BENCHMARK · ARM64 GEGEN X86_64

Imaxe Monitor Solution - Zabbix 7.0 1.2

Gemessener Vergleich zwischen dem veröffentlichten Graviton-Image und einem identischen x86_64-Build auf gleichwertigen Instanztypen.

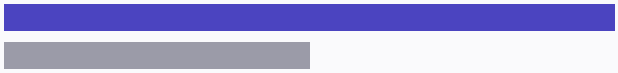
Ref. PERF-zabbix70-1.2-20261107 Erstellt 2026-11-07 us-east-1

KERNAUSSAGE

Graviton liefert im Median +14,1 % Leistung über die gemessenen Tests.

Farbkonvention im gesamten Dokument: **ARM64 (Graviton)** · **x86_64**. Deltas tragen zusätzlich ein Symbol (▲ / ▼) für Schwarzweißdruck.

1 Ergebnisse

TEST	ARM64 · C7G.LARGE	X86_64 · C6I.LARGE	Δ ARM64	VERGLEICH
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 %	

Medianvorteil von Graviton über die vergleichbaren Tests: +14,1 %. Jeder Balken ist auf das Maximum seines eigenen Paares skaliert, da jeder Test seine eigene Einheit hat.

2 Skalierung über die Threads

PARAMETER	ARM64	X86_64
CPU mit 1 Thread	**1113 events/s**	1074 events/s
CPU mit allen Threads	**2221 events/s**	1113 events/s
Skalierungsfaktor	**×2,00 (2 vCPU)**	×1,04 (2 vCPU)
Effizienz gegenüber dem Ideal	**99,8 %**	51,8 %



Effizienz gegenüber dem Ideal

Von einem Thread auf alle vCPU vervielfacht Graviton seine Arbeit um 2,00, die x86_64-Instanz um 1,04.

> Eine vCPU ist in beiden Familien nicht dasselbe: bei Graviton ist sie ein physischer Kern, bei Intel ein SMT-Thread, der sich die Ausführungseinheiten mit seinem Zwilling teilt. Deshalb kann ein einzelner Thread auf beiden ähnlich aussehen, und der Unterschied zeigt sich erst, wenn alle ausgelastet sind — genau so arbeitet das Produkt: nginx mit seinen Workern, PHP-FPM mit seinem Pool und die Datenbank mit ihren Verbindungen nutzen alle vCPU gleichzeitig, nicht eine.

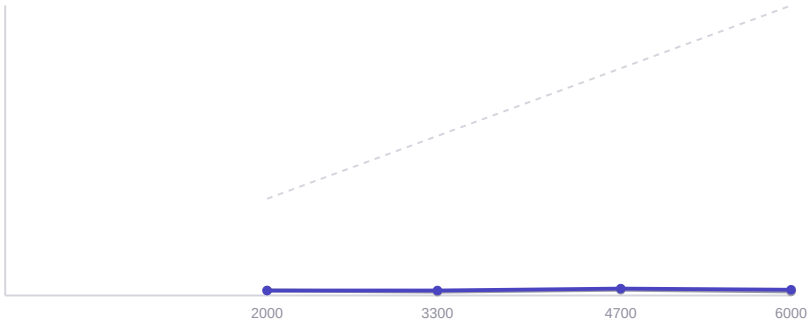
3 Methodik

▸ Testumgebung

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

> Jede Architektur wird auf ihrer eigenen Testinstanz gemessen, erst nachdem diese die Produkttests bestanden hat. Das Produkt läuft während der Messung — genau so sieht die Maschine eines Käufers aus.

4 Lastkurve



Ausgelieferte gegen angeforderte Anfragen. Die gestrichelte Diagonale ist die angeforderte Rate: wo eine Linie sie verlässt, ist die Maschine gesättigt.

ANGEFORDERTE 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

▲ markiert eine Stufe, auf der die Maschine die angeforderte Rate nicht mehr hält. Danach misst der p99 Wartezeit, nicht Latenz.

Die Last erzeugt eine separate Maschine — derselbe Instanztyp für beide Architekturen, in derselben Verfügbarkeitszone —, die die gemessene Maschine über ihre private Adresse mit konstanter Ankunftsrate belastet. Der Client ist in beiden Durchläufen identisch: es ändert sich nur die Maschine, die antwortet.

▸ Kapazität und Kosten der Arbeit

POSITION	ARM64	X86_64	Δ ARM64
Maximal gehaltene Rate	**142,4 req/s**	119,5 req/s	**▲ +19,1 %**

Die Rampe begann oberhalb dessen, was eine der beiden tragen konnte; für diese gibt es keine saubere Stufe, gezeigt wird der höchste beobachtete Durchsatz — eine Obergrenze, keine gehaltene Rate.

Die maximal gehaltene Rate ist die letzte Stufe, die die Maschine vollständig bedient hat, nicht der Höchstwert der Grafik: eine gesättigte Stufe liefert viel und antwortet schlecht, und sie als Kapazität zu zählen hieße, eine Last zu versprechen, bei der das Produkt schon schwächelt. Die Kosten pro Million Anfragen werden bei dieser Rate berechnet, und das ist der Vergleich, der zählt, wenn beide Maschinen nicht dieselbe Last tragen: der Stundenpreis sagt, was das Anlassen kostet, nicht was die geleistete Arbeit kostet.

5 Fazit

Imaxe Monitor Solution - Zabbix 7.0 wird auf Graviton (arm64) veröffentlicht, weil es besser misst und weniger kostet: +14,1 % Medianvorteil gegenüber der gleichwertigen x86_64-Instanz.

> Alle Zahlen stammen aus einem Lauf auf dem veröffentlichten Image. Tests, die nicht auf beiden Architekturen gemessen werden konnten, sind als solche ausgewiesen und fließen nicht in den Median ein.

A Anhang — ausgeführte Befehle

> Genau das, was ausgeführt wurde, damit die Zahlen reproduziert oder bestritten werden können.

TEST	BEFEHL
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`