2022 Cloud Report

Authors:
Keith McClellan
Lidor Carmel
Charlie Custer
Yevgeniy Miretskiy
Stan Rosenberg
Jane Xing

Cockroach Labs

Introduction

The 2022 Cloud Report evaluates the performance of Amazon Web Services (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP) for common OLTP workloads.

Using the product we know best – CockroachDB – we ran an industry-standard-based database benchmark configured consistently across runs and instance types, as well as leading micro-benchmarks in the areas of compute, networking, and storage, to compare the performance and value of 56 instance types and 107 discrete configurations across the three clouds. In this report, we aim to provide an unbiased picture of the performance users are paying for when they provision a specific configuration in one of the clouds.

2022 testing methodology

We tested multiple node sizes, performing an 8 vCPU test and a roughly 32 vCPU test for each instance type. Because not all the large instances are identical (large nodes varied from 30 to 36 vCPUs), we decided to focus on per-vCPU metrics for the majority of our analysis. This allows us to compare instances that are similar but not quite the same, and to compare different node sizes for relative performance and cost effectiveness.

We expanded and improved our OLTP benchmarking this year, running over 3,000 different OLTP iterations leveraging CockroachDB. We disabled wait times in the benchmark because it enabled us to narrow the variation in results across runs. This change has allowed us to achieve a margin of error below 1.5%, but it does mean that the results in this report aren't directly comparable to results from previous years.

We enhanced our micro-benchmarking suites, particularly the networking benchmark, which now includes cross-region tests. CockroachDB is a multi-region, multi-active database and we felt that network performance, including intra-AZ and cross-region performance, was likely to have a direct impact on customers' experiences running applications like ours in production.

General methodology

For this year's Cloud Report, we focused on a question we get a lot when we are working with CockroachDB users: **is it better to use more, smaller nodes, or fewer large ones? This was our primary motivation for testing multiple node sizes and focusing on per-vCPU metrics.**

For instance types that present multiple CPU generations, we explicitly requested the newest available CPU platforms whenever possible. In cases where that was not a configuration option, we ran some additional iterations so we had sufficient runs using the best available processors.

In a few cases, we ended up doing some additional testing outside our preferred region to round out the comparison due to availability so as not to overly skew the results. All instance types tested were in GA (general availability) and should be available for use by the general public as of the time of writing.

We chose not to test ARM instance types this year as CockroachDB still does not provide official binaries for that processor platform. Official support for ARM binaries is slated for our Fall release (22.2), so we expect to return to testing this processor platform next year.

Since each of the cloud providers is now offering at least two performant persistent storage options that meet CockroachDB's requirements, we chose not to test large direct-attached volumes like we did in previous years. However, we enhanced the storage benchmark to better reflect the way CockroachDB interacts with storage to determine if there was a relative performance or latency impact to picking one storage type over another.

For the network benchmarks, we picked the Northern Virginia and Pacific Northwest regions for each cloud provider except where otherwise noted.

All of our pricing calculations were based on the publicly available pricing for the Northern Virginia region for each of the cloud providers.

All testing was conducted on instances running the latest LTS version of Ubuntu at the time we started testing, which was 20.04 LTS (Focal Fossa).

For more details about the methodology used for each benchmark, please refer to the "Methodology" section that is included with each of four detailed benchmarks:

OLTP Methodology
CPU Methodology
Network Methodology
Storage Methodology

Open source testing methodology and reproduction steps

In the 2022 Cloud Report, we benchmarked 56 instance types and 107 discrete configurations across the following broad axes:

- OLTP Performance
- CPU Performance
- Network Performance
- Storage I/O Performance

Machine selection was finalized on November 24, 2021 and tests were run over a variety of dates from November 24, 2021 through March 14, 2022.

The full list of all instance types tested is available in Appendix I.

Reproduction steps are fully open source, available in this repo: github.com/cockroachlabs/cloud-report

Additional benchmark-specific methodology and configuration details are available in the "Methodology" sections included with each of four detailed benchmarks.

Cloud instance naming conventions

AWS

Of all the three clouds, AWS has the most machine configurations. Instead of specifying a flag during provisioning to get better network performance or locally-attached disks, AWS users must choose a different machine type. AWS provides a uniform pattern of naming across all instances.

a variants: denotes an AMD processor

b variants: denotes instances optimized for EBS storage

n variants: suggests better network performance

Azure

Similar to the other clouds, Azure has its own unique naming scheme. The VM series is referenced by the capital letter preceding the vCPU count.

a suffix: denotes an AMD processor

s suffix: designates the ability to attach a premium managed disk, such as premium-disk or ultra-disk

v4 suffix: previous-generation nodes, i.e. Intel Cascade Lake or AMD Rome processors.

v5 suffix: current-generation nodes, i.e. Intel Ice Lake or AMD Milan processors.

GCP

In contrast to AWS, GCP has very few named machines but they can be configured appropriately to serve user needs. For example, GCP is the only cloud that does not market a "storage-optimized machine," but it is also the only cloud that allows users to configure the number of local SSDs.

d suffix: denotes an AMD processor (for example, the n2d machine is the AMD counterpart to the n2 machine)

highcpu/standard/highmem nomenclature: dictates the vCPU to memory (RAM) ratio — 1:1, 1:4, 1:8 respectively.

For our purposes, **custom** refers to a vCPU to RAM ratio of 1:2

Cockroach Labs

2022 Cloud Report

Insights

AMD beats Intel on overall performance

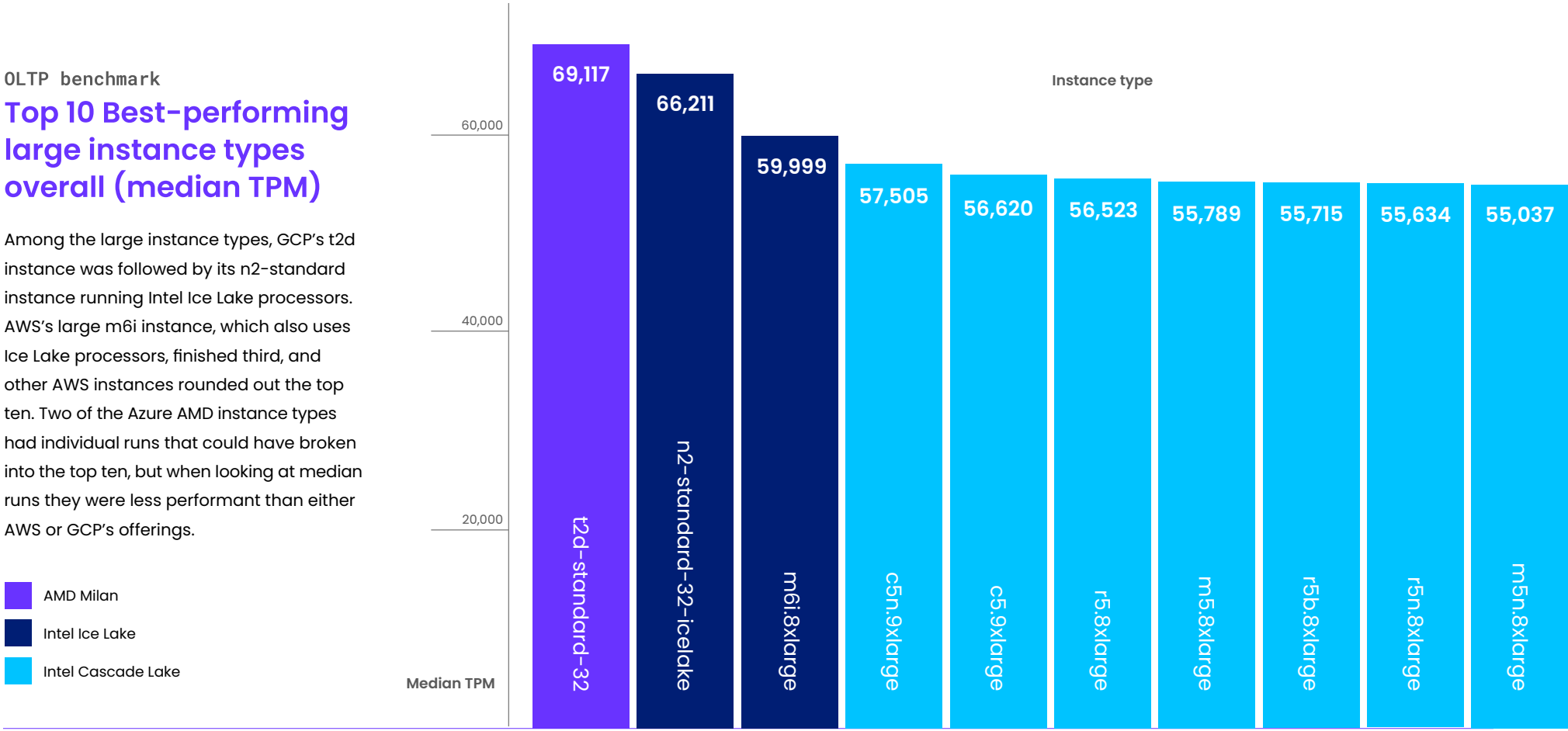
This year's report features brand-new instance types featuring both Intel Ice Lake processors and AMD Milan (3rd Generation EPYC) processors. While both performed well, GCP's t2d instance type (which uses AMD Milan processors) took the top overall spot for both large and small nodes in our OLTP testing.

Across both the OLTP and CPU benchmarks, AMD's Milan processors outperformed Intel in many cases, and at worst statistically tied with Intel's latest-gen Ice Lake processors. In past years, we saw Intel lead the pack in overall performance, with AMD competing on price-for-performance metrics. This year, both the overall performance leader and the price-for-performance leader were AMD-based instances.

OLTP benchmark

Top 10 Best-performing large instance types overall (median TPM)

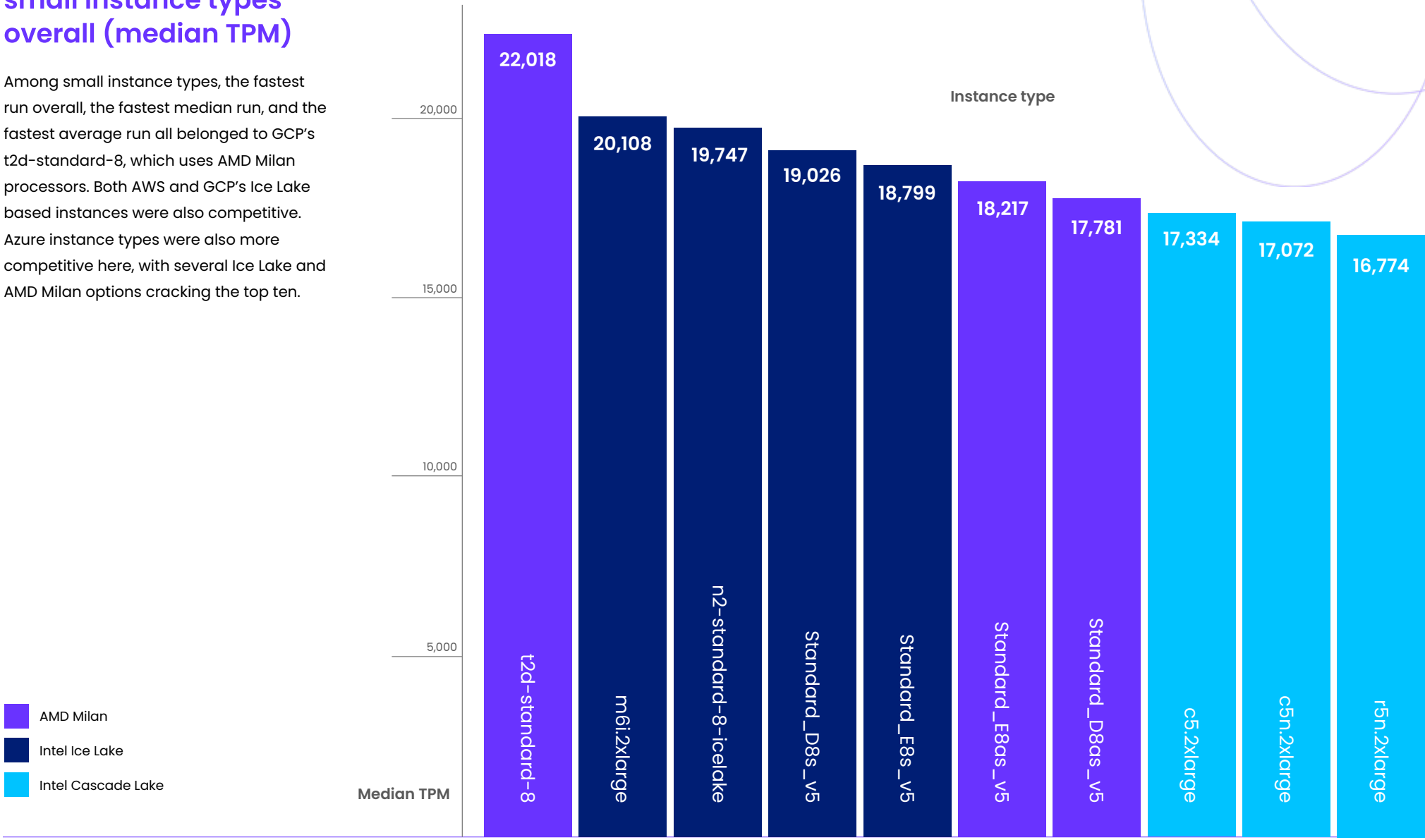
Among the large instance types, GCP's t2d instance was followed by its n2-standard instance running Intel Ice Lake processors. AWS's large m6i instance, which also uses Ice Lake processors, finished third, and other AWS instances rounded out the top ten. Two of the Azure AMD instance types had individual runs that could have broken into the top ten, but when looking at median runs they were less performant than either AWS or GCP's offerings.



