

BUDAPEST

KCD

GROWING CLOUD NATIVE TOGETHER

563 Things Karpenter Didn’t Know About

Gergely Daroczi & Attila Nagy, Spare Cores

Oct 30, 2026 @ Budapest, Hungary



<spare_cores>

Press Space or click the green arrow icons to navigate the slides →

A couple weeks ago
at an undisclosed location
not too far, far away...
Gergely and Attila met
and among other topics
they discussed how badly
their Python app scales...



<spare_cores>

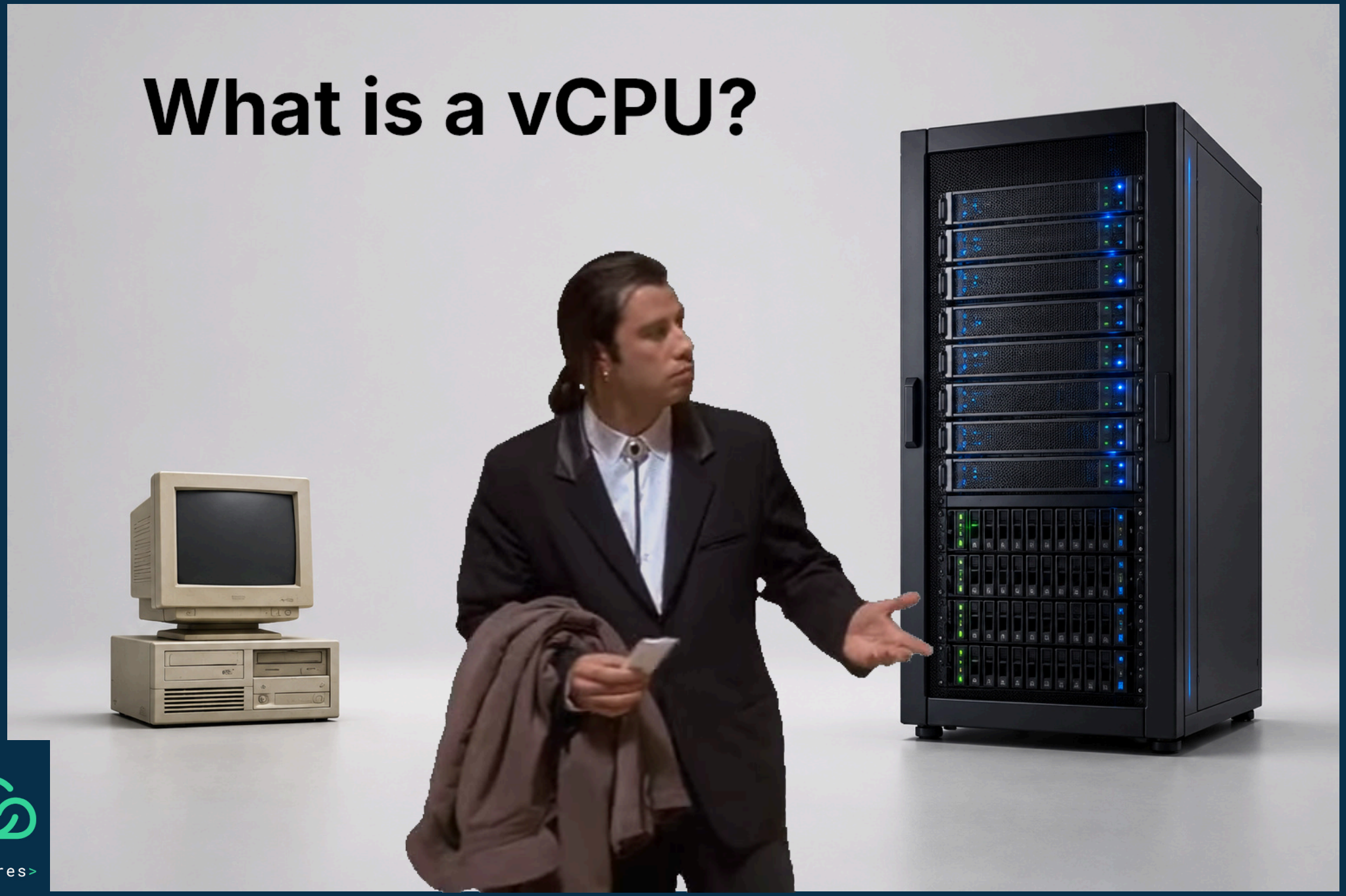

CORE ALONE
GILty




There is no cloud
it's just someone else's computer



What is a vCPU?



Guess the Number!

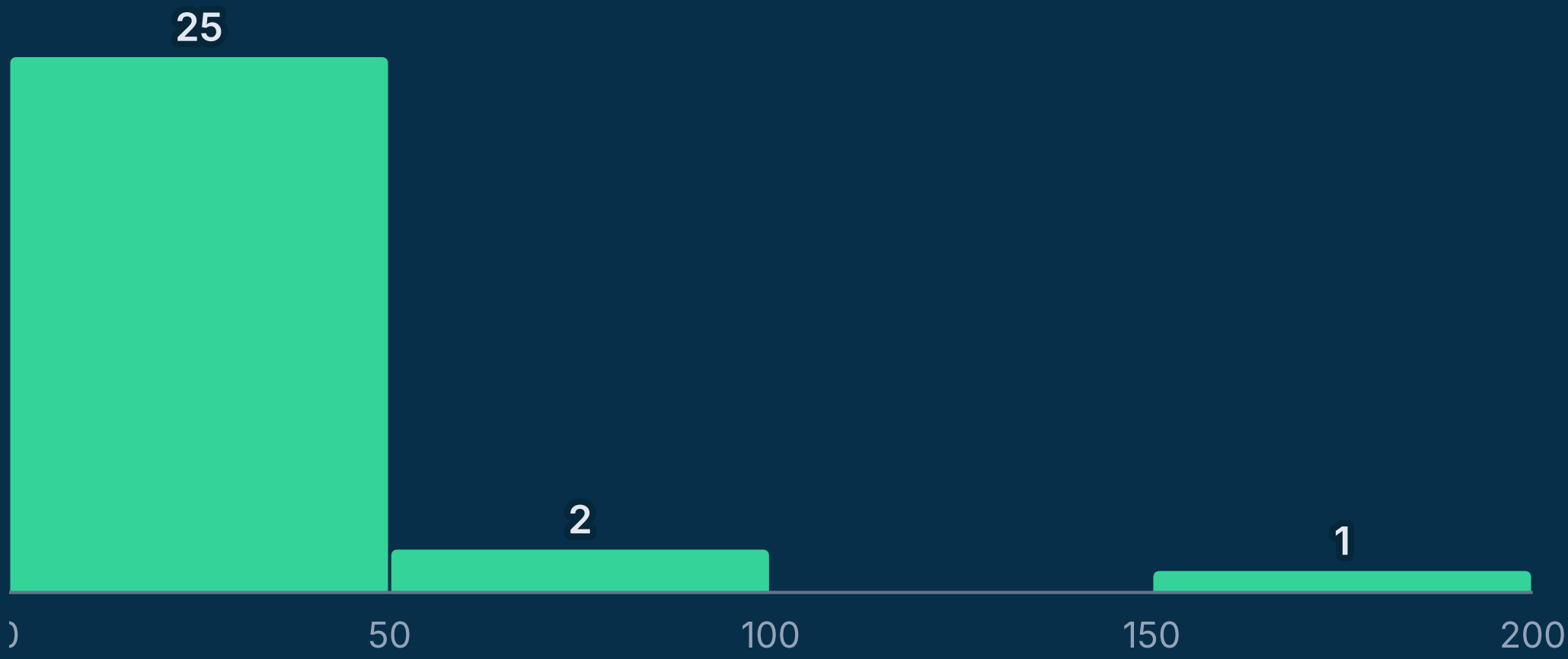
How much better is the highest performing 2-vCPU cloud server's single-core performance compared to the lowest?



https://sparecores.com/poll/kcd-2026

28 responses

Your guesses

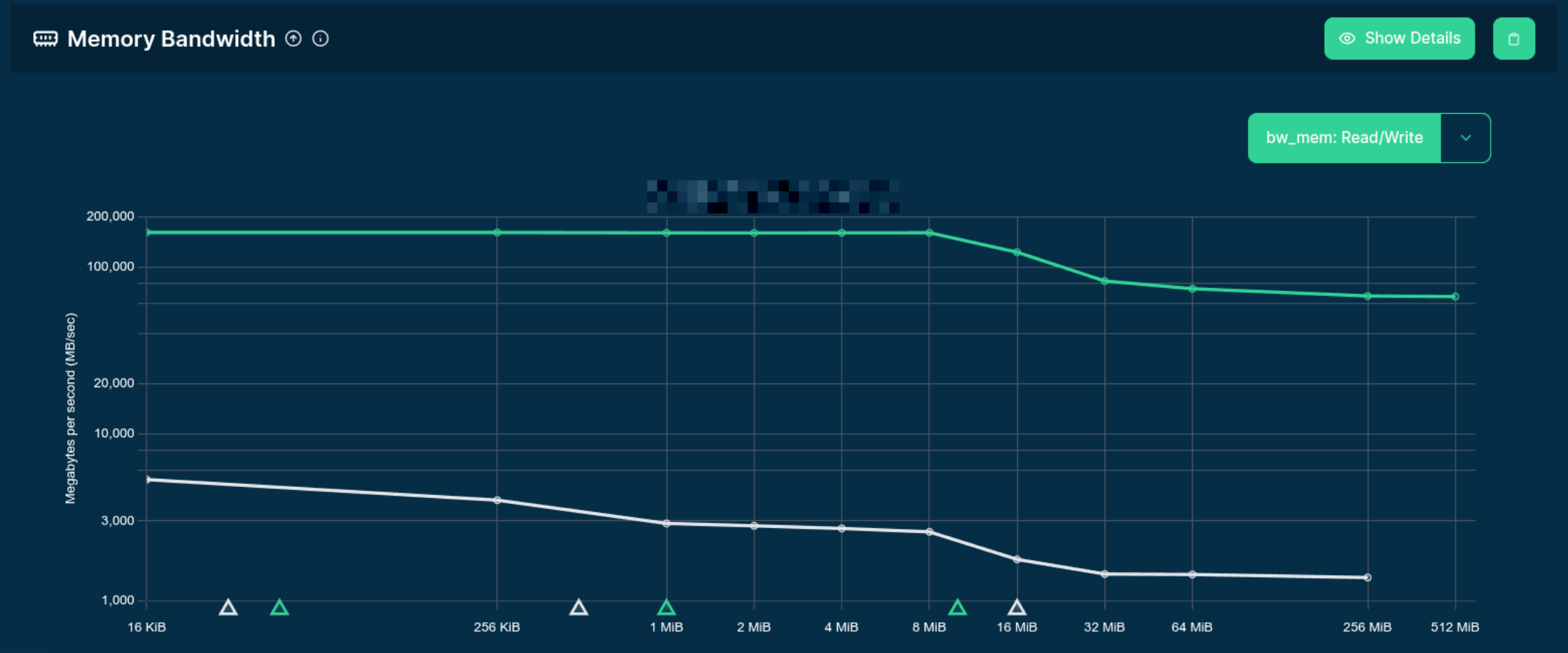


28 votes

	SERVER A	SERVER B



Credit: the_coding_love / When I launch a benchmark


Memory Bandwidth
Show Details
bw_mem: Read/Write
Megabytes per second (MB/sec)
200,000
100,000
20,000
10,000
3,000
1,000
16 KiB
256 KiB
1 MiB
2 MiB
4 MiB
8 MiB
16 MiB
32 MiB
64 MiB
256 MiB
512 MiB

