura aleatoria 4k	**3047 IOPS**	3047 IOPS	▼ 0,0 %	
sysbench — CPU, latencia p95	**0,90 ms**	1,82 ms	▲ +102,2 %	
fio — lectura 4k, latencia p99	**5800 µs**	29230 µs	▲ +404,0 %	

Median advantage for Graviton across the comparable tests: +14,1 %. Each bar is scaled to the larger of its own pair, because every test has its own unit.

2 Scaling across threads

PARAMETER	ARM64	X86_64
CPU on 1 thread	**1113 events/s**	1074 events/s
CPU on all threads	**2221 events/s**	1113 events/s
Scaling factor	**×2,00 (2 vCPU)**	×1,04 (2 vCPU)
Efficiency against the ideal	**99,8 %**	51,8 %



Efficiency against the ideal

Going from one thread to every vCPU, Graviton multiplies its work by 2,00 and the x86_64 instance by 1,04.

> A vCPU is not the same thing in the two families: on Graviton it is a physical core, on Intel it is an SMT thread sharing execution units with its twin. That is why a single thread can look similar on both and the difference only shows once all of them are busy — which is how the product actually works: nginx with its workers, PHP-FPM with its pool and the database with its connections use every vCPU at once, not one.

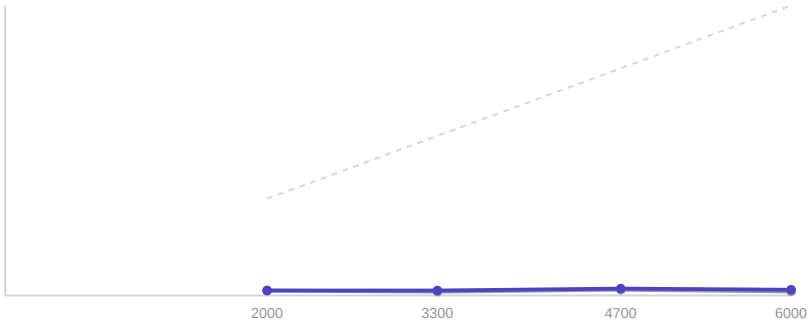
3 Methodology

▸ Test environment

PARAMETER	ARM64	X86_64
Region	us-east-1	us-east-1
Instance type	c7g.large	c6i.large
CPU and memory	2 vCPU · 3799 MiB	2 vCPU · 3811 MiB
Processor	Neoverse-V1	Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz
Kernel	7.0.0-1017-aws	7.0.0-1017-aws

> Both architectures are measured on their own test instance, after that instance has passed the product test suite. The product is running while the benchmark runs, which is what a buyer's machine actually looks like.

4 Load curve



Requests delivered against requests requested. The dashed diagonal is the requested rate: where a line leaves it, that machine is saturated.

REQUESTED RATE	ARM64	X86_64	ARM64 P99	X86_64 P99
2000	**103,9** ▲	104,2 ▲	**67057 ms**	56975 ms
3300	**101,7** ▲	78,8 ▲	**49891 ms**	41113 ms
4700	**142,4** ▲	119,5 ▲	**43587 ms**	53015 ms
6000	**119,7** ▲	84,0 ▲	**44825 ms**	62663 ms

> ▲ marks a step where the machine no longer sustains the requested rate. Past that point the p99 measures queueing, not latency.

> The load comes from a separate machine — the same instance type for both architectures, in the same availability zone — hitting the measured machine over its private address at a constant arrival rate. The client is identical in both runs: the only thing that changes is the machine answering.

▸ Capacity and cost of the work

CONCEPT	ARM64	X86_64	Δ ARM64
Maximum sustained rate	**142,4 req/s**	119,5 req/s	**▲ +19,1 %**

> The ramp started above what one of the two could take, so for that one there is no clean step and what is shown is the highest throughput observed — a ceiling, not a sustained rate.

> The maximum sustained rate is the last step the machine served in full, not the peak of the chart: a saturated step delivers a lot and answers badly, and counting it as capacity would promise a load at which the product is already struggling. Cost per million requests is computed at that rate, and it is the comparison that holds when the two machines do not take the same load: price per hour says what it costs to keep a machine on, not what the work it does costs.

5 Conclusion

Imaxe Monitor Solution - Zabbix 7.0 is published on Graviton (arm64) because it measures better and costs less: +14,1 % median advantage over the equivalent x86_64 instance.

> Every figure in this report comes from a run on the image being published. Tests that could not be measured on both architectures are shown as such and excluded from the median.

A Appendix — commands run

> Exactly what was executed, so the numbers can be reproduced or disputed.

TEST	COMMAND
sysbench — CPU (1 hilo)	`sysbench cpu --cpu-max-prime=20000 --threads=1 --time=60 run`
sysbench — CPU (todos los hilos)	`sysbench cpu --cpu-max-prime=20000 --threads=$(nproc) --time=60 run`
sysbench — memoria	`sysbench memory --memory-block-size=1M --memory-total-size=100G --threads=2 run`
openssl — AES-256-GCM	`openssl speed -elapsed -evp aes-256-gcm -seconds 3 2>&1`
openssl — SHA-256	`openssl speed -elapsed -evp sha256 -seconds 3 2>&1`
gzip — compresión de un corpus fijo	`f=/mnt/bench/corpus.txt; s=$(stat -c%s $f); t0=$(date +%s%N); i=0; while [ $i -lt 3 ]; do gzip -6 -c $f >/dev/null; i=$((i+1)); done; t1=$(date +%s%N); awk -v s=$s -v a=$t0 -v b=$t1 -v n=3 'BEGIN{printf "throughput: %.2f MB/s\n", n*s/1000000/((b-a)/1000000000)}'`
fio — lectura aleatoria 4k	`fio --name=randread --ioengine=libaio --direct=1 --bs=4k --iodepth=16 --rw=randread --size=4G --runtime=60 --time_based --filename=/mnt/bench/testfile`
fio — escritura aleatoria 4k	`fio --name=randwrite --ioengine=libaio --direct=1 --bs=4k --iodepth=16 --rw=randwrite --size=4G --runtime=60 --time_based --filename=/mnt/bench/testfile`
sysbench — CPU, latencia p95	`sysbench cpu --cpu-max-prime=20000 --threads=$(nproc) --time=60 run`
fio — lectura 4k, latencia p99	`fio --name=randread --ioengine=libaio --direct=1 --bs=4k --iodepth=16 --rw=randread --size=4G --runtime=60 --time_based --filename=/mnt/bench/testfile`