OLTP benchmark

Top 10 Best-performing small instance types overall (median TPM)

Among small instance types, the fastest run overall, the fastest median run, and the fastest average run all belonged to GCP's t2d-standard-8, which uses AMD Milan processors. Both AWS and GCP's Ice Lake based instances were also competitive. Azure instance types were also more competitive here, with several Ice Lake and AMD Milan options cracking the top ten.

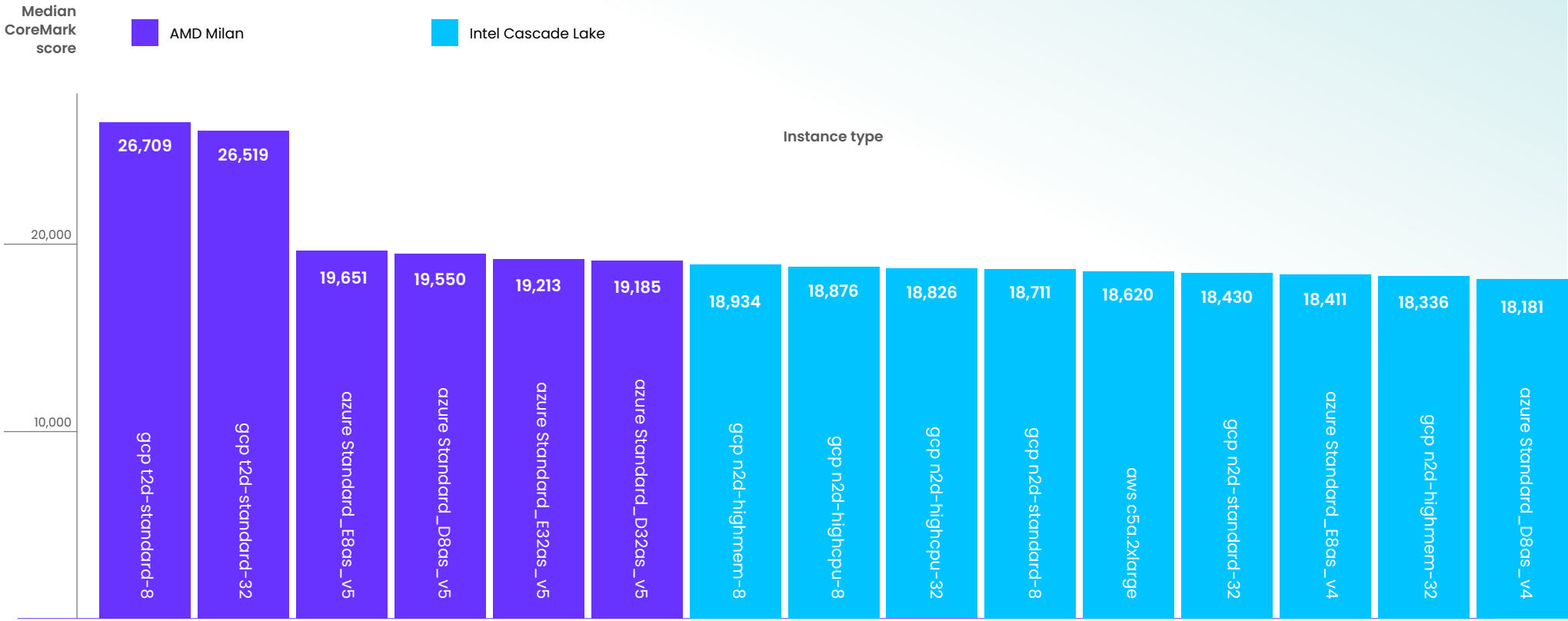


CPU benchmark

Top 15 CoreMark scores by instance type

The power of the new AMD Milan processors was even more apparent in the multi-core CPU microbenchmark. Instances running AMD Milan outperformed instances running both Intel Ice Lake and Intel Cascade Lake processors. One big surprise here is that instances running Intel Ice Lake processors performed worse than older Intel Cascade Lake processors across all three clouds. This is contrary to the results we saw in the OLTP benchmark.

Note: Because of our machine selection and testing cutoff times, we were unable to test AWS's m6a instances, which also run AMD's Milan processors. Based on the rest of our testing, we expect that the m6a instances could have outperformed m6i.

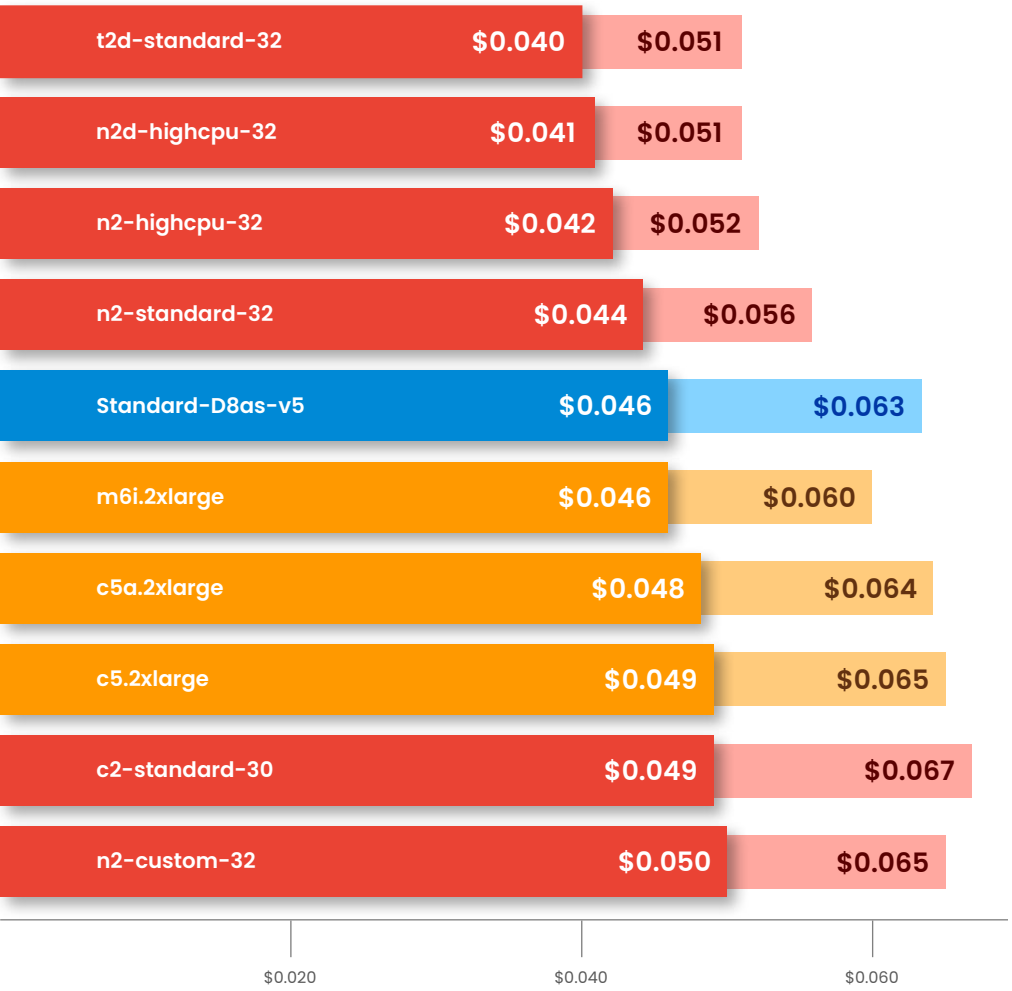


All three clouds have price-competitive offerings

Regardless of the cloud you choose, all three providers offer competitive price-for-performance solutions.[1]

Using our OLTP benchmark as the baseline, all three providers had at least one instance type and storage combination in the $0.04-$0.05 reserved $/TPM range (all prices are per month, reserved assumes a one-year commitment). Even instance and storage combinations that are a bit more expensive than that are potentially very competitive depending on the requirements of a specific workload. Note that the winning instance type, GCP's t2d-standard-32, is not available in all GCP regions as of this writing.

[1]All instance types were priced using Northern Virginia pricing for consistency (all three clouds have regions there) but some of our tests were run in other regions due to availability at the time of testing.



OLTP benchmark

Top 10 instances in price-for-performance ($/TPM)

Median $/TPM for reserved and on-demand offerings, 1,200 warehouse test

At a fixed workload complexity of 1,200 warehouses, larger nodes offered the best price-for-performance, primarily because a greater proportion of the small nodes' price is storage. This is partially due to having the smaller nodes (relatively) overprovisioned in comparison to the larger nodes – we feel this reflects likely real-world use cases, but it does give the larger nodes a pricing advantage in this comparison.

Unsurprisingly given the impact of storage costs on overall cost-to-operate (discussed later in this report), all of the top ten instances here use the clouds' more affordable general-purpose storage options.

Overall, pricing is very linear, with pricing scaling by the number of vCPUs, GB of RAM, or the amount of disk allocated. In most cases you can expect to pay roughly the same amount for these resources across all three clouds, at least for all the configurations we considered.

Note that the winning instance type, GCP's t2d-standard-32, is not available in all GCP regions as of this writing.

You get what you pay for, but not always what you don't

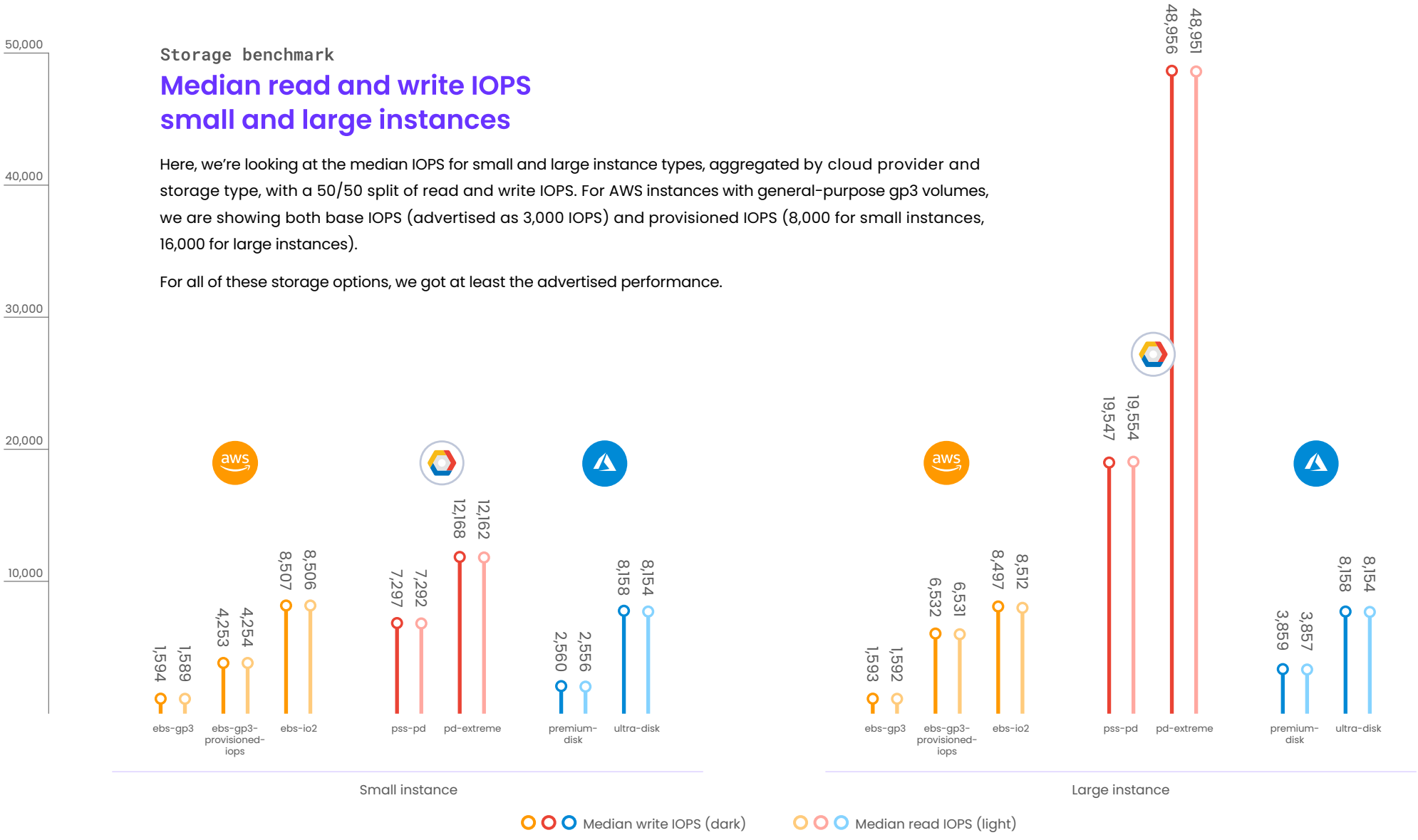
In general, when we paid explicitly for a resource we got the promised performance out of that resource. For example, cross-region throughput was extremely consistent for all three providers, as was storage performance.