<spare_cores>

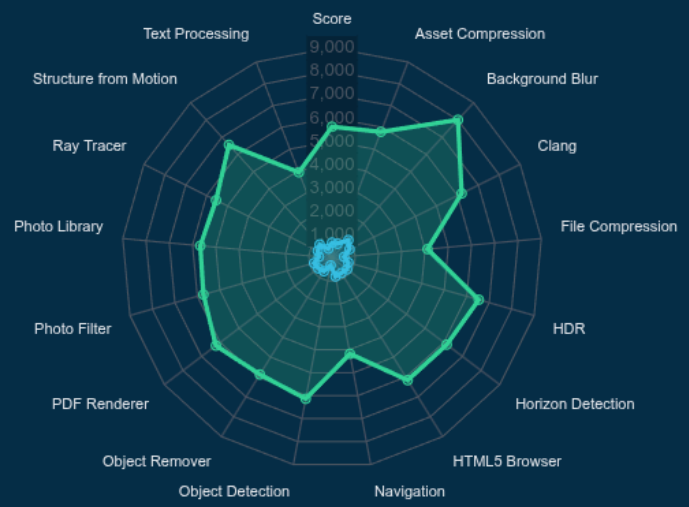
Geekbench

Show Details

Single-core performance



Multi-core performance

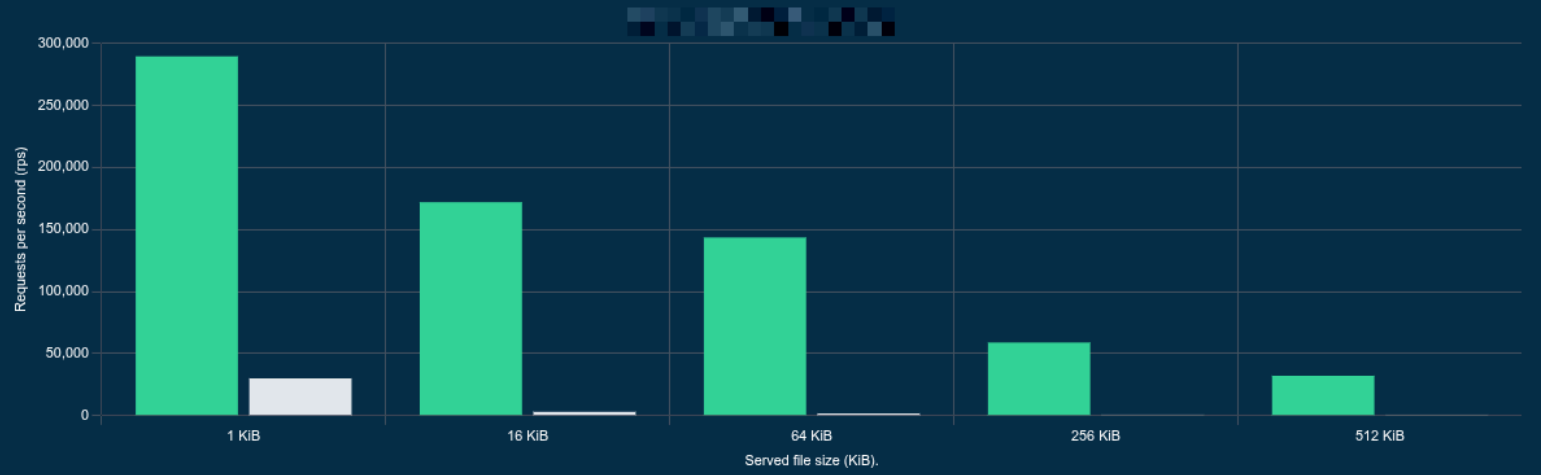


<spare_cores>


OpenSSL
Show Details
sha256
5,000,000,000
4,500,000,000
4,000,000,000
3,500,000,000
3,000,000,000
2,500,000,000
2,000,000,000
1,500,000,000
1,000,000,000
500,000,000
0
byte/s
16
64
256
1024
8192
16384
Block Size (byte)

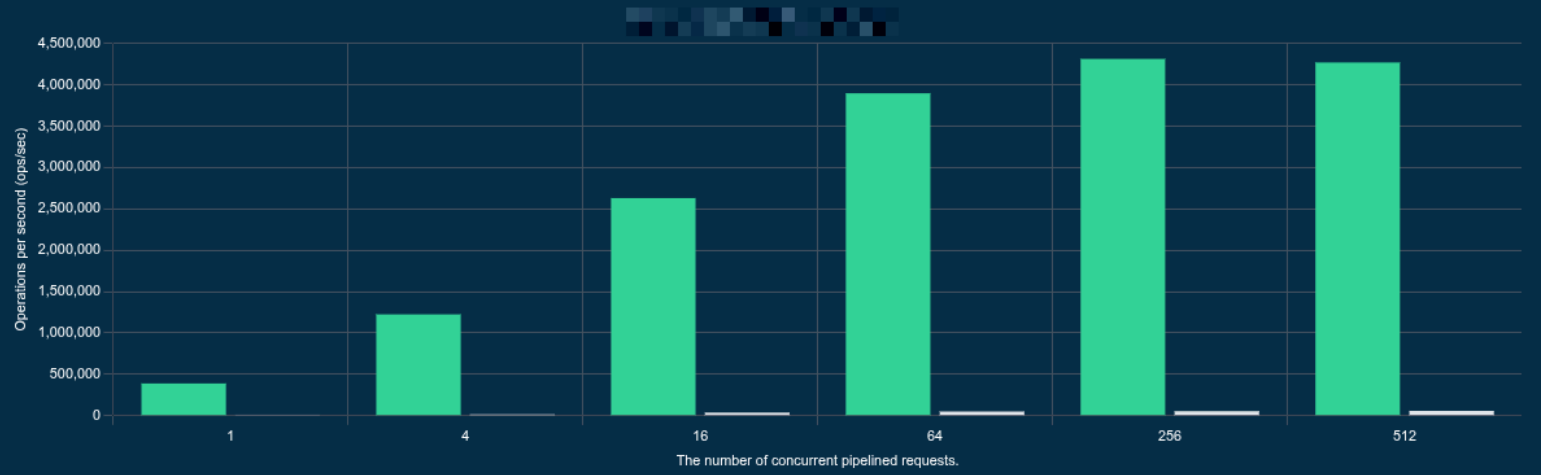
Static Web Server

Show Details



Redis

Show Details





Guess the number of 2 vCPU servers at the supported (8) vendors!



```sql
SELECT
  vendor_id,
  COUNT(*) AS servers
FROM server
WHERE vcpus = 2
GROUP BY 1;
```



$ apropos sparecores.com



$ curl -I github.com/sparecores

Actions Usage Metrics

Period: Last year

Showing data from 4/1/2024 to 41 minutes ago

Total minutes
739,156
Total minutes across all workflows in this organization for last year

Total job runs
95,751
Total job runs across all workflows in this organization for last year

Workflows | Jobs | Repositories | Runtime OS | Runner type

Filter | Search or filter

Workflow	Source repository	Total minutes	Workflow runs	Jobs	Runner type	Runtime OS
crawl-spot.yaml	sc-data	417,011	36,000+	4	hosted	linux
cleanup.yml	sc-inspector-data	109,860	25,000+	3	hosted	linux
start.yml	sc-inspector-data	84,370	16,000+	3	hosted	linux
cleanup-daily.yml	sc-inspector-data	54,004	224	2	hosted	linux
parse.yml	sc-inspector-data	31,176	355	3	hosted	linux
start.yml	sc-inspector-data	11,180	801	1	self-hosted	linux
push.yml	sc-images	9,913	128	15	hosted	linux
parse.yml	sc-inspector-data	3,632	11	1	self-hosted	linux
push.yml	sc-images	2,225	56	11	self-hosted	linux

$ ls -1 sparecores

COMPONENT	SOURCE LICENSE	REFERENCES	DESCRIPTION
Runner	MPL 2.0		Pulumi wrapper to launch instances across clouds.
Images	MPL 2.0		Container images for hardware discovery and benchmark workloads.
Inspector	MPL 2.0		Python tools to inspect and benchmark cloud servers.
Inspector Data	CC BY-SA 4.0		Raw hardware information and benchmark results.
Crawler	MPL 2.0		Cloud vendor API and Inspector Data integrations.
Crawler Data	CC BY-SA 4.0		JSON dumps of collected cloud resource information.
Database	BSL 1.1		SQLite database dump of current cloud resource information.
Data Wrapper	MPL 2.0		Python package for easy integration of the Database.
HTTP API	MPL 2.0		Programmatic access to the Database.
Web UI	MPL 2.0		Web frontend for human access to the Database.

$ curl -I sparecores.com/navigator

```
pgsql=> SELECT
  MIN(updates), AVG(updates), MAX(updates),
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY updates)
    AS median
FROM (
  SELECT
    to_char(observed_at, 'YYYY-MM-DD HH24'),
    COUNT(*) AS updates
  FROM server_price_scd
  GROUP BY 1)

 min  |          avg          | median | max
------+-----------------------+--------+------
 4925 | 5124.0425531914893617 | 5119.5 | 5345
(1 row)
```

- **8** (out of 10 planned) vendors
- **580** availability zones
- **5,950** server types
- **2,629,812** benchmark scores
- **592,330** live price records
- **~5k** records updated hourly
- **100M+** historical records

Tracked every 5 mins.

Source: sparecores.com

<spare_cores>

$ curl sparecores.com/servers



<spare_cores>

$ curl sparecores.com/server/gcp/g2-s…

g2-standard-16 by Google Cloud Platform

g2-standard-16 is a Graphics Optimized: 1 NVIDIA L4 GPU, 16 vCPUs, 64GB RAM server offered by Google Cloud Platform with 16 vCPUs, 64 GiB of memory and 0 GB of storage. The pricing starts at 0.1612 USD per hour.

