BENCHMARK · ARM64 CONTRA X86_64

Imaxe Monitor Solution - Zabbix 7.0 1.2

Comparação medida entre a imagem Graviton que publicamos e uma compilação x86_64 idêntica, em tipos de instância equivalentes.

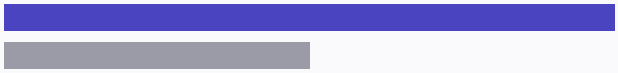
Ref. PERF-zabbix70-1.2-20261107 Gerado 2026-11-07 us-east-1

NUMA LINHA

O Graviton rende +14,1 % de desempenho mediano nos testes medidos.

Convenção de cor em todo o documento: **ARM64 (Graviton)** · **x86_64**. Os deltas levam também símbolo (▲ / ▼) para leitura a preto e branco.

1 Resultados

TESTE	ARM64 · C7G.LARGE	X86_64 · C6I.LARGE	Δ ARM64	COMPARAÇÃO
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 %	

Vantagem mediana do Graviton nos testes comparáveis: +14,1 %. Cada barra é escalada ao maior do seu próprio par, porque cada teste tem a sua unidade.

2 Escalabilidade nos threads

PARÂMETRO	ARM64	X86_64
CPU com 1 thread	**1113 events/s**	1074 events/s
CPU com todos os threads	**2221 events/s**	1113 events/s
Fator de escalabilidade	**×2,00 (2 vCPU)**	×1,04 (2 vCPU)
Eficiência face ao ideal	**99,8 %**	51,8 %



Eficiência face ao ideal

Ao passar de um thread para todas as vCPU, o Graviton multiplica o seu trabalho por 2,00 e a instância x86_64 por 1,04.

> Uma vCPU não é a mesma coisa nas duas famílias: no Graviton é um núcleo físico, na Intel é um thread SMT que partilha as unidades de execução com o gémeo. Por isso um único thread pode dar números parecidos e a diferença só aparece quando estão todos ocupados — que é como o produto trabalha: o nginx com os seus workers, o PHP-FPM com o seu pool e a base de dados com as suas ligações usam todas as vCPU ao mesmo tempo, não uma.

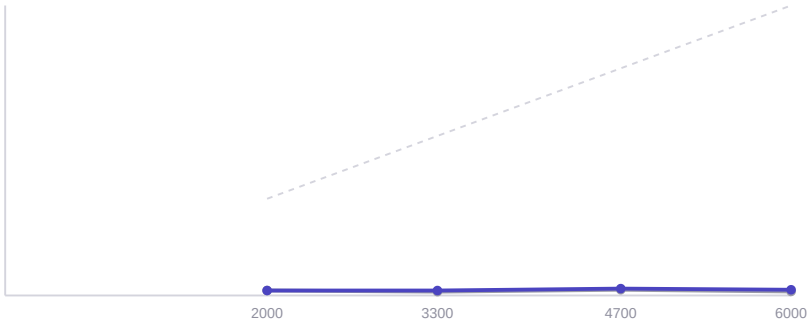
3 Metodologia

▸ Ambiente de testes

PARÂMETRO	ARM64	X86_64
Região	`us-east-1`	`us-east-1`
Tipo de instância	`c7g.large`	`c6i.large`
CPU e memória	`2 vCPU · 3799 MiB`	`2 vCPU · 3811 MiB`
Processador	`Neoverse-V1`	`Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz`
Kernel	`7.0.0-1017-aws`	`7.0.0-1017-aws`

> Cada arquitetura é medida na sua própria instância de teste, e só depois de essa instância ter passado os testes do produto. O produto está a correr durante a medição, que é como está a máquina de um comprador.

4 Curva de carga



Pedidos entregues face a pedidos solicitados. A diagonal tracejada é a taxa pedida: onde uma linha se afasta, essa máquina satura.

TAXA PEDIDA	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

> ▲ marca um degrau onde a máquina já não sustenta a taxa pedida. A partir daí o p99 mede fila de espera, não latência.

> A carga é gerada por uma máquina à parte — do mesmo tipo nas duas arquiteturas e na mesma zona de disponibilidade — que ataca a máquina medida pelo seu endereço privado a uma taxa de chegada constante. O cliente é idêntico nas duas passagens: só muda a máquina que responde.

▸ Capacidade e custo do trabalho

RUBRICA	ARM64	X86_64	Δ ARM64
Taxa máxima sustentada	**142,4 req/s**	119,5 req/s	**▲ +19,1 %**

> A rampa começou acima do que uma das duas aguentava, pelo que para essa não há nenhum degrau limpo e o que se mostra é o caudal máximo observado: um teto, não uma taxa sustentada.

> A taxa máxima sustentada é o último degrau que a máquina serviu por inteiro, não o pico do gráfico: um degrau saturado entrega muito e responde mal, e contá-lo como capacidade seria prometer uma carga em que o produto já vai mal. O custo por milhão de pedidos calcula-se a essa taxa, e é a comparação que vale quando as duas máquinas não aguentam o mesmo: o preço por hora diz quanto custa tê-la ligada, não quanto custa o trabalho que faz.

5 Conclusão

Imaxe Monitor Solution - Zabbix 7.0 publica-se em Graviton (arm64) porque mede melhor e custa menos: +14,1 % de vantagem mediana sobre a instância x86_64 equivalente.

> Todos os números vêm de uma passagem sobre a imagem publicada. Os testes que não puderam ser medidos nas duas arquiteturas aparecem como tal e ficam fora da mediana.

A Apêndice — comandos executados

> Exatamente o que foi executado, para que os números possam ser reproduzidos ou contestados.

TESTE	COMANDO
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`