Storage benchmark

Median read and write IOPS small and large instances

Here, we're looking at the median IOPS for small and large instance types, aggregated by cloud provider and storage type, with a 50/50 split of read and write IOPS. For AWS instances with general-purpose gp3 volumes, we are showing both base IOPS (advertised as 3,000 IOPS) and provisioned IOPS (8,000 for small instances, 16,000 for large instances).

For all of these storage options, we got at least the advertised performance.

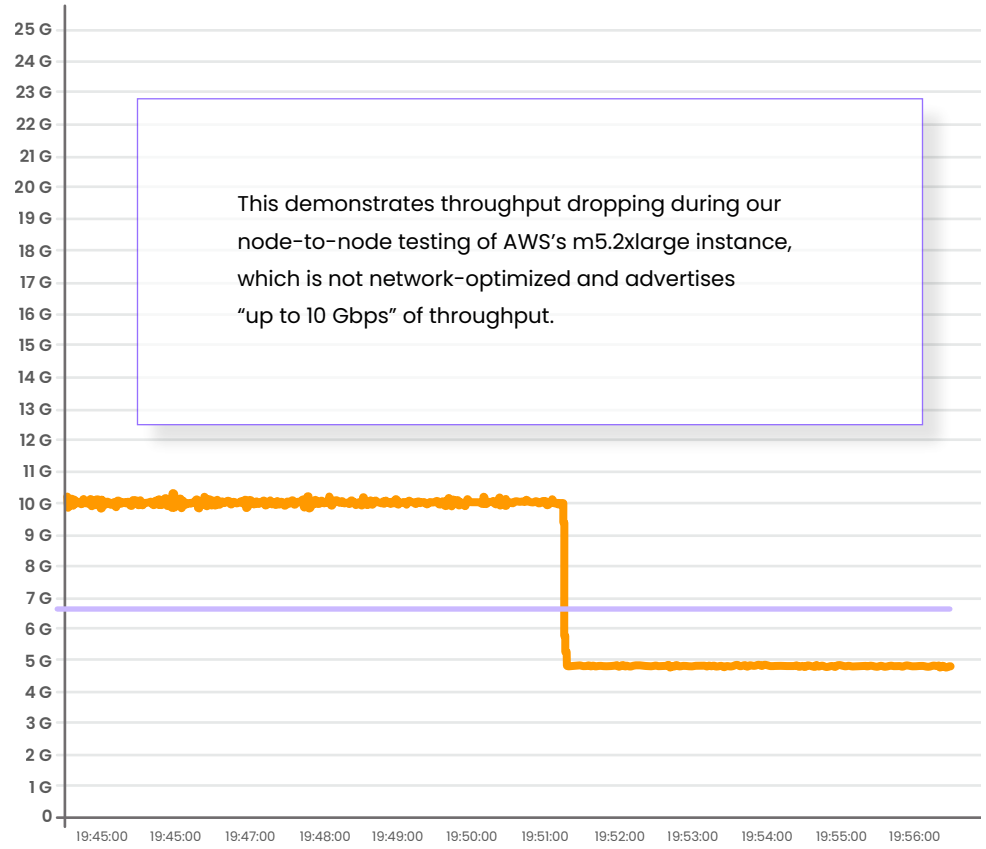


Node-to-node throughput tests

In cases where the cloud provider advertised a performance range (e.g. “up to 10 Gbps”) and did not explicitly charge for a specific level of performance, the results varied between cloud providers and between runs.

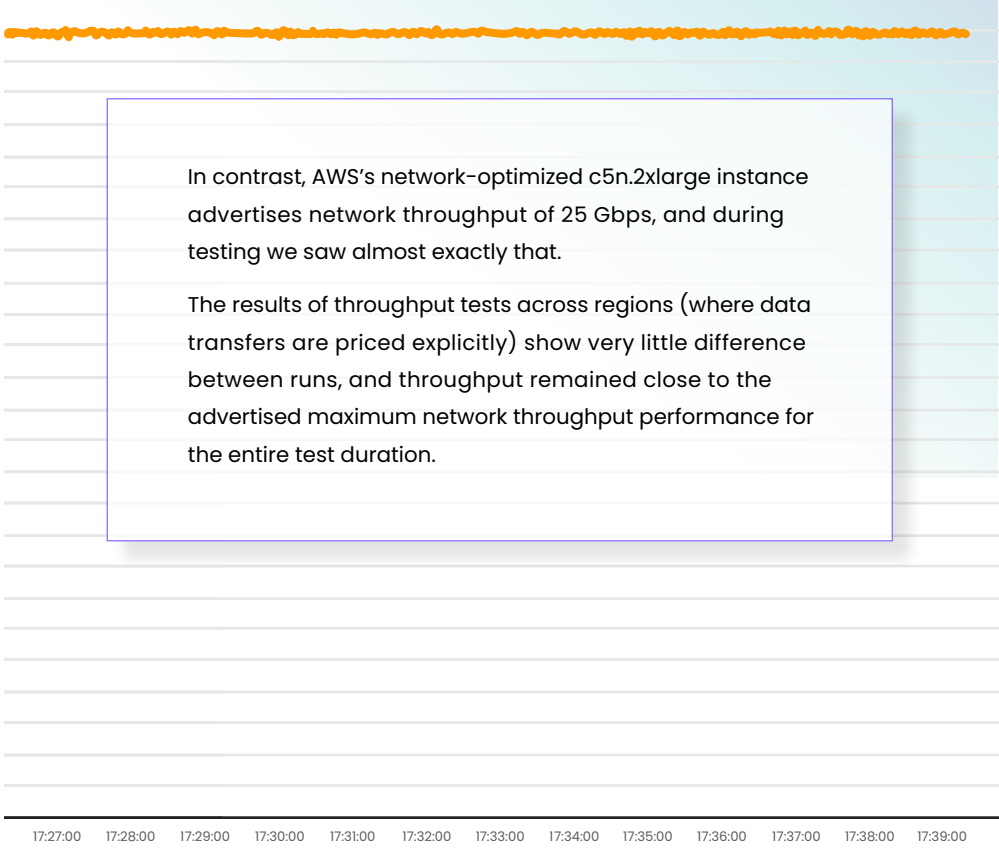
This was best highlighted by the throughput tests in AWS. For intra-AZ data transfers (which are not priced explicitly), we saw a wide range in performance for non-network optimized instance types, including some apparent throttling.

3/5/22 throughput test (AWS m5.2xlarge)



This demonstrates throughput dropping during our node-to-node testing of AWS’s m5.2xlarge instance, which is not network-optimized and advertises “up to 10 Gbps” of throughput.

3/5/22 throughput test (AWS c5n.2xlarge)



In contrast, AWS’s network-optimized c5n.2xlarge instance advertises network throughput of 25 Gbps, and during testing we saw almost exactly that.

The results of throughput tests across regions (where data transfers are priced explicitly) show very little difference between runs, and throughput remained close to the advertised maximum network throughput performance for the entire test duration.

Storage and transfer costs have an outsized impact on total cost to operate

For even relatively small amounts of persistent block storage, the cost of running a particular workload is much more influenced by the cost of the storage than it is the cost of the instance. For persistent workloads, it is extremely important to optimize cost calculations based on this consideration.

This means preferring mid-tier storage options (pd-ssd on GCP, gp3 on AWS, premium-disk on Azure) unless your workload requires an either extremely high number of IOPS or very low storage latency. Among the instance types we tested this year, we did not find a single case where the most cost-effective choice was to use top-tier storage (pd-extreme on GCP, io2 on AWS, ultra-disk on Azure) for any of the OLTP benchmarks.

OLTP benchmark

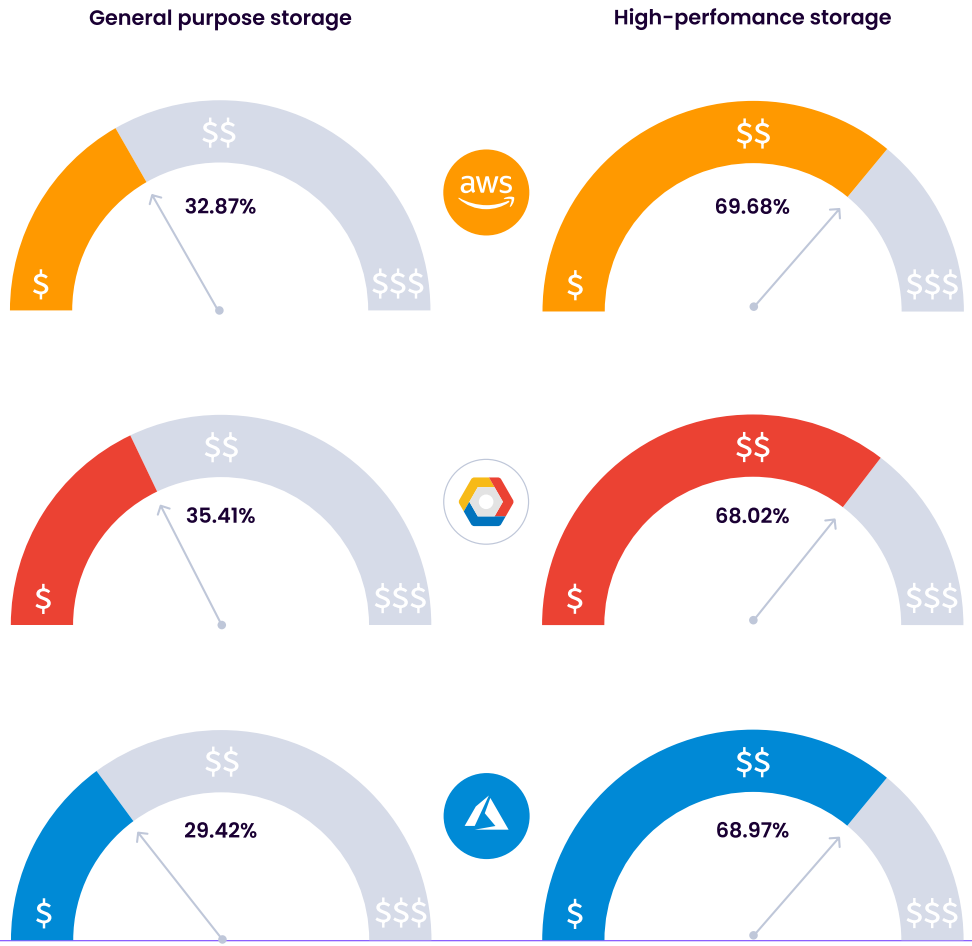
Average storage cost as a percentage of total instance cost

Total instance cost includes storage cost

This chart shows average storage costs as a percentage of the total instance costs (including storage). For configurations using high-performance storage, the storage cost accounted for the majority of the total cost to operate.

While not accounted for in our $/TPM analysis, the biggest cost for this testing was data transfer. None of the three cloud providers explicitly charges for node-to-node data transfers within an availability zone, but all of them charge $0.01-$0.02 per GB transferred across zones within a region. Cross-region costs vary substantially depending on location. Within the United States, all of the cloud providers charge between $0.01-$0.02 per GB transferred between regions as well (as of this writing).

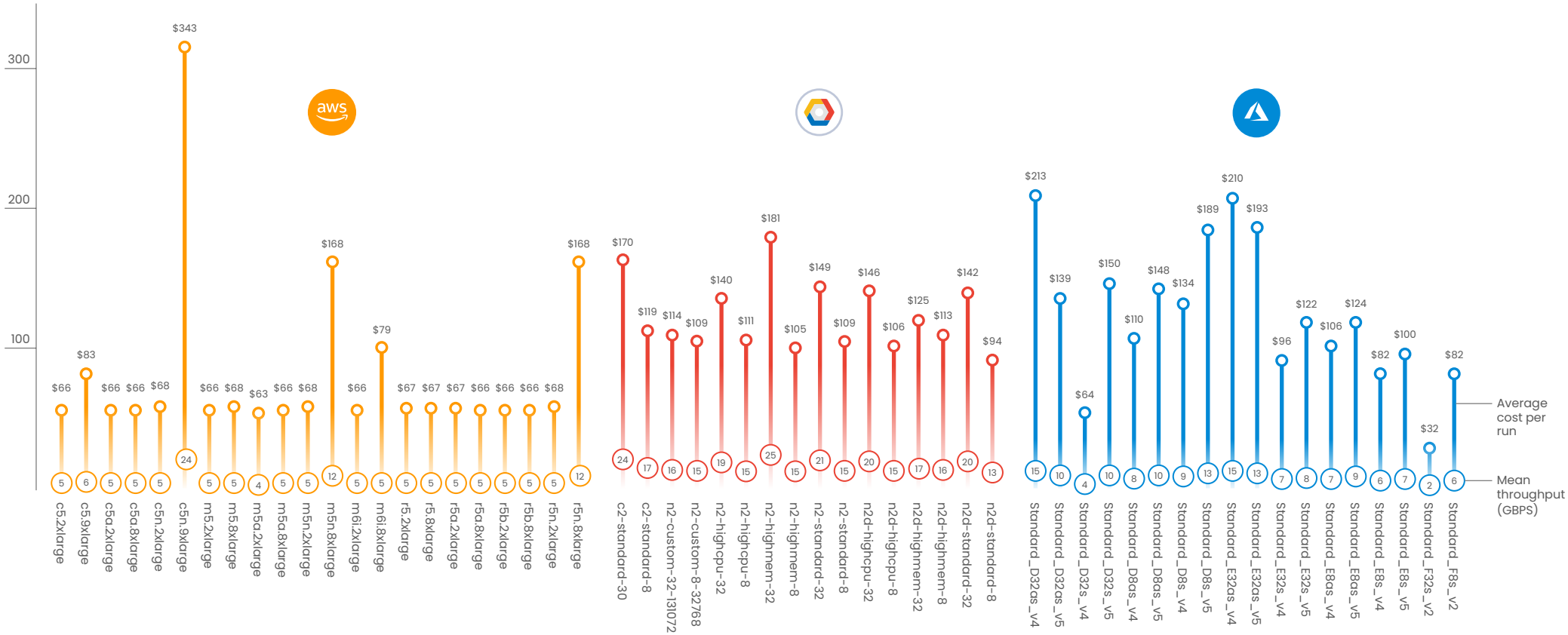
This means that in some cases, it may cost the same to run a multi-region workload as it does to run a multi-AZ workload, while improving the survivability of the application.