Compare | Visit Vendor | Status Page

- **16** vCPU
- **64 GiB** Memory
- **1** GPU

Spare SCore

12653
(All-cores)

1195
(Single-core)

Specifications

Server Metadata

Vendor ID	gcp
Server ID	1600016
Name	g2-standard-16
Description	Graphics Optimized: 1 NVIDIA L4 GPU, 16 vCPUs, 64GB RAM
Family	g2
Status	active
Observed At	2026-06-05T11:16:00.639499

Availability

REGION / ID	SPOT	ONDEMAND
Ashburn (US) / us-east4	0.2458 USD/h	0.5852 USD/h
Council Bluffs (US) / us-central1	0.259 USD/h	0.5872 USD/h
The Dalles (US) / us-west1	0.259 USD/h	0.5872 USD/h
Moncks Corner (US) / us-east1	0.259 USD/h	0.5872 USD/h
Mumbai (IN) / asia-south1	0.2566 USD/h	0.6112 USD/h
Eemshaven (NL) / europe-west4	0.3085 USD/h	0.6171 USD/h
St. Ghislain (BE) / europe-west1	0.2335 USD/h	0.6465 USD/h

Show more

$ curl sparecores.com/server/gcp/g2-s…

Processor

vCPUs	16
CPU Allocation	Dedicated
CPU Cores	8
CPU Speed	2.2 GHz
CPU Architecture	x86_64
CPU Manufacturer	Intel
CPU Family	Xeon
CPU L1D Cache	

View more details

System Resources and Accelerators

MEMORY

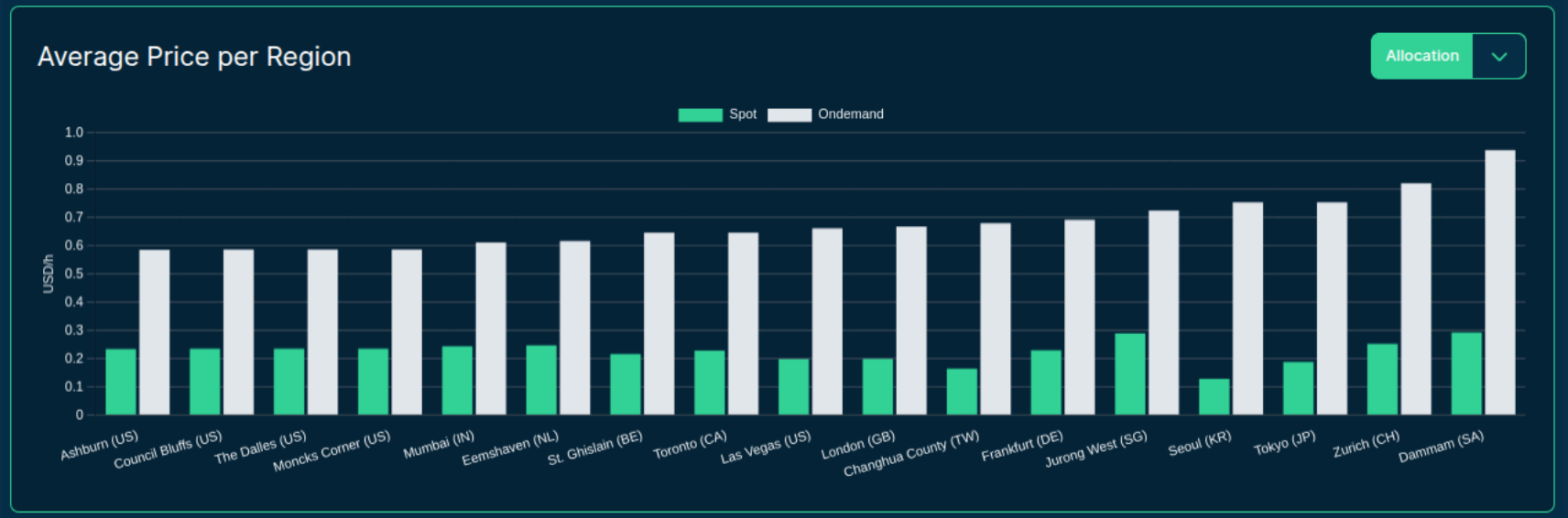
Memory Amount	64 GiB
Memory Amount Actual	64 GiB

