BENCHMARK · ARM64 CONTRO X86_64

Imaxe Monitor Solution - Zabbix 7.0 1.2

Confronto misurato tra l'immagine Graviton che pubblichiamo e una build x86_64 identica, su tipi di istanza equivalenti.

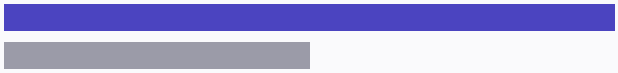
Rif. PERF-zabbix70-1.2-20261107 Generato 2026-11-07 us-east-1

IN UNA RIGA

Graviton rende +14,1 % in prestazioni mediane sui test misurati.

Convenzione di colore in tutto il documento: **ARM64 (Graviton)** · **x86_64**. I delta portano anche un simbolo (▲ / ▼) per la lettura in bianco e nero.

1 Risultati

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

Vantaggio mediano di Graviton sui test comparabili: +14,1 %. Ogni barra è scalata al maggiore della propria coppia, perché ogni test ha la sua unità.

2 Scalabilità sui thread

PARAMETRO	ARM64	X86_64
CPU con 1 thread	**1113 events/s**	1074 events/s
CPU con tutti i thread	**2221 events/s**	1113 events/s
Fattore di scalabilità	**×2,00 (2 vCPU)**	×1,04 (2 vCPU)
Efficienza rispetto all'ideale	**99,8 %**	51,8 %



Efficienza rispetto all'ideale

Passando da un thread a tutte le vCPU, Graviton moltiplica il proprio lavoro per 2,00 e l'istanza x86_64 per 1,04.

> Una vCPU non è la stessa cosa nelle due famiglie: su Graviton è un core fisico, su Intel è un thread SMT che condivide le unità di esecuzione con il gemello. Per questo un singolo thread può dare numeri simili e la differenza emerge solo quando sono tutti occupati — che è esattamente come lavora il prodotto: nginx con i suoi worker, PHP-FPM con il suo pool e il database con le sue connessioni usano tutte le vCPU insieme, non una.

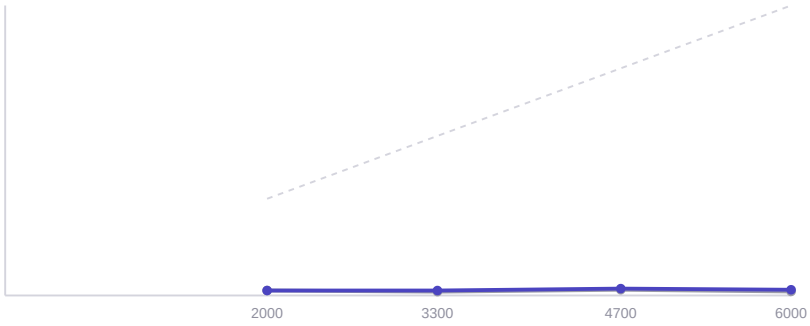
3 Metodologia

▸ Ambiente di prova

PARAMETRO	ARM64	X86_64
Regione	`us-east-1`	`us-east-1`
Tipo di istanza	`c7g.large`	`c6i.large`
CPU e memoria	`2 vCPU · 3799 MiB`	`2 vCPU · 3811 MiB`
Processore	`Neoverse-V1`	`Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz`
Kernel	`7.0.0-1017-aws`	`7.0.0-1017-aws`

> Ogni architettura viene misurata sulla propria istanza di test, e solo dopo che questa ha superato i test del prodotto. Il prodotto è in esecuzione durante la misura, che è com'è davvero la macchina di un acquirente.

4 Curva di carico



Richieste servite rispetto a quelle richieste. La diagonale tratteggiata è il tasso richiesto: dove una linea se ne stacca, quella macchina è satura.

TASSO RICHIESTO	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

> ▲ segna un gradino in cui la macchina non regge più il tasso richiesto. Da lì in poi il p99 misura la coda, non la latenza.

> Il carico lo genera una macchina separata — stesso tipo di istanza per entrambe le architetture, nella stessa zona di disponibilità — che colpisce la macchina misurata sul suo indirizzo privato a tasso di arrivo costante. Il client è identico nelle due passate: cambia solo la macchina che risponde.

▸ Capacità e costo del lavoro

VOCE	ARM64	X86_64	Δ ARM64
Frequenza massima sostenuta	**142,4 req/s**	119,5 req/s	**▲ +19,1 %**

> La rampa è partita sopra ciò che una delle due reggeva, quindi per quella non c'è alcun gradino pulito e ciò che si mostra è la portata massima osservata: un tetto, non una frequenza sostenuta.

> La frequenza massima sostenuta è l'ultimo gradino che la macchina ha servito per intero, non il picco del grafico: un gradino saturo consegna molto e risponde male, e contarlo come capacità significherebbe promettere un carico in cui il prodotto già arranca. Il costo per milione di richieste si calcola a quella frequenza, ed è il confronto che regge quando le due macchine non reggono lo stesso carico: il prezzo orario dice quanto costa tenerla accesa, non quanto costa il lavoro che fa.

5 Conclusione

Imaxe Monitor Solution - Zabbix 7.0 si pubblica su Graviton (arm64) perché misura meglio e costa meno: +14,1 % di vantaggio mediano sull'istanza x86_64 equivalente.

> Tutte le cifre provengono da un'esecuzione sull'immagine pubblicata. I test non misurabili su entrambe le architetture sono indicati come tali ed esclusi dalla mediana.

A Appendice — comandi eseguiti

> Esattamente ciò che è stato eseguito, perché i numeri si possano riprodurre o contestare.

TEST	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`