Average cost per run and mean throughput

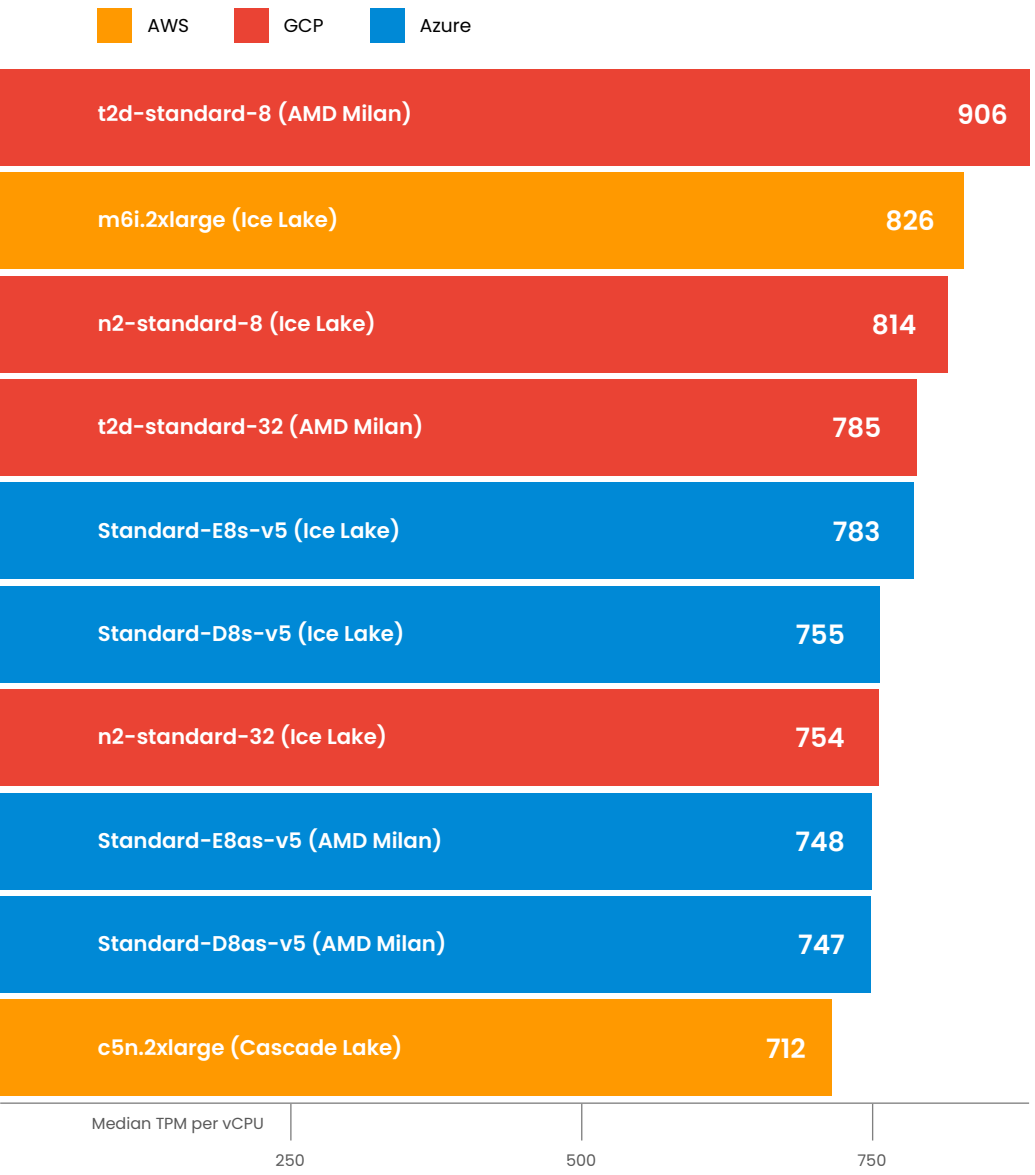
Here, we see the average cost for a single run of the cross-region network throughput test for each instance type and the corresponding throughput we saw. For our specific tests, AWS and Azure both charged $0.02/per GB and GCP charged $0.01/per GB. These tests ran for a total of 12 minutes each. Given those conditions, GCP had an overall advantage in both performance and cost.

When building a highly resilient stateful application, the "hidden" costs of storage and data transfer can have a larger impact on total cost than the price of the instances themselves. In future years, we hope to include more of these "hidden costs" in our price-for-performance analysis as a part of the OLTP benchmark.



Small instances outperform large instances

In both the OLTP and CPU benchmarks, we saw a per-vCPU performance advantage to running on smaller instances of a particular instance type, regardless of CPU platform, cloud, or instance type. Depending on the instance type, these differences were anywhere from 4-5% on the low end and upwards of 30% on the high end.



OLTP benchmark

Top 10 instances by TPM per vCPU 1,200 warehouses

Looking at the top ten instance types on a TPM per vCPU basis, small instance sizes largely won in terms of both overall and average performance (regardless of storage configuration). GCP's t2d-standard-8 and AWS's m6i.2xlarge led the pack, and overall eight out of the top ten instances were the small 8 vCPU nodes.

To account for some of these differences, we performed tests including:

- Short-Running Tests – In the short-running CPU benchmark, the smaller nodes had a 3-4% per-vCPU speed advantage.
- Scaled-Up Tests – We did some scaled-out runs of our OLTP benchmark that had an identical number of cores allocated to both large-node and small-node database clusters to account for benchmark overhead. While this closed the gap slightly, it didn't account for the entire performance difference.

We did discover some factors that contributed to the overall finding, although none is sufficient to explain the entire difference in performance for all instance types:

- Instances that have multiple NUMA nodes substantially under-perform instances running on a single NUMA node for many workloads. For some instance types, this accounted for the entire performance difference between small and large instances.
- All instances have some form of variable clock-speed technology implemented. Sample tests showed that smaller instances tended to run at slightly higher average clock speeds than larger instances under sustained load, but this is not enough to explain the performance difference.

This is a non-exhaustive list of things we looked at to account for the performance differences between the instance sizes, and there were several additional dimensions we didn't have the chance to fully explore. More details on our investigation into this performance discrepancy are available in the Appendix II.

vCPU to RAM ratio directly impacts performance consistency

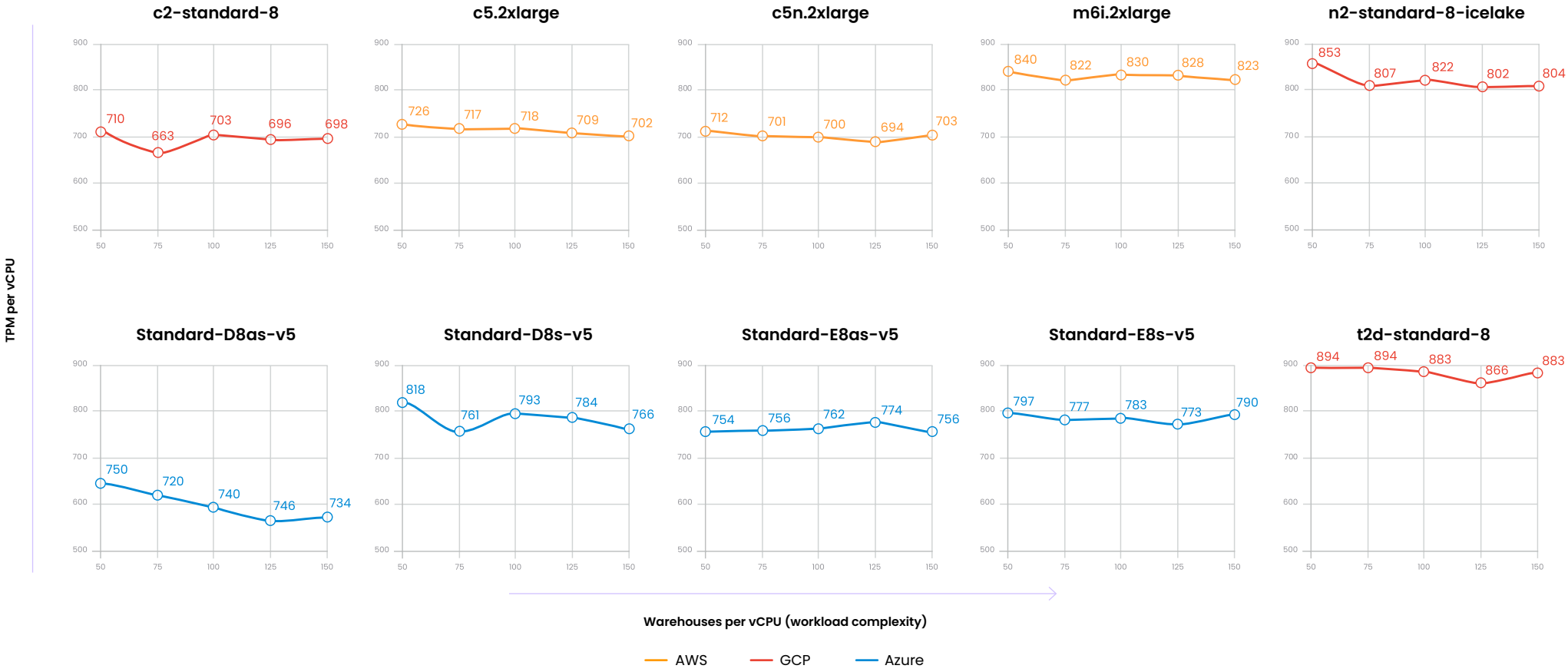
When looking across all of the different OLTP benchmark configurations we ran, all of the top ten instance types for both small and large instance types have a ratio of at least 4 GB of RAM per vCPU.

In these charts, warehouses per vCPU serves as a proxy for workload complexity; runs with higher Warehouses Per vCPU counts involve larger, more complicated queries running in the background. That complexity scales non-linearly with the number of vCPUs and the number of warehouses (this is why there are separate charts for the small and large instances).

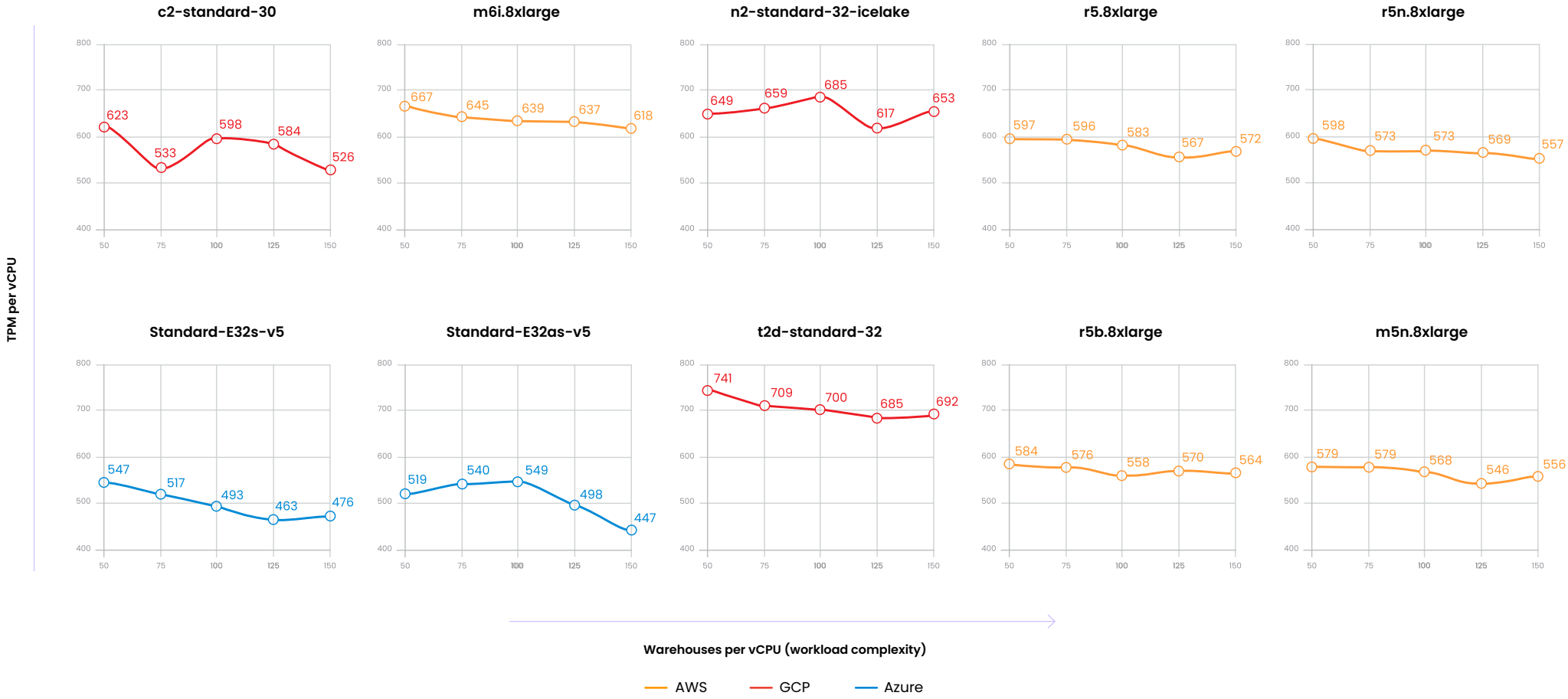
Looking at TPM across a variety of workload complexities can tell us the suitability of a particular instance type under different types of load, and can expose situations where the performance of an instance type degrades as workload complexity increases.