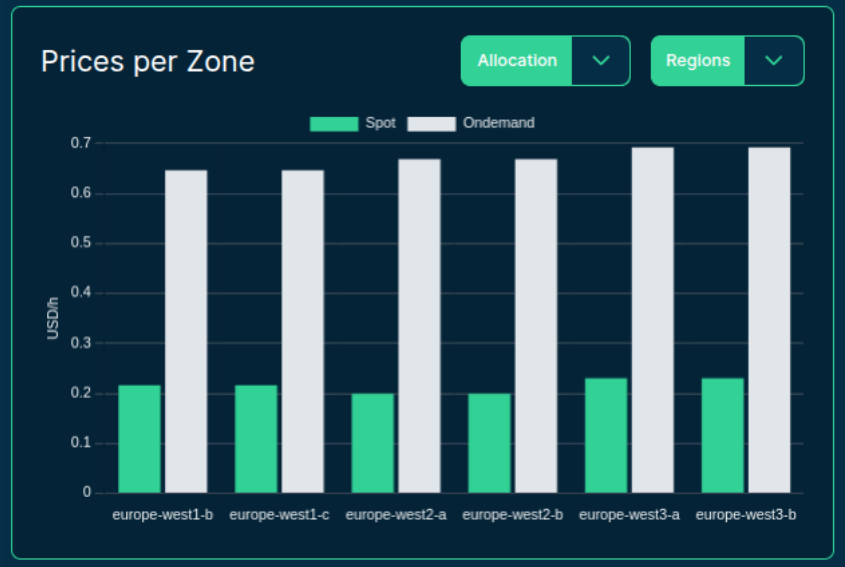
GPU

GPU Count	1
GPU Memory Min	22 GiB
GPU Memory Total	

View more details

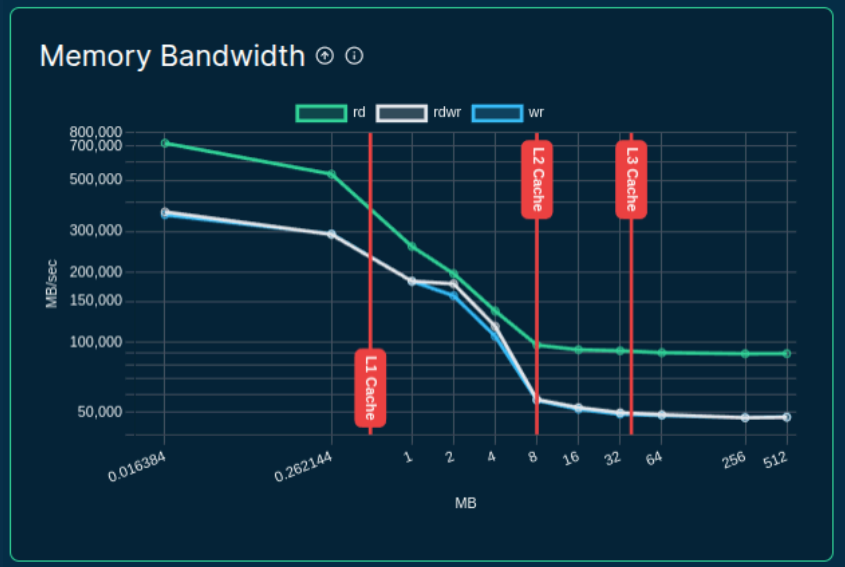
$ curl sparecores.com/server/gcp/g2-s…

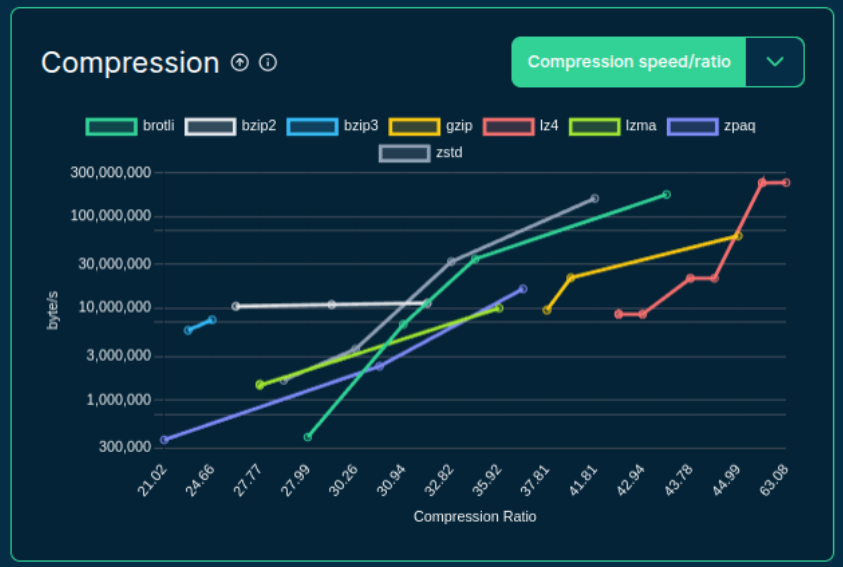


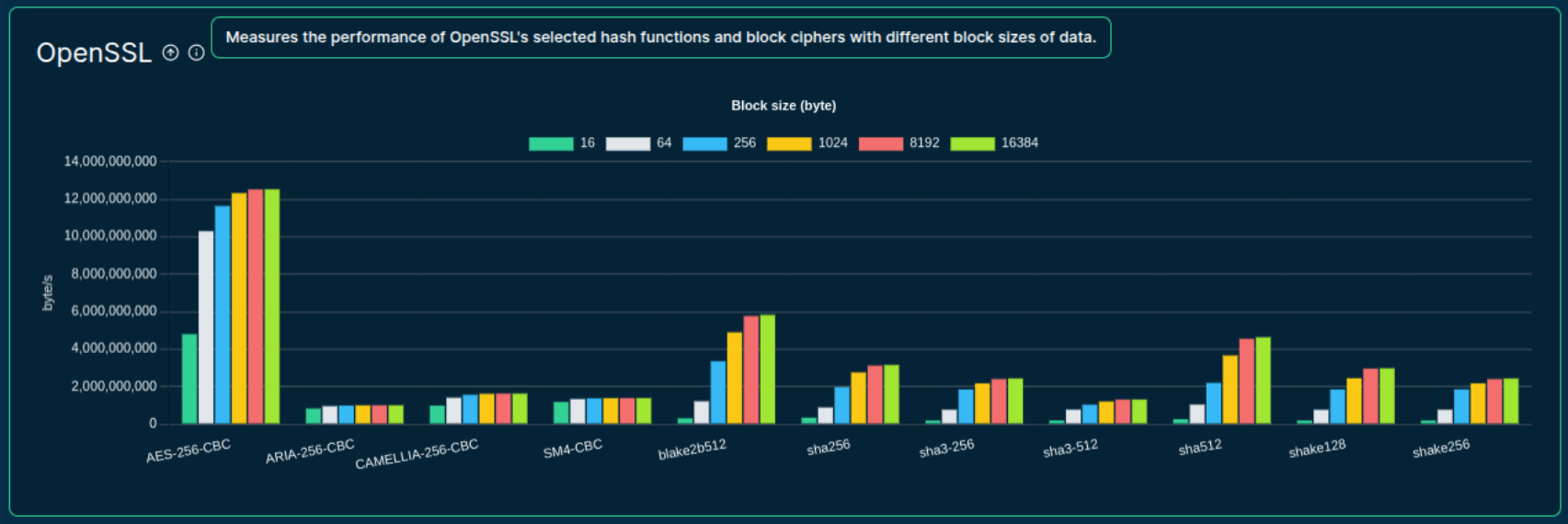




$ curl sparecores.com/server/gcp/g2-s…







$ curl sparecores.com/server/gcp/g2-s…

Passmark CPU Scores

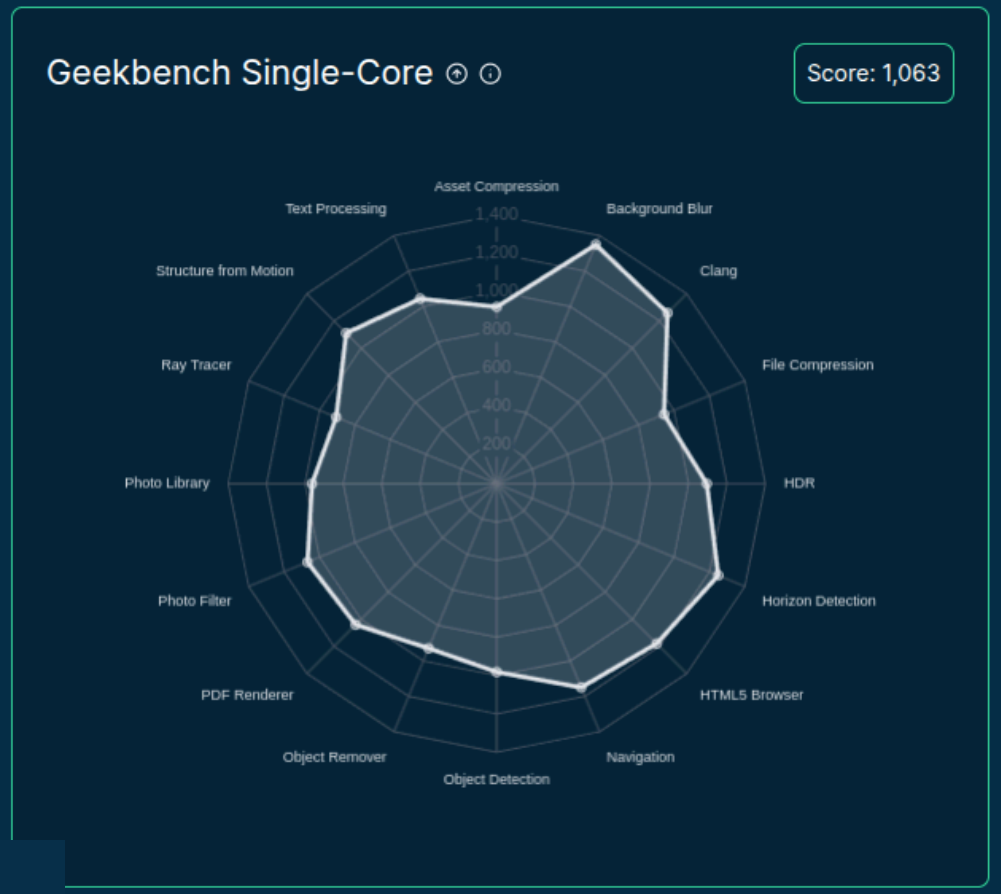
BENCHMARK	SCORE
Compression	187062
Encryption	5810
Extended Instructions	11945
Floating Point Maths	22743
Integer Maths	42240
Mark	13538
Physics	1521
Prime Numbers	76
Single Threaded	1691
String Sorting	22203

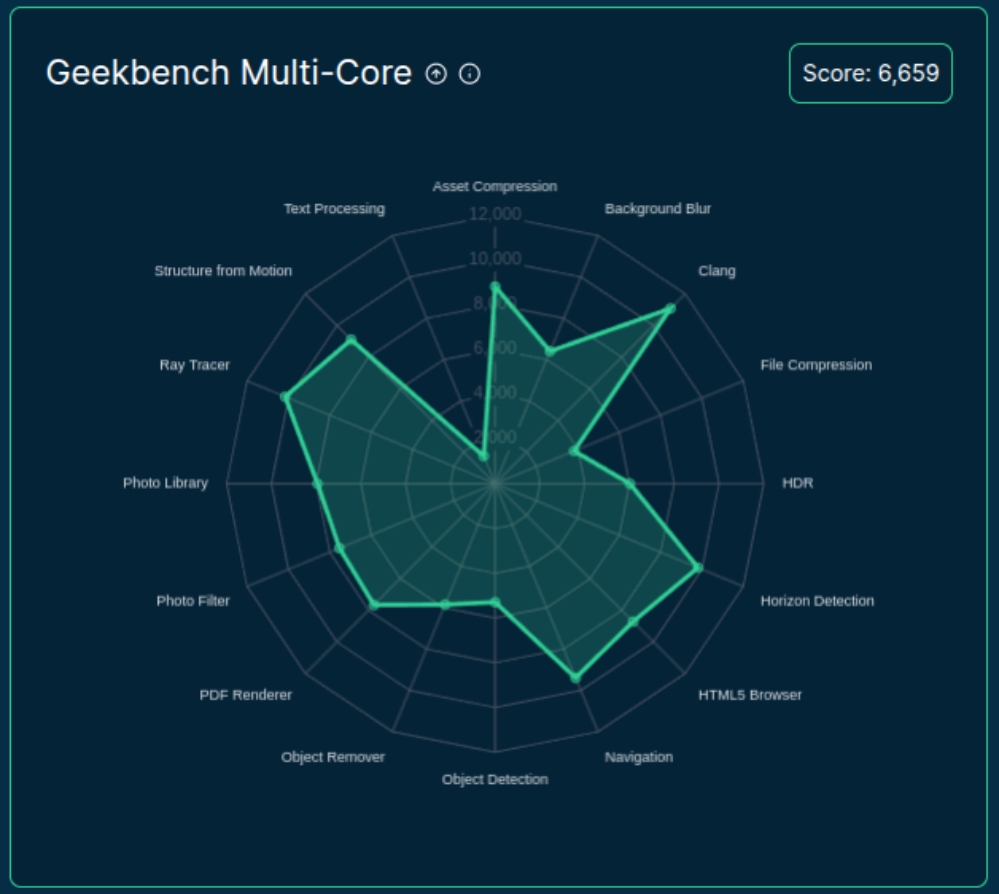
Show less

Passmark Memory Scores

BENCHMARK	SCORE
Database Operations	5263
Memory Latency	52
Memory Mark	2521
Memory Read Cached	20704
Memory Read Uncached	11185
Memory Write	10359