`Network benchmark`

Top 10 small instances by performance (TPM), increasing workload complexity



Top 10 large instances by performance (TPM), increasing workload complexity



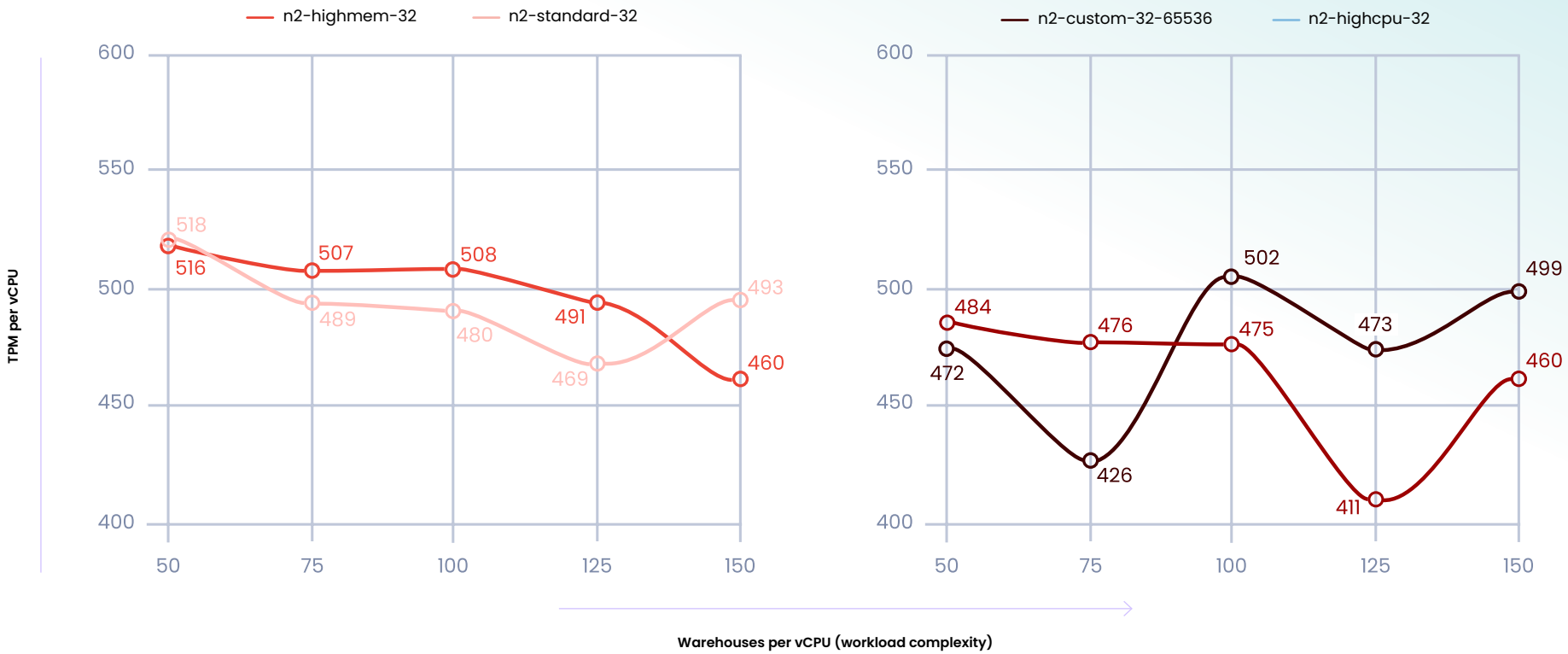
For example, when comparing otherwise-identical GCP instance types (highcpu with a 1:1 ratio, custom with a 2:1 ratio, standard with a 4:1 ratio, and highmem with an 8:1 ratio), we can see that both n2-custom and n2-highcpu have inconsistent performance, whereas n2-standard and n2-highmem remained relatively steady as workload complexity increased.

When comparing this to other testing results, we interpret this to mean that while you may save a bit of money by choosing instance types with a lower vCPU to RAM ratio, you will likely see more consistent performance from instance types with more available memory, at least for the purposes of running CockroachDB.
The impact of this decision is more apparent with larger, more complicated workloads.

In our tests, we found the sweet spot to be a vCPU:RAM ratio of 1:4.

`OLTP benchmark`

Selected GCP instances by performance (TPM), increasing workload complexity



Cockroach Labs

2022 Cloud Report

Detailed benchmarks

OLTP benchmarks

Methodology

This year, we scaled up the number and variations of our OLTP benchmark dramatically, and designed a Cockroach Labs Derivative TPC-C nowait benchmark that allowed us to scale the benchmark with a fine granularity using a fixed load multiplier per vCPU.

As the name suggests, our benchmark is based on TPC-C. TPC-C is a standards-based benchmark from the **Transaction Processing Performance Council** that attempts to simulate a real-world logistics system, creating and taking virtual customer orders from initial receipt through the manufacturing process and then out for delivery. It imitates real-life user interactions with the system by limiting how fast warehouses move orders through the fulfillment process.

As warehouses are added to the system, both query complexity and system load increase. However, this means that the core metric of TPC-C (transactions per minute, i.e. the number of new orders that can be processed per minute while the rest of the workload is running) is not directly comparable across runs with different warehouse counts. Also, because of the simulated wait times, you need to be relatively close to over-saturating the database before differences between different cloud configurations become apparent.

In an effort to compare across instance types and instance sizes fairly, we attempted to separate scaling the number of transactions processed from the complexity of the workload by removing wait times from this year's testing. This allowed us to get a better comparative signal across instances all running the same database. To do this, we moved to a new Cockroach Labs Derivative TPC-C nowait benchmark. While we think that this has improved our ability to discern small performance differences between configurations, it does mean that the numbers we're reporting for this benchmark are not directly comparable to what we reported in the 2021 Cloud Report.

OLTP benchmark: configuration details

Our testing with the Cockroach Labs Derivative TPC-C nowait benchmark used the following configuration parameters:

- Wait times disabled (wait=0 in our test harness).
- "Warehouse" per vCPU scaling (50, 75, 100, 125, 150 warehouses per vCPU).
- Since query complexity scales with the warehouse count, we picked a discrete warehouse count (1200 warehouses) for our direct performance comparisons between large and small nodes.
- Set the "Active Workers" count to be equal to the "Warehouse" count.
- Set the number of active connections at the load generator as equal to 4x the number of vCPUs, as per CockroachDB's production guidelines.

Most of our comparisons were made on a per-vCPU basis to avoid a bias toward any instances with higher core counts. This includes pricing comparisons as well as raw performance numbers. Since these are all multi-server runs, we also collected information to determine whether or not we got identical nodes (i.e. CPU info including the number of NUMA nodes our vCPUs are running across).

All configurations (combinations of instance type, instance size, and storage used) had a minimum of four runs executed for that configuration. In cases where we had fewer than three valid runs (for example, if one of the instances got a slightly different processor, or a different NUMA count than the others), we attempted to run additional runs until we got a minimum of four wherever possible. For the few cases where we couldn't reliably get a consistent configuration across all nodes, we may have chosen to report the other runs so we could include that instance type in the final report.

All reported database tests were run on a cluster of three database nodes and leveraged one additional load generator node, all identically provisioned with CPU, network, and storage.

Metrics

The core metrics measured by the Cockroach Labs Derivative TPC-C nowait benchmark are new-order transactions per minute (TPM) and cost per new-order transaction per minute ($/TPM). These are all calculated while running the other queries in the workload as well; they are not raw query metrics.

All prices were calculated as cost per month per month, with reserved pricing assuming a one-year commitment.

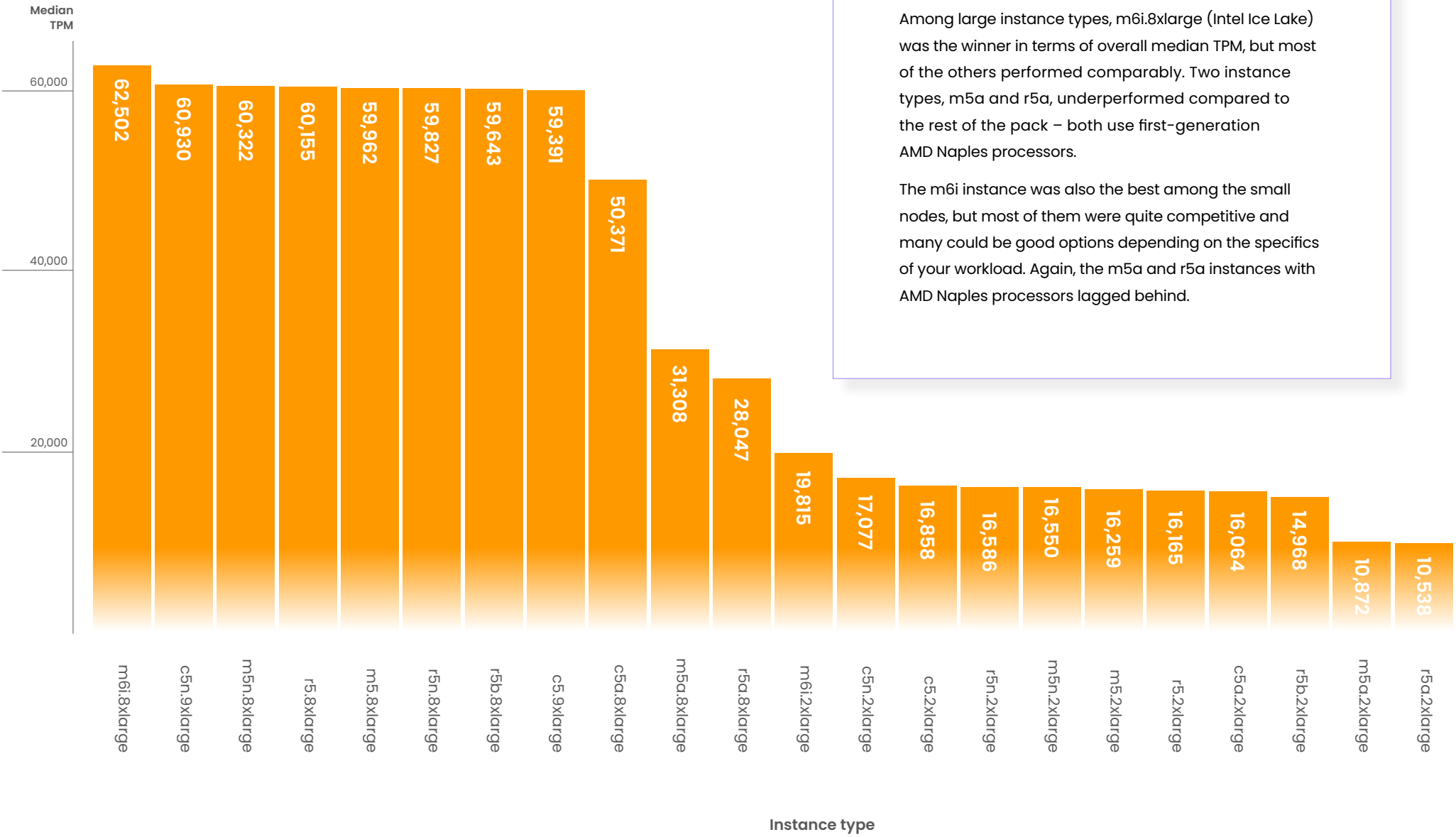
AWS results

OLTP benchmark

Overall median TPM 1,200 warehouses

Among large instance types, m6i.8xlarge (Intel Ice Lake) was the winner in terms of overall median TPM, but most of the others performed comparably. Two instance types, m5a and r5a, underperformed compared to the rest of the pack – both use first-generation AMD Naples processors.

The m6i instance was also the best among the small nodes, but most of them were quite competitive and many could be good options depending on the specifics of your workload. Again, the m5a and r5a instances with AMD Naples processors lagged behind.



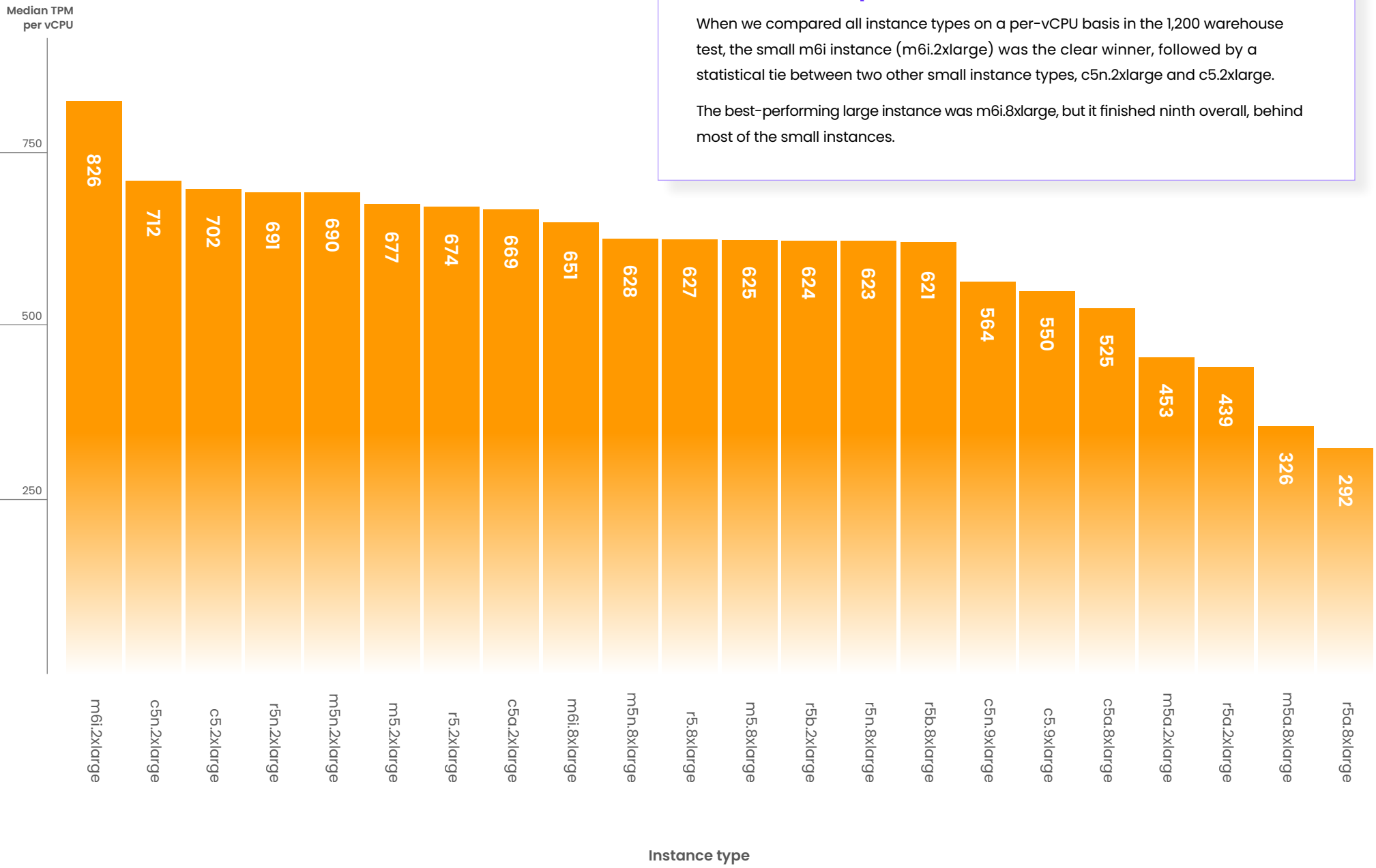
AWS results

OLTP benchmark

Median TPM per vCPU 1,200 warehouses

When we compared all instance types on a per-vCPU basis in the 1,200 warehouse test, the small m6i instance (m6i.2xlarge) was the clear winner, followed by a statistical tie between two other small instance types, c5n.2xlarge and c5.2xlarge.

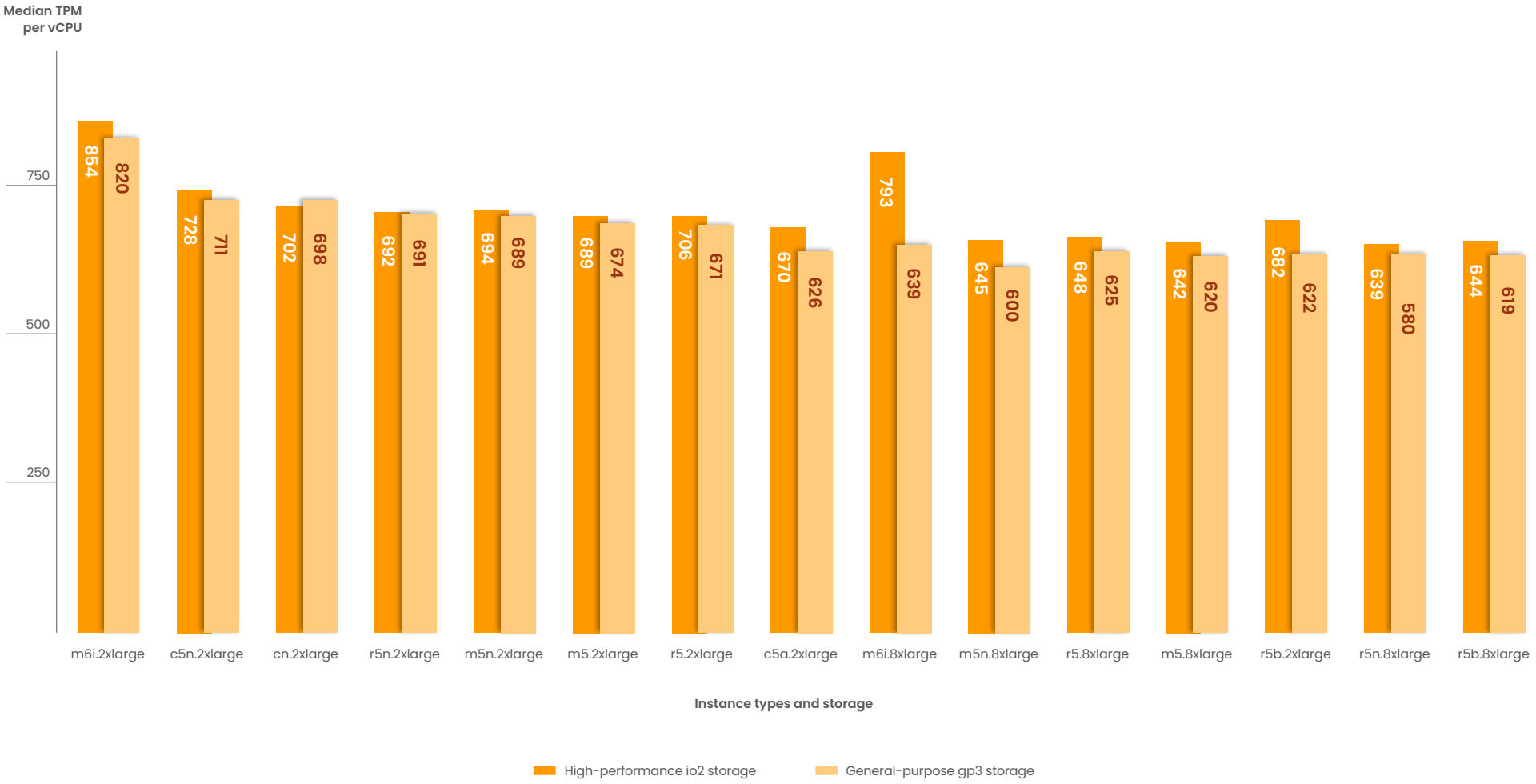
The best-performing large instance was m6i.8xlarge, but it finished ninth overall, behind most of the small instances.



OLTP benchmark

Median TPM per vCPU with storage 1,200 warehouses

When we looked at the 1,200 warehouse test for each instance with general-purpose and high-performance storage options, we saw that generally the performance difference between high-performance io2 storage and general-purpose gp3 storage was negligible. The variation between them was within the margin of error for many instance types. One notable exception was m6i.8xlarge, which was much more competitive with the smaller instances on a per-vCPU basis when using the high-performance io2 storage.

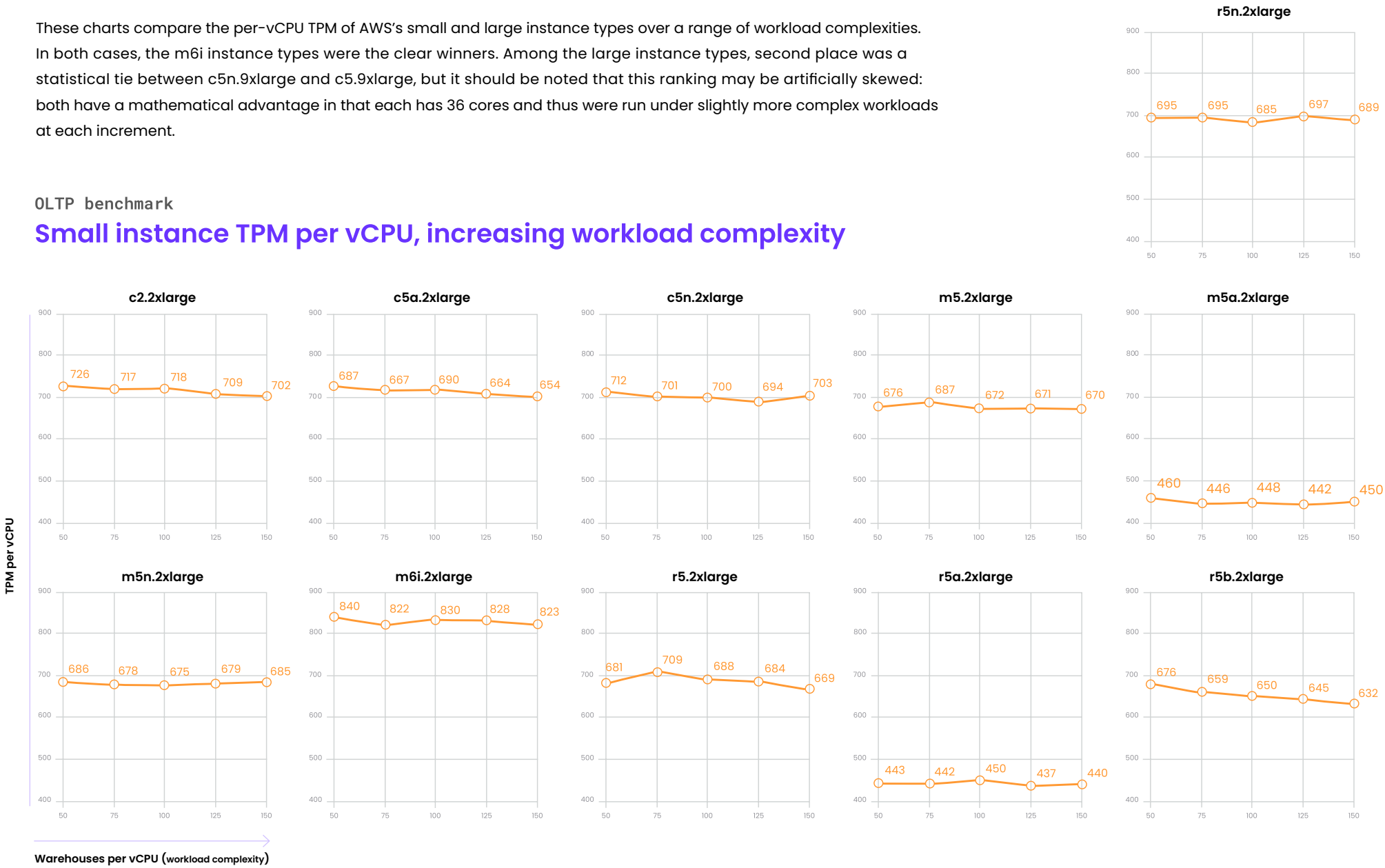


AWS results

These charts compare the per-vCPU TPM of AWS's small and large instance types over a range of workload complexities. In both cases, the m6i instance types were the clear winners. Among the large instance types, second place was a statistical tie between c5n.9xlarge and c5.9xlarge, but it should be noted that this ranking may be artificially skewed: both have a mathematical advantage in that each has 36 cores and thus were run under slightly more complex workloads at each increment.