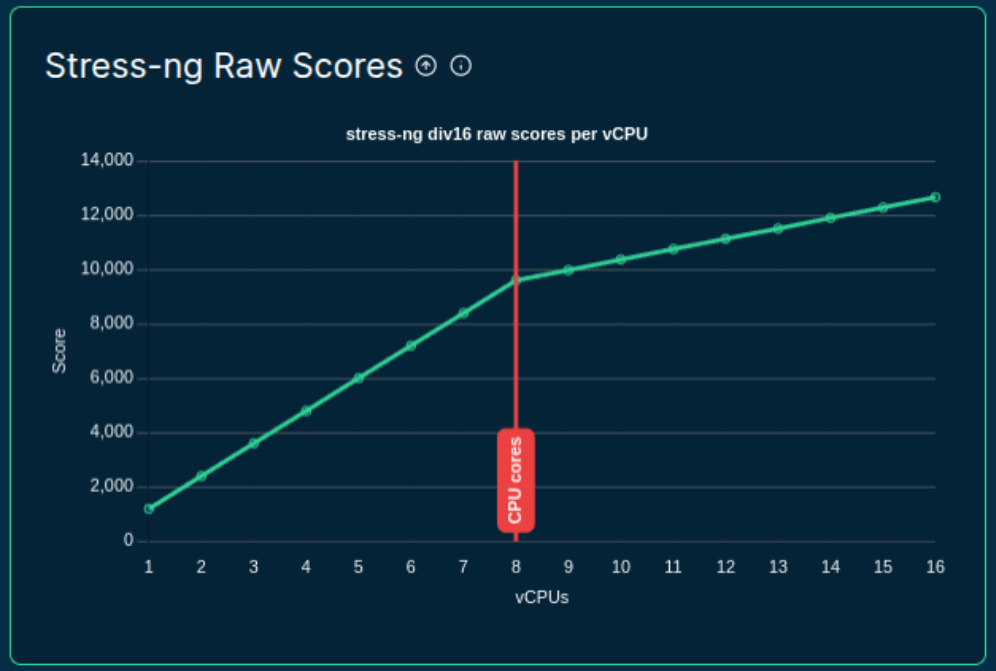
$ curl sparecores.com/server/gcp/g2-s…

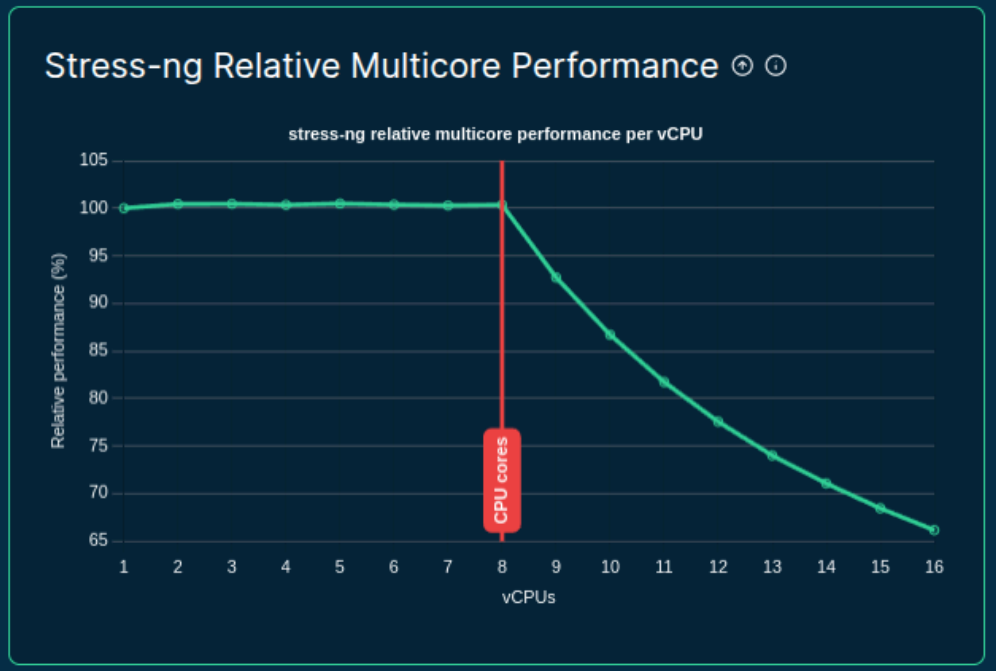




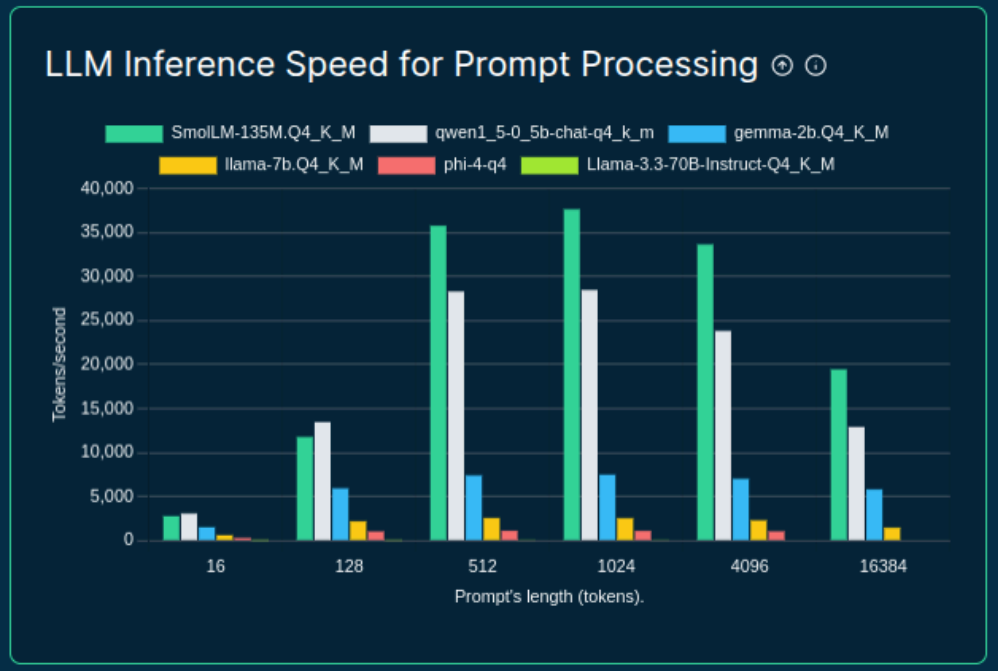
<spare_cores>

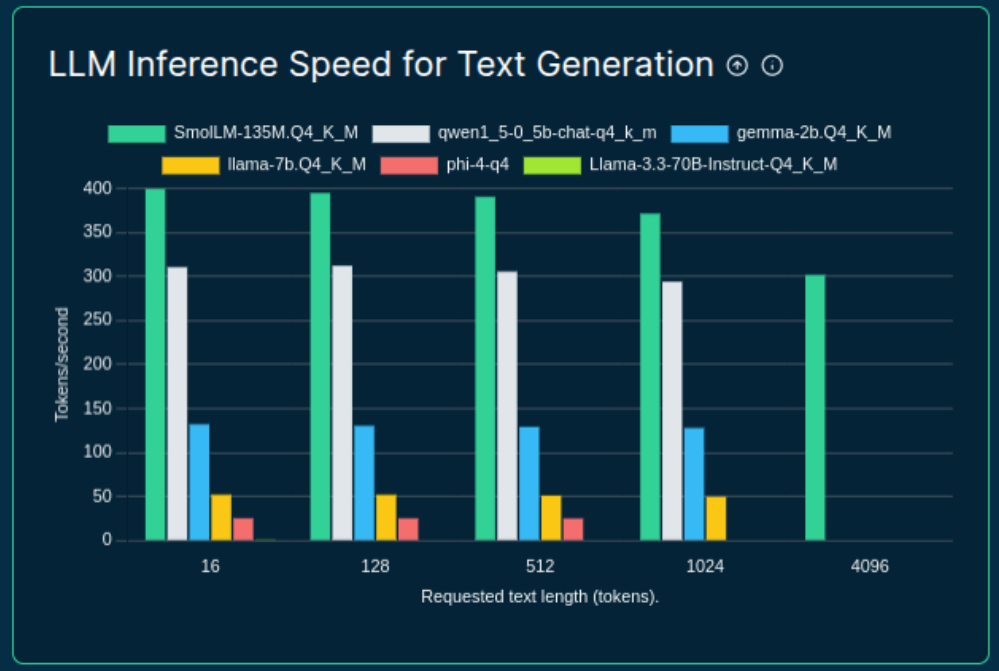
$ curl sparecores.com/server/gcp/g2-s…



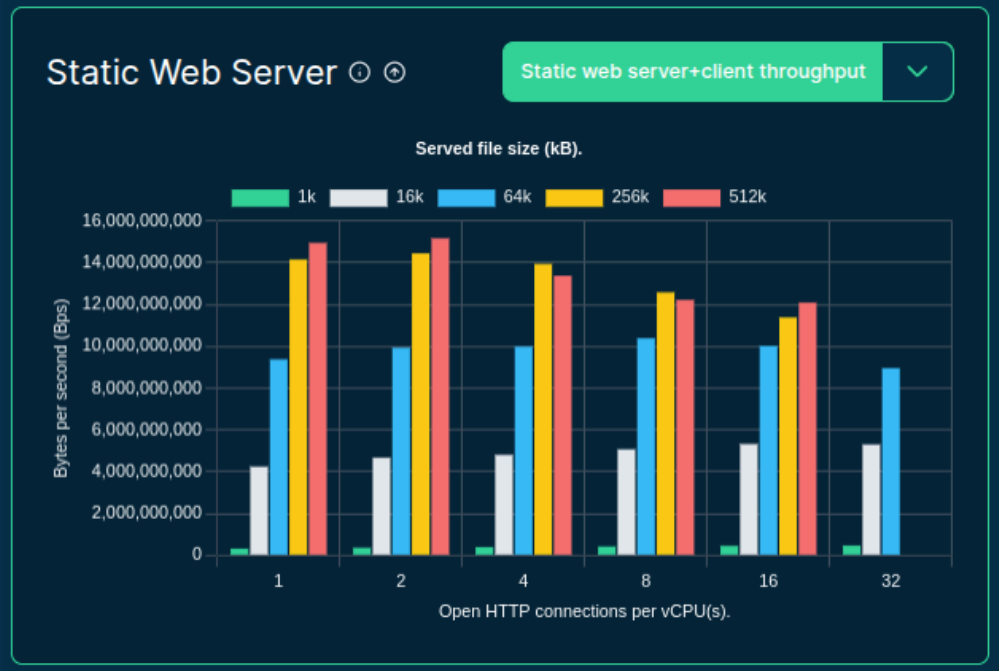


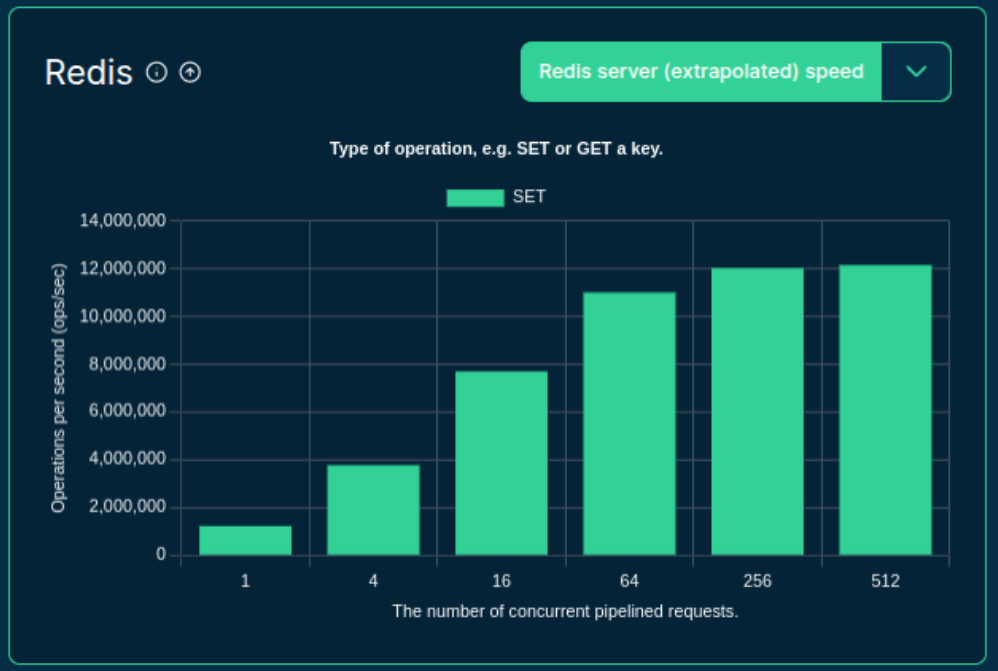
$ curl sparecores.com/server/gcp/g2-s…





>>> from navigator.www import server





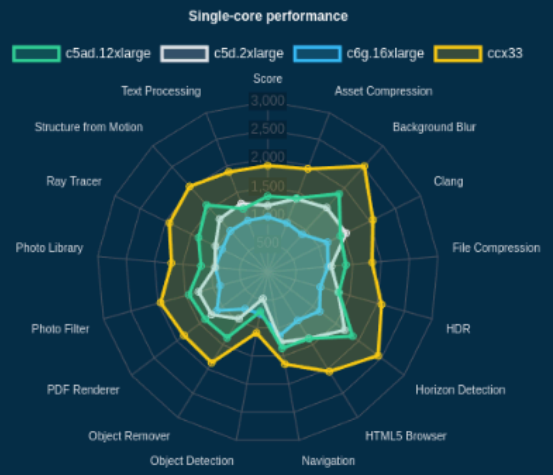
$ curl sparecores.com/servers/compare

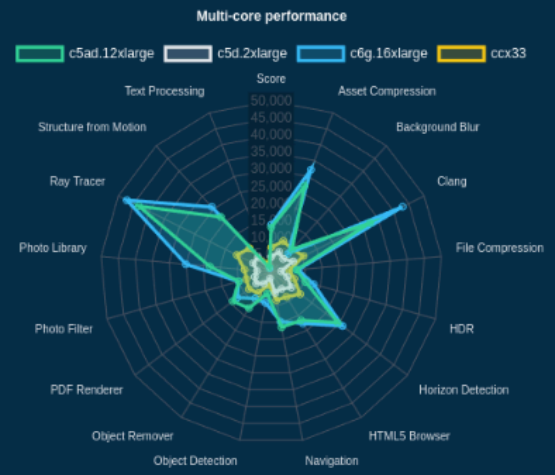
	CCX63 💪	CCX53	CPX51	T2D-STANDARD-16 🏅	M7G.4XLARGE	CAX41 💰	B16PLS_V2
Vendor							
SCore All	73650	49227	12463	43101	24504	29118	28891
SCore Single	3059	3066	772	2695	1532	1822	1823
Best spot price	- -	- -	- -	0.06093 USD us-east5-c	0.0816 USD ap-southeast-3b	- -	0.273603 USD centralindia-1
Best ondemand price	0.5398 USD nbg1-dc3	0.3599 USD nbg1-dc3	0.102 USD nbg1-dc3	0.44613 USD asia-south1-b	0.4666 USD ap-south-1a	0.0449 USD nbg1-dc3	0.317 USD jioindiacentral-0
CPU							
vCPUs	48	32	16	16	16	16	16
Hypervisor	QEMU	QEMU	QEMU		nitro	QEMU	Microsoft Hyper-V
CPU Allocation	Dedicated	Dedicated	Shared	Dedicated	Dedicated	Shared	Dedicated
CPU Cores	24	16	16	16	16	16	16
CPU Speed	2 GHz	2 GHz	2 GHz		2.6 GHz	2 GHz	3 GHz
CPU Architecture	x86_64	x86_64	x86_64	x86_64	arm64	arm64	arm64
CPU Manufacturer	AMD	AMD	AMD	AMD	AWS	AMD	Ampere
CPU Family	EPYC	EPYC	EPYC	EPYC	ARMv8	Ampere Altra	ARMv8
CPU Model				7B13	Neoverse-V1	Neoverse-N1	Neoverse-N1
CPU L1 Cache	2 MiB	1 MiB	1 MiB	1 MiB	2 MiB		2 MiB
CPU L2 Cache	12 MiB	8 MiB	8 MiB	8 MiB	16 MiB		16 MiB
CPU L3 Cache	32 MiB	32 MiB	16 MiB	64 MiB	32 MiB		32 MiB
Memory							
Memory Amount	192 GiB	128 GiB	32 GiB	64 GiB	64 GiB	32 GiB	32 GiB