OLTP benchmark

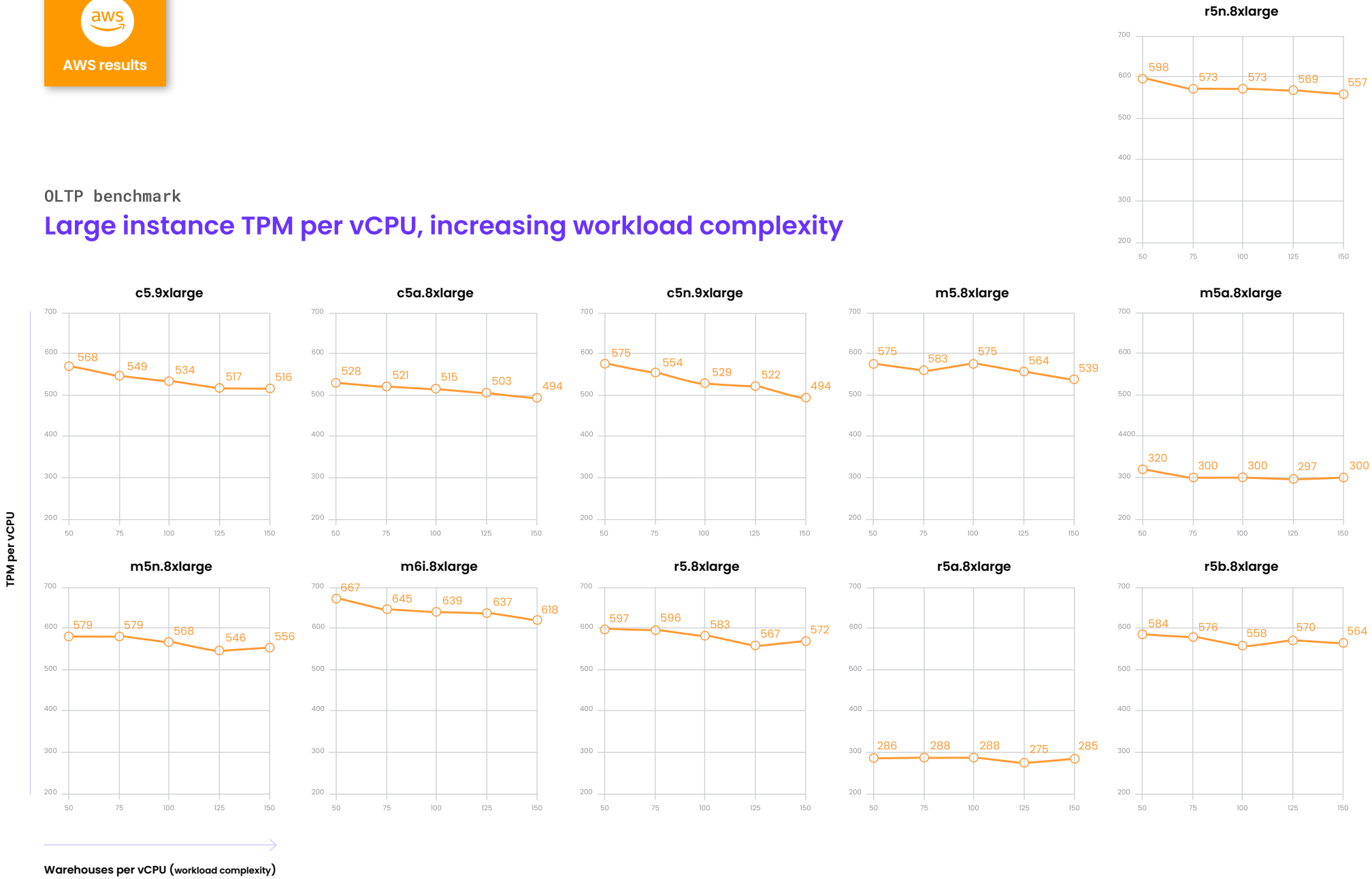
Small instance TPM per vCPU, increasing workload complexity



AWS results

OLTP benchmark

Large instance TPM per vCPU, increasing workload complexity



AWS results

OLTP benchmark

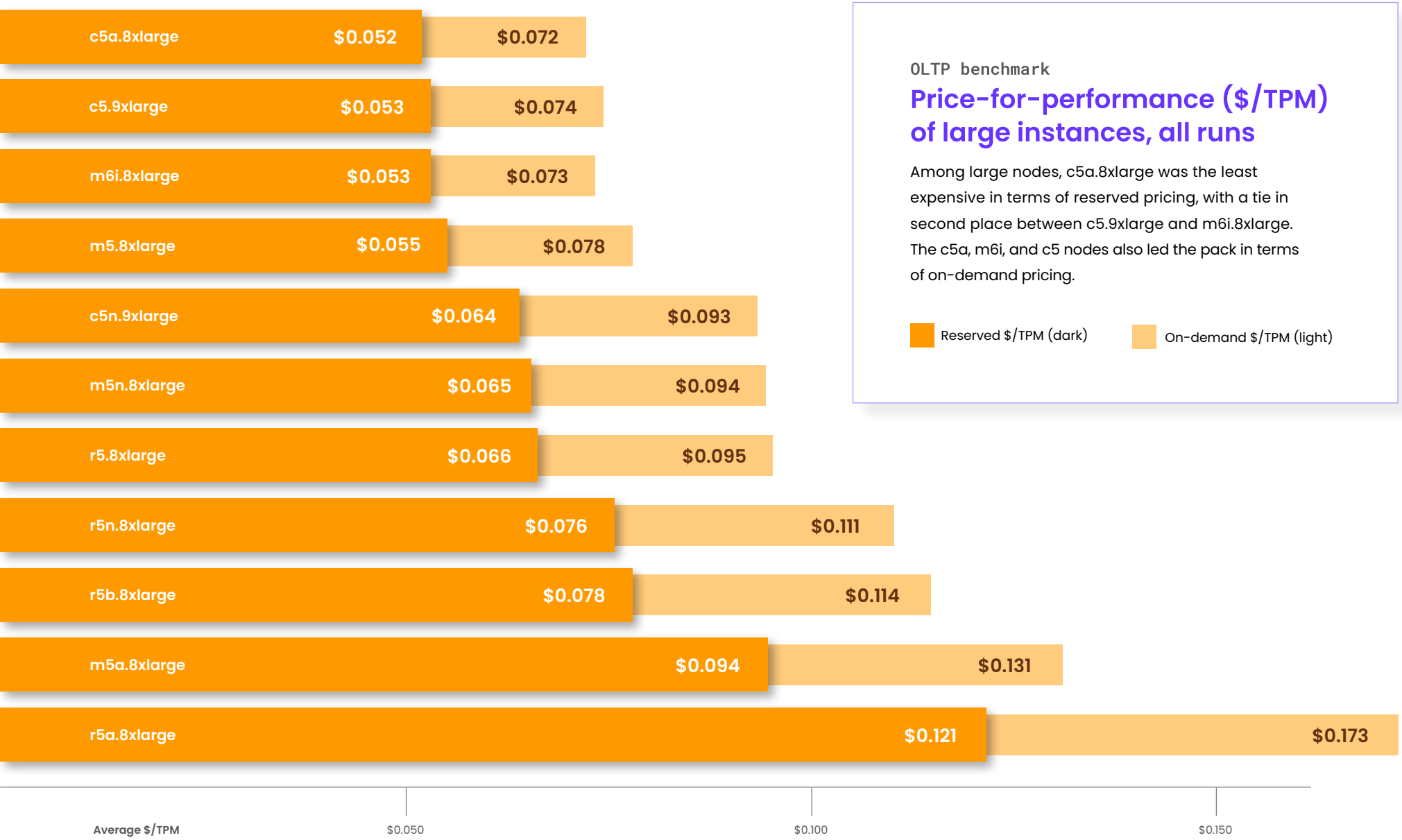
Price-for-performance ($/TPM) of small instances, all runs

Here, we have broken down the price-for-performance ($/TPM) of small instance types across all workload complexities (warehouse counts). When comparing reserved pricing, m6i.2xlarge was the least expensive, followed by a tie for second place between c5a.2xlarge and c5.2xlarge. The m6i and c5a nodes also led the pack in $/TPM using on-demand pricing.

Reserved $/TPM (dark)

On-demand $/TPM (light)

AWS results



OLTP benchmark

Price-for-performance ($/TPM) of large instances, all runs

Among large nodes, c5a.8xlarge was the least expensive in terms of reserved pricing, with a tie in second place between c5.9xlarge and m6i.8xlarge. The c5a, m6i, and c5 nodes also led the pack in terms of on-demand pricing.

Reserved $/TPM (dark)

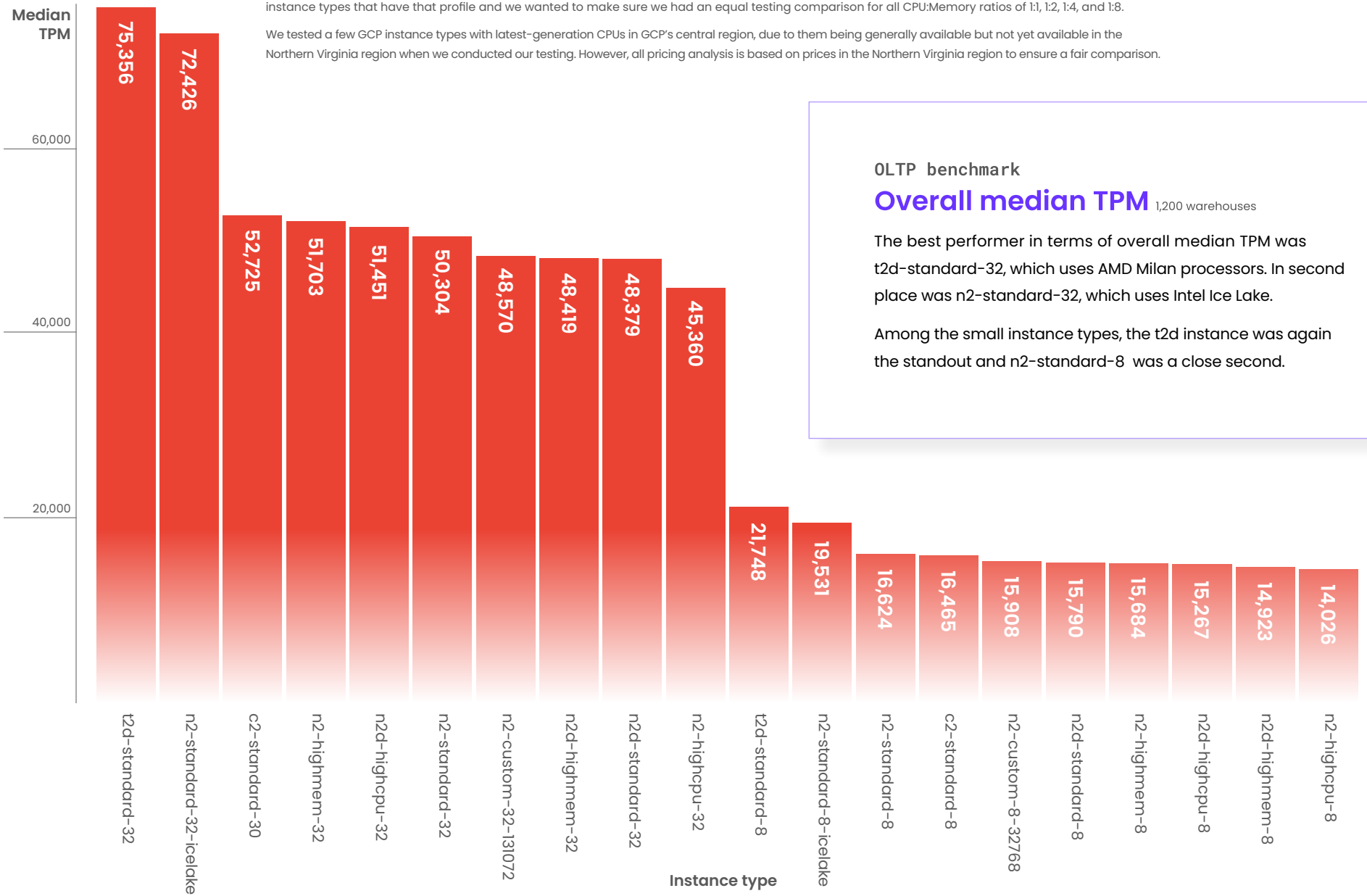
On-demand $/TPM (light)

GCP results

Understanding the results

In the results below, the *-custom-* instance types are built at a 1:2 vCPU to RAM ratio (i.e. one vCPU for every 2 GB of RAM). Azure and AWS both have standard instance types that have that profile and we wanted to make sure we had an equal testing comparison for all CPU:Memory ratios of 1:1, 1:2, 1:4, and 1:8.

We tested a few GCP instance types with latest-generation CPUs in GCP's central region, due to them being generally available but not yet available in the Northern Virginia region when we conducted our testing. However, all pricing analysis is based on prices in the Northern Virginia region to ensure a fair comparison.



OLTP benchmark

Overall median TPM 1,200 warehouses

The best performer in terms of overall median TPM was t2d-standard-32, which uses AMD Milan processors. In second place was n2-standard-32, which uses Intel Ice Lake.

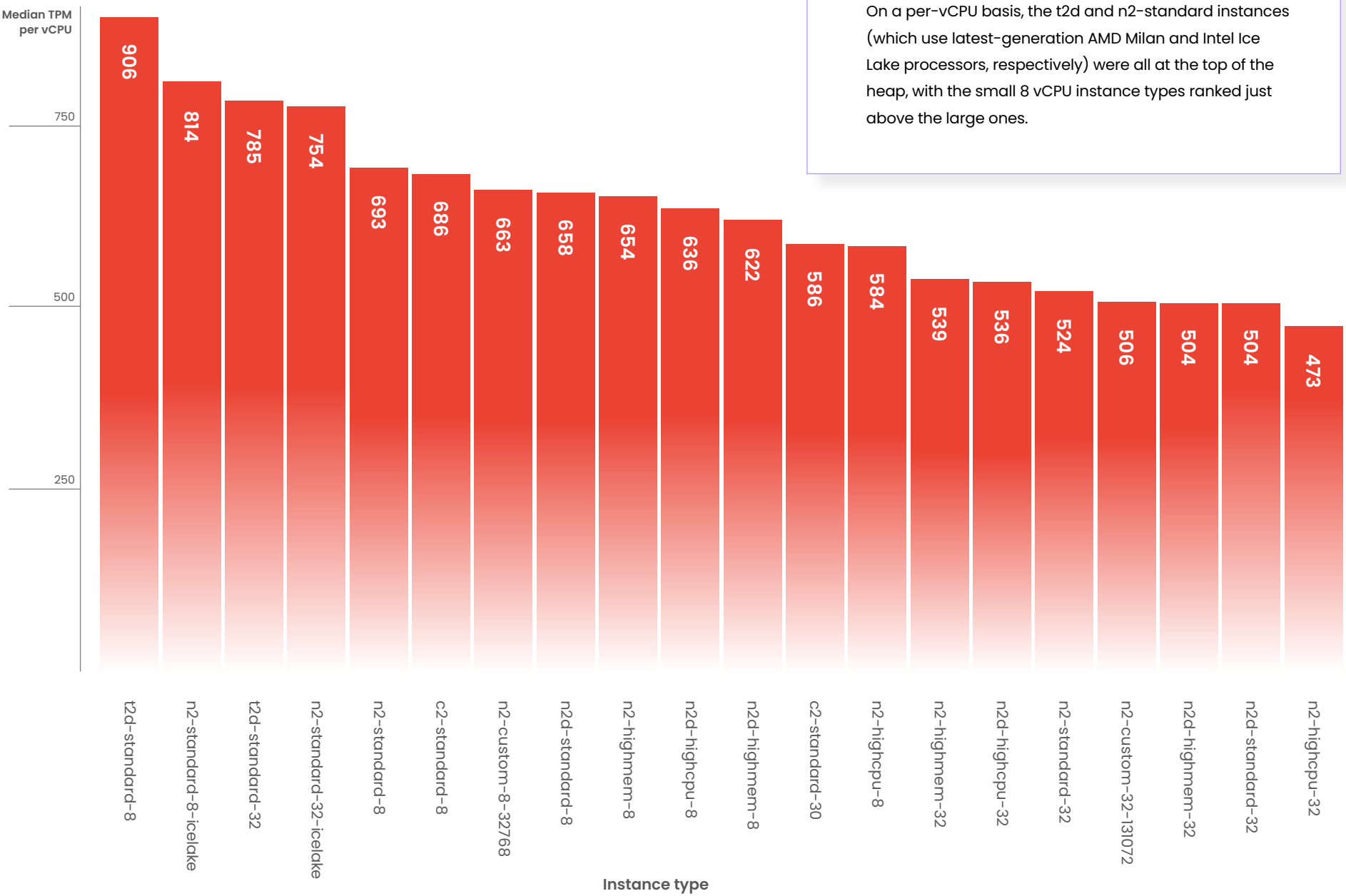
Among the small instance types, the t2d instance was again the standout and n2-standard-8 was a close second.