<spare_cores>

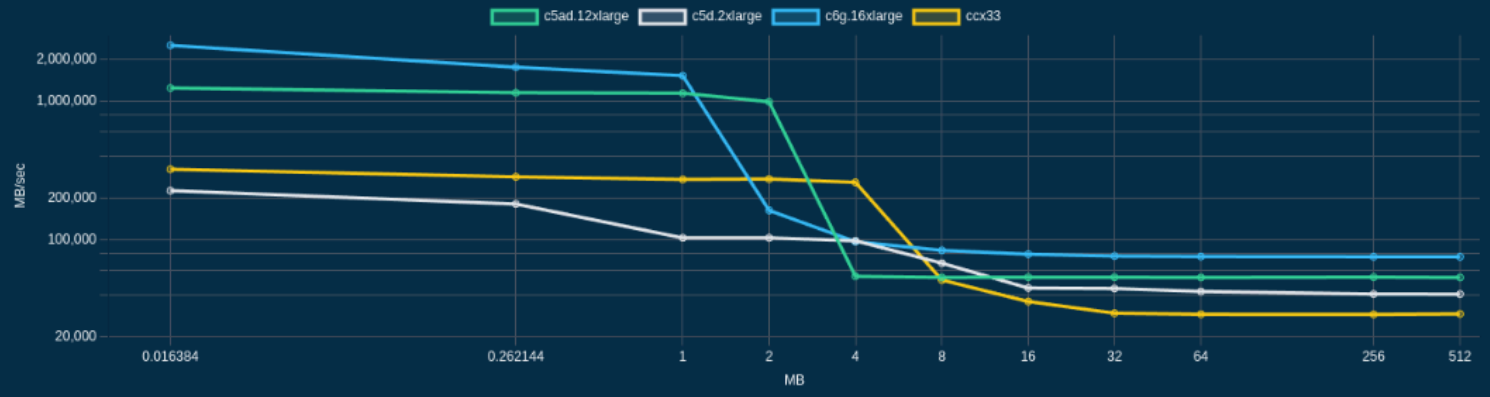
$ curl sparecores.com/servers/compare

Geekbench



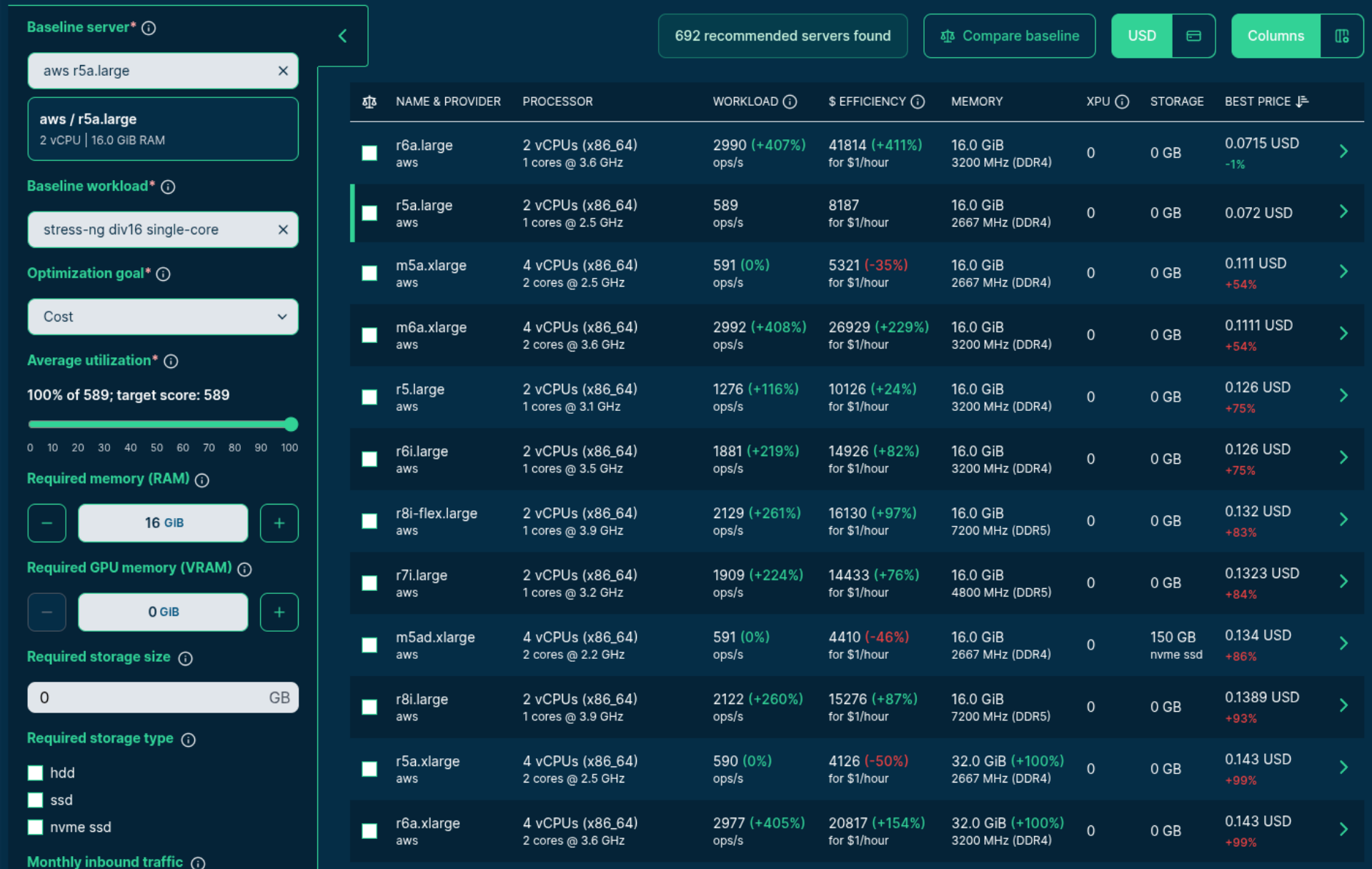


Memory bandwidth



<spare_cores>

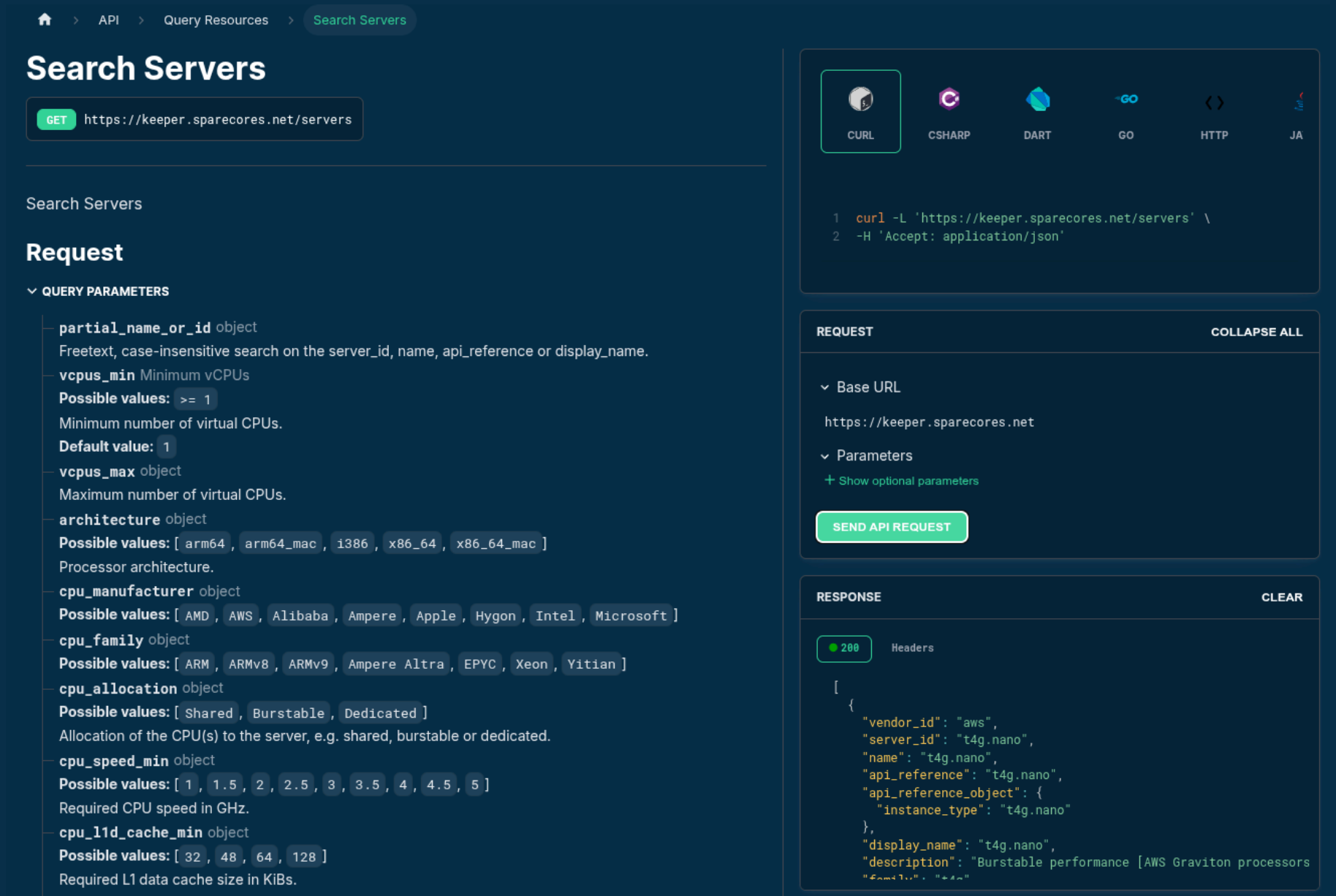
$ curl sparecores.com/advisor



NAME & PROVIDER	PROCESSOR	WORKLOAD	$ EFFICIENCY	MEMORY	XPU	STORAGE	BEST PRICE
r6a.large aws	2 vCPUs (x86_64) 1 cores @ 3.6 GHz	2990 (+407%) ops/s	41814 (+411%) for $1/hour	16.0 GiB 3200 MHz (DDR4)	0	0 GB	0.0715 USD -1%
r5a.large aws	2 vCPUs (x86_64) 1 cores @ 2.5 GHz	589 ops/s	8187 for $1/hour	16.0 GiB 2667 MHz (DDR4)	0	0 GB	0.072 USD
m5a.xlarge aws	4 vCPUs (x86_64) 2 cores @ 2.5 GHz	591 (0%) ops/s	5321 (-35%) for $1/hour	16.0 GiB 2667 MHz (DDR4)	0	0 GB	0.111 USD +54%
m6a.xlarge aws	4 vCPUs (x86_64) 2 cores @ 3.6 GHz	2992 (+408%) ops/s	26929 (+229%) for $1/hour	16.0 GiB 3200 MHz (DDR4)	0	0 GB	0.1111 USD +54%
r5.large aws	2 vCPUs (x86_64) 1 cores @ 3.1 GHz	1276 (+116%) ops/s	10126 (+24%) for $1/hour	16.0 GiB 3200 MHz (DDR4)	0	0 GB	0.126 USD +75%
r6i.large aws	2 vCPUs (x86_64) 1 cores @ 3.5 GHz	1881 (+219%) ops/s	14926 (+82%) for $1/hour	16.0 GiB 3200 MHz (DDR4)	0	0 GB	0.126 USD +75%
r8i-flex.large aws	2 vCPUs (x86_64) 1 cores @ 3.9 GHz	2129 (+261%) ops/s	16130 (+97%) for $1/hour	16.0 GiB 7200 MHz (DDR5)	0	0 GB	0.132 USD +83%
r7i.large aws	2 vCPUs (x86_64) 1 cores @ 3.2 GHz	1909 (+224%) ops/s	14433 (+76%) for $1/hour	16.0 GiB 4800 MHz (DDR5)	0	0 GB	0.1323 USD +84%
m5ad.xlarge aws	4 vCPUs (x86_64) 2 cores @ 2.2 GHz	591 (0%) ops/s	4410 (-46%) for $1/hour	16.0 GiB 2667 MHz (DDR4)	0	150 GB nvme ssd	0.134 USD +86%
r8i.large aws	2 vCPUs (x86_64) 1 cores @ 3.9 GHz	2122 (+260%) ops/s	15276 (+87%) for $1/hour	16.0 GiB 7200 MHz (DDR5)	0	0 GB	0.1389 USD +93%
r5a.xlarge aws	4 vCPUs (x86_64) 2 cores @ 2.5 GHz	590 (0%) ops/s	4126 (-50%) for $1/hour	32.0 GiB (+100%) 2667 MHz (DDR4)	0	0 GB	0.143 USD +99%
r6a.xlarge aws	4 vCPUs (x86_64) 2 cores @ 3.6 GHz	2977 (+405%) ops/s	20817 (+154%) for $1/hour	32.0 GiB (+100%) 3200 MHz (DDR4)	0	0 GB	0.143 USD +99%

<spare_cores>

$ curl docs.sparecores.com/api



API > Query Resources > Search Servers

Search Servers

GET https://keeper.sparecores.net/servers

Search Servers

Request

QUERY PARAMETERS

- **partial_name_or_id** object
  Freetext, case-insensitive search on the server_id, name, api_reference or display_name.
- **vcpus_min** Minimum vCPUs
  Possible values: >= 1
  Minimum number of virtual CPUs.
  Default value: 1
- **vcpus_max** object
  Maximum number of virtual CPUs.
- **architecture** object
  Possible values: [arm64 , arm64_mac , i386 , x86_64 , x86_64_mac]
  Processor architecture.
- **cpu_manufacturer** object
  Possible values: [AMD , AWS , Alibaba , Ampere , Apple , Hygon , Intel , Microsoft]
- **cpu_family** object
  Possible values: [ARM , ARMv8 , ARMv9 , Ampere Altra , EPYC , Xeon , Yitian]
- **cpu_allocation** object
  Possible values: [Shared , Burstable , Dedicated]
  Allocation of the CPU(s) to the server, e.g. shared, burstable or dedicated.
- **cpu_speed_min** object
  Possible values: [1 , 1.5 , 2 , 2.5 , 3 , 3.5 , 4 , 4.5 , 5]
  Required CPU speed in GHz.
- **cpu_l1d_cache_min** object
  Possible values: [32 , 48 , 64 , 128]
  Required L1 data cache size in KiBs.

CURL CSHARP DART GO HTTP JA

```
curl -L 'https://keeper.sparecores.net/servers' \
-H 'Accept: application/json'
```

REQUEST COLLAPSE ALL

Base URL

https://keeper.sparecores.net

Parameters

+ Show optional parameters

SEND API REQUEST

RESPONSE CLEAR

200 Headers


```
[
  {
    "vendor_id": "aws",
    "server_id": "t4g.nano",
    "name": "t4g.nano",
    "api_reference": "t4g.nano",
    "api_reference_object": {
      "instance_type": "t4g.nano"
    },
    "display_name": "t4g.nano",
    "description": "Burstable performance [AWS Graviton processors
```


<spare_cores>


One Core.
No Limits.
GILty
Python
The GIL
Concurrency?
Not today!
More cores?
Later...
GILty
1
CORE


CORE ALONE
GILty

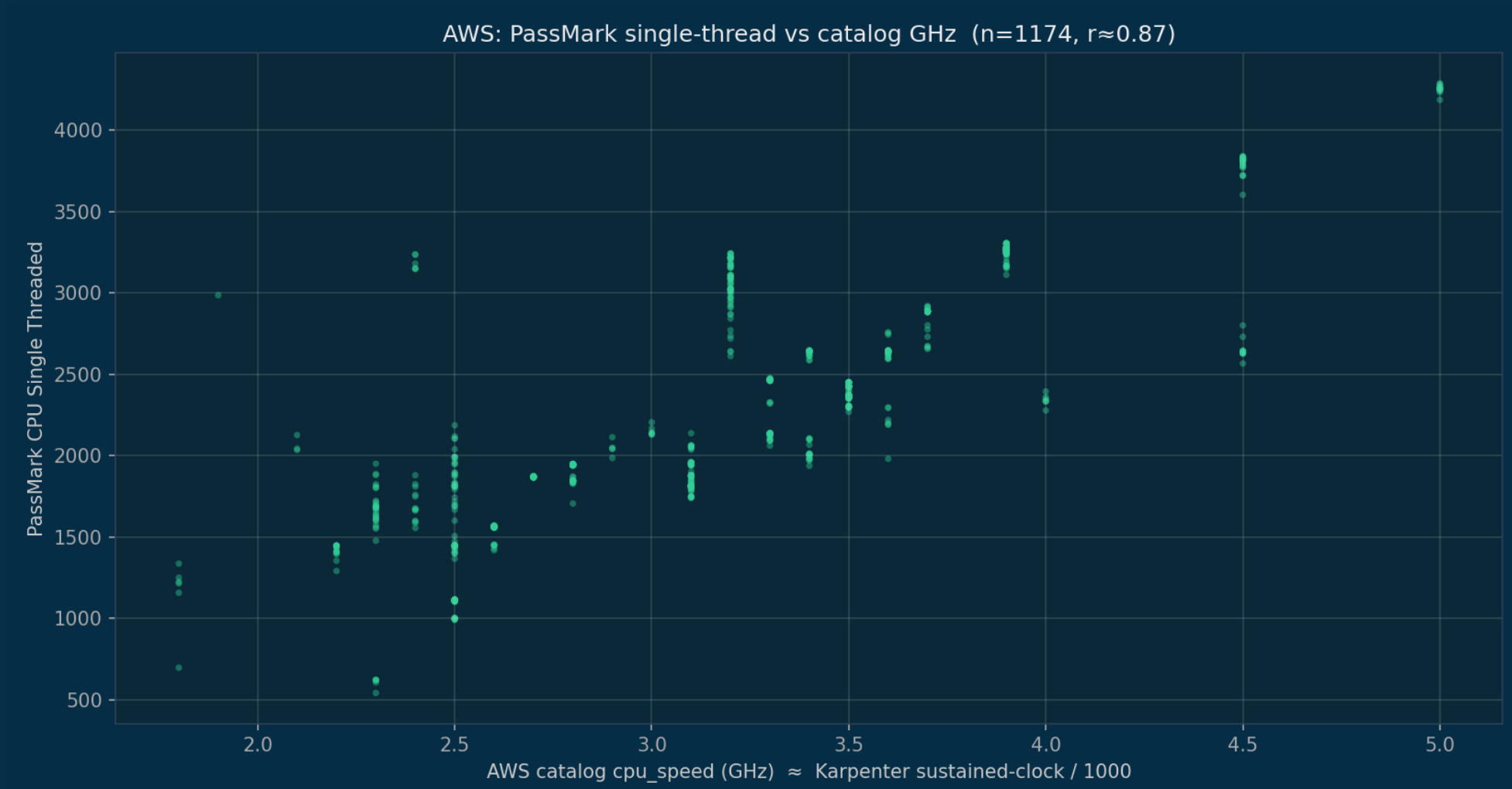
$ strace -e environment gilty.py

Before: AWS · EKS · Cluster Autoscaler + fixed node groups
→ capacity, not **single-core** / **latency**