GCP results

OLTP benchmark

Median TPM per vCPU 1,200 warehouses

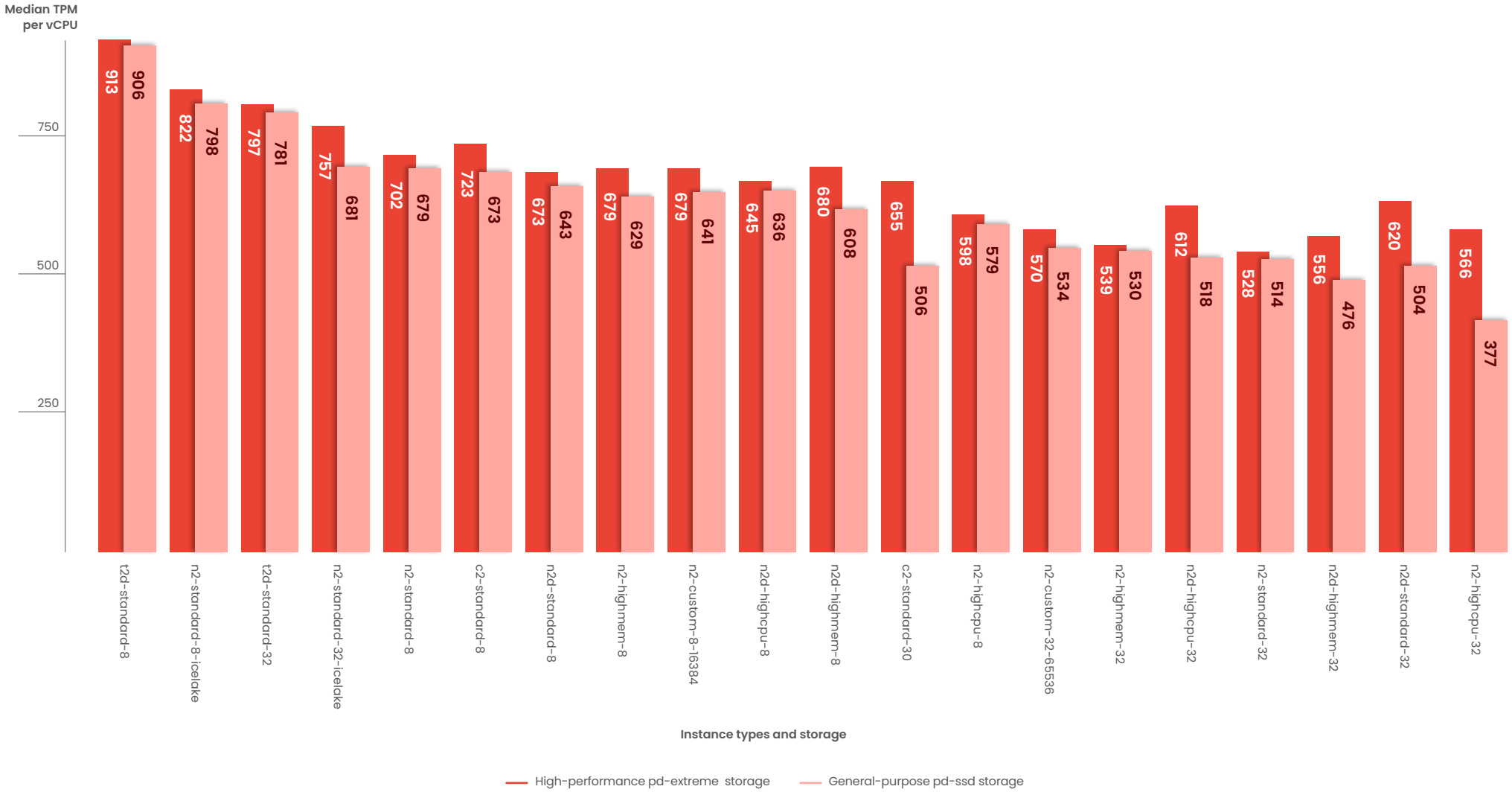
On a per-vCPU basis, the t2d and n2-standard instances (which use latest-generation AMD Milan and Intel Ice Lake processors, respectively) were all at the top of the heap, with the small 8 vCPU instance types ranked just above the large ones.



OLTP benchmark

Median TPM per vCPU with storage 1,200 warehouses

Breaking out the per-vCPU results by storage type, we saw that even on the latest-generation processors, there seemed to be little OLTP performance difference between pd-ssd and pd-extreme storage. Our best run overall on GCP was actually using the general-purpose pd-ssd storage.

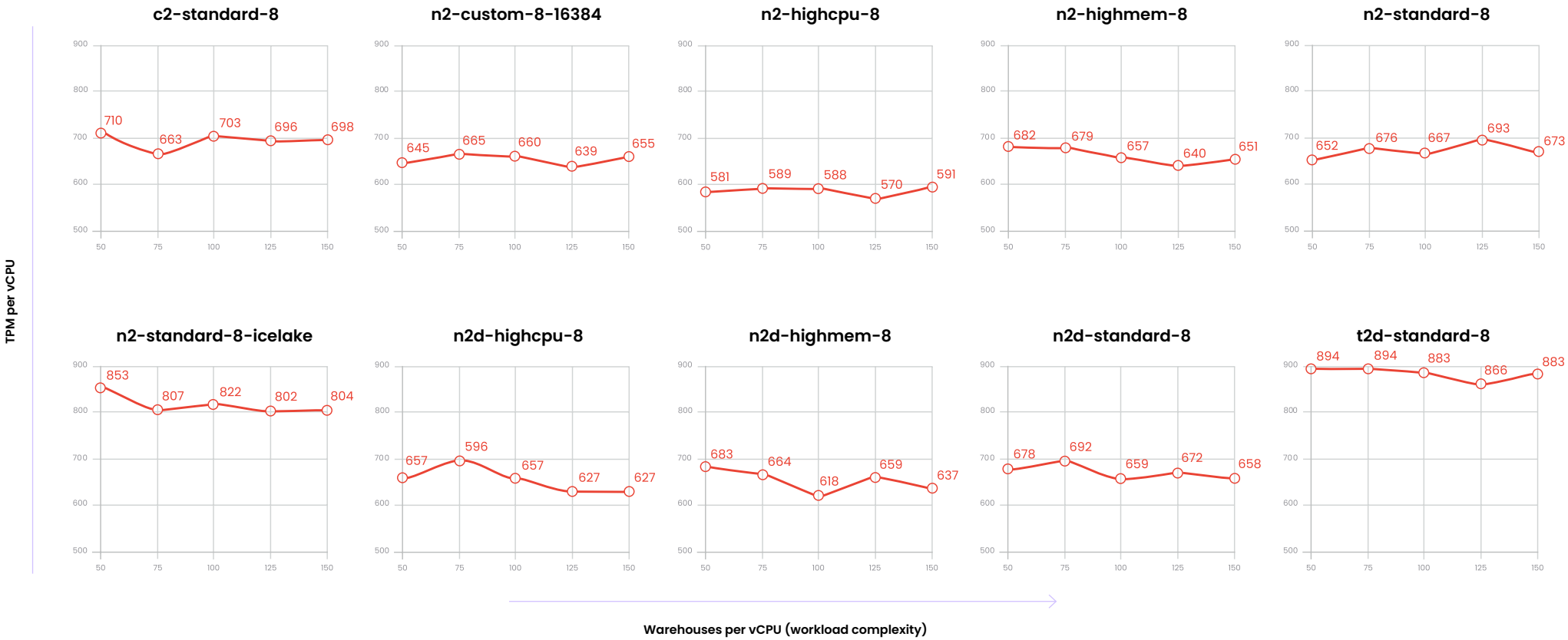


GCP results

These charts compare the per-vCPU TPM of GCP's small and large instance types over a range of workload complexities. In both cases, we saw the t2d and n2-icelake instances come out on top. Particularly among large nodes, a vCPU:RAM ratio of 1:4 appeared to be the sweet spot for achieving consistent performance.

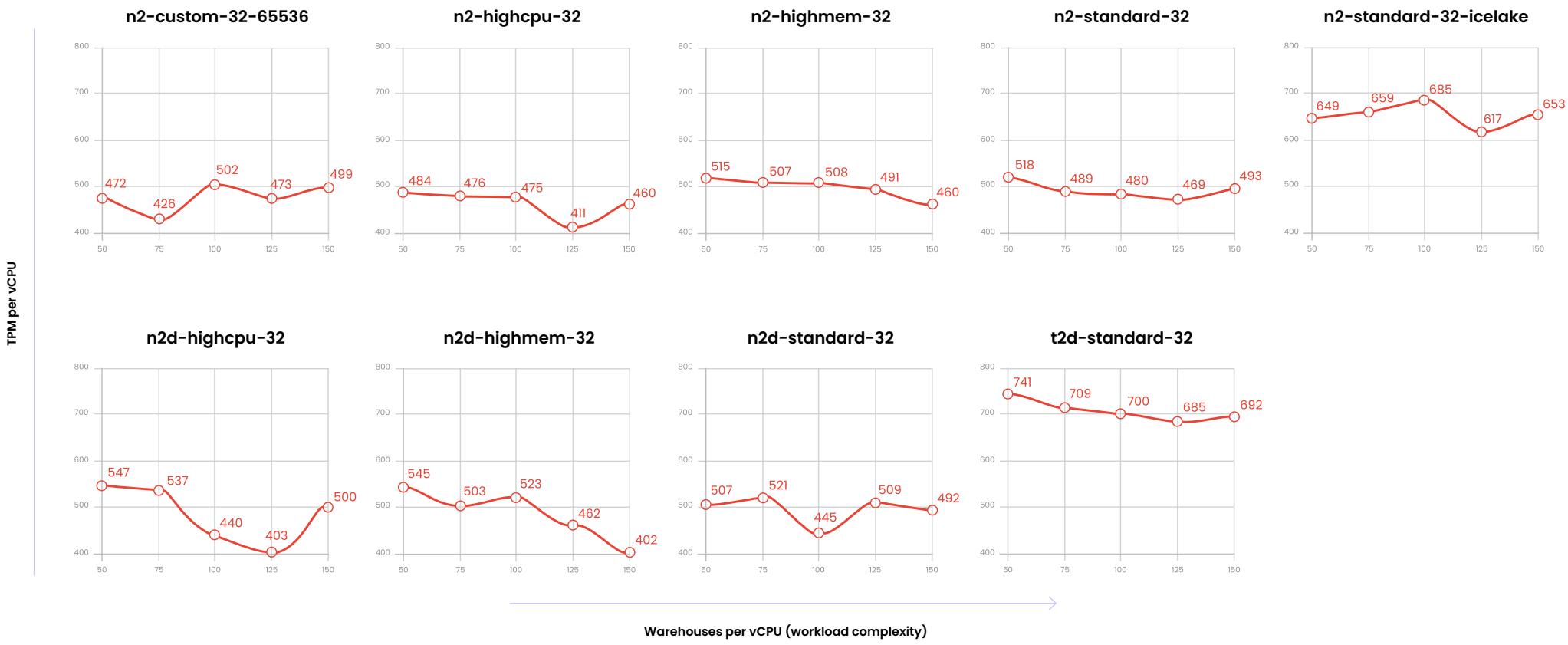
Network benchmark

Small instance TPM per vCPU, increasing workload complexity



Network benchmark

Large instance TPM per vCPU, increasing workload complexity



GCP results

OLTP benchmark

Price-for-performance ($/TPM) of small instances, all runs

When looking at price-for-performance ($/TPM) among the small instance types, the t2d and n2-icelake instances, with latest-generation processors and vCPU:RAM ratios of 1:4, offered the best value in terms of both reserved and on-demand pricing.

Reserved $/TPM (dark)