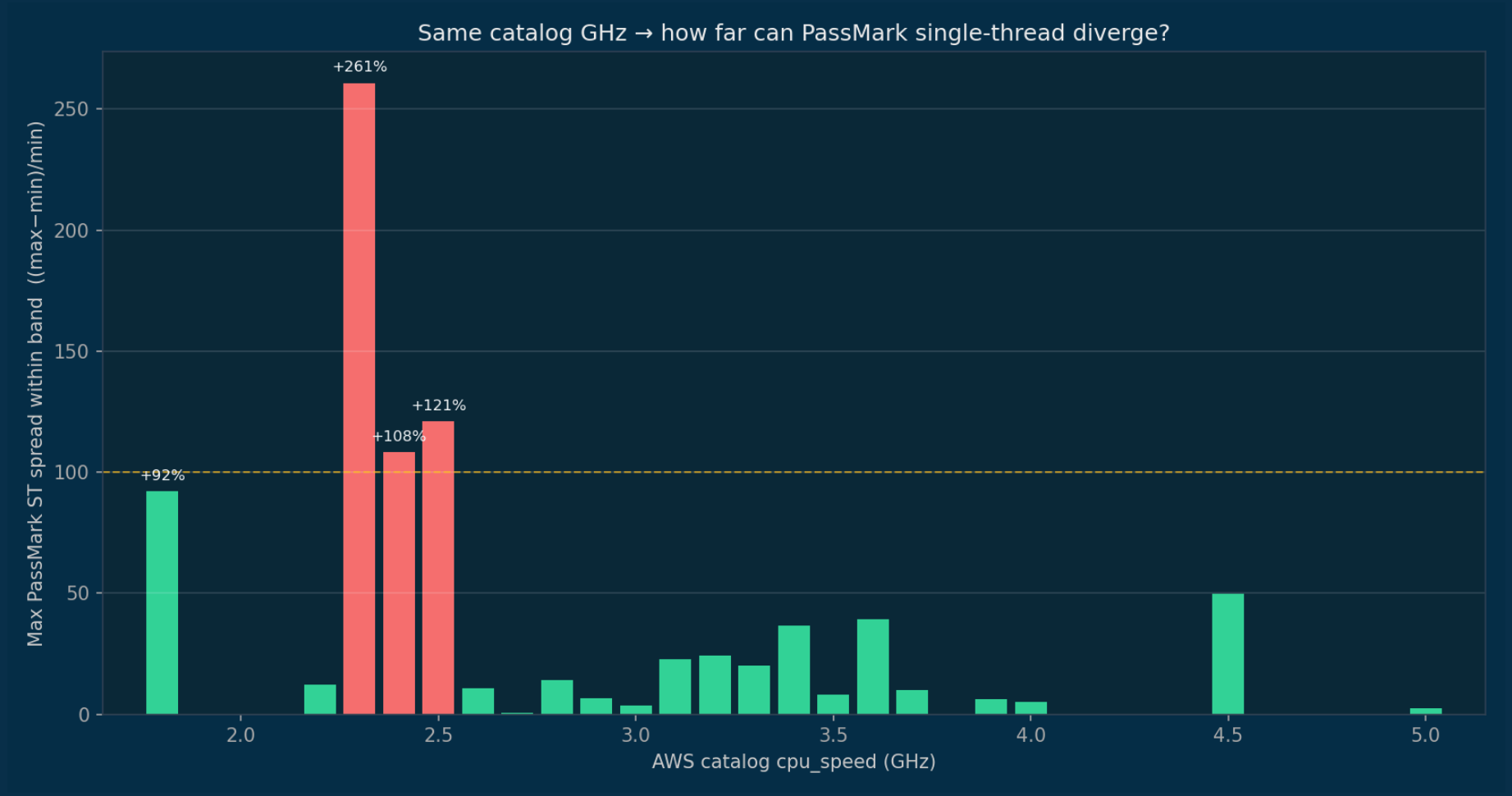
<spare_cores>

PassMark ST vs catalog GHz



<spare_cores>

Same GHz, how wide is PassMark?



<spare_cores>

Within one clock band: up to about **+200%** PassMark ST spread

Just set the filter to 5 GHz?

Simple idea: `instance-cpu-sustained-clock-speed-mhz >= 5000`

	r5a.large	c8a.large	m8azn.large
catalog GHz	2.5	4.5	**5.0**
PassMark ST	~1400	~3800	~4300
on-demand	~$0.11	~$0.11	~$0.21
regions	~16	~7	**~6**

- only **~16** types · **2** families · **~1%** of AWS catalog
- **~2×** the money for ~**+10%** PassMark vs **c8a**
- tiny fallback pool → **capacity / spot** pain

$ claude –plan “fix problem”

Pub talk → need **benchmark**-ranked types in **Karpenter** (incl. **EKS Auto**)

Plan (`sc-karpenter-nodepool` ADRs + sequenced v1 plans):

1. **Own NodePools only** — never patch user pools / no `NodeOverlay`
   → **managed** Karpenter **+ GitOps** (Argo CD, …) — no fights over a shared object
2. `SCNodePool` → **one NodePool** — template verbatim + selected `instance-type` list
3. Emit **universal keys only** (`instance-type` · arch · capacity)
   → avoid **RequirementsDrifted** loops
4. **Selectors + ranking** from **Keeper** (e.g. PassMark ST · family band)

Churn control — 24h refresh · hysteresis · never empty list

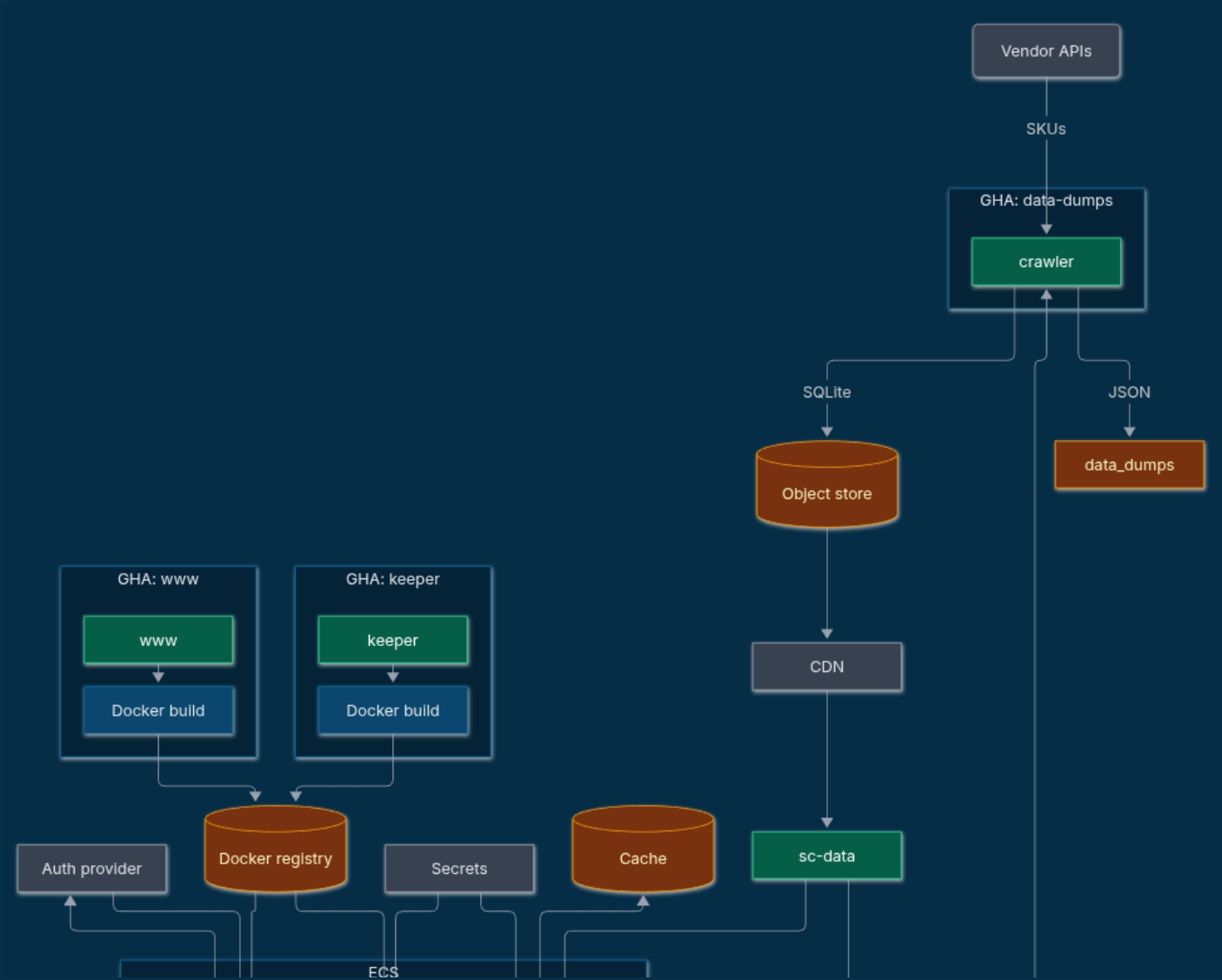
Go / kubebuilder — reuse Karpenter validation + provider test fakes

```
$ kubectl get nodes
NAME                  STATUS   ROLES    AGE    VERSION
i-0af7332289af07856   Ready    <none>   2m5s   v1.35.6-eks-bca9cf6
i-0c8cd5e7f8bbcd997   Ready    <none>   48s    v1.35.6-eks-bca9cf6
$ kubectl get nodepools,nod
```



sc-karpenter-nodepool demo

$ strace -e environment sparecores.com



<spare_cores>

ENTRYPOINT [“sc-karpenter-nodepool”]

Great for **picking** types · **not** for inspector-style **multi-cloud JIT**

Limitations (vs “pod annotation → any cloud / region / AZ / SKU”):

$ cat sc-karpenter-nodepool/ROADMAP.md

Ideal stack for **sc-inspector** (replace GH Actions + Pulumi `start.yml`):



> /review plan



```
$ ping sparecores.com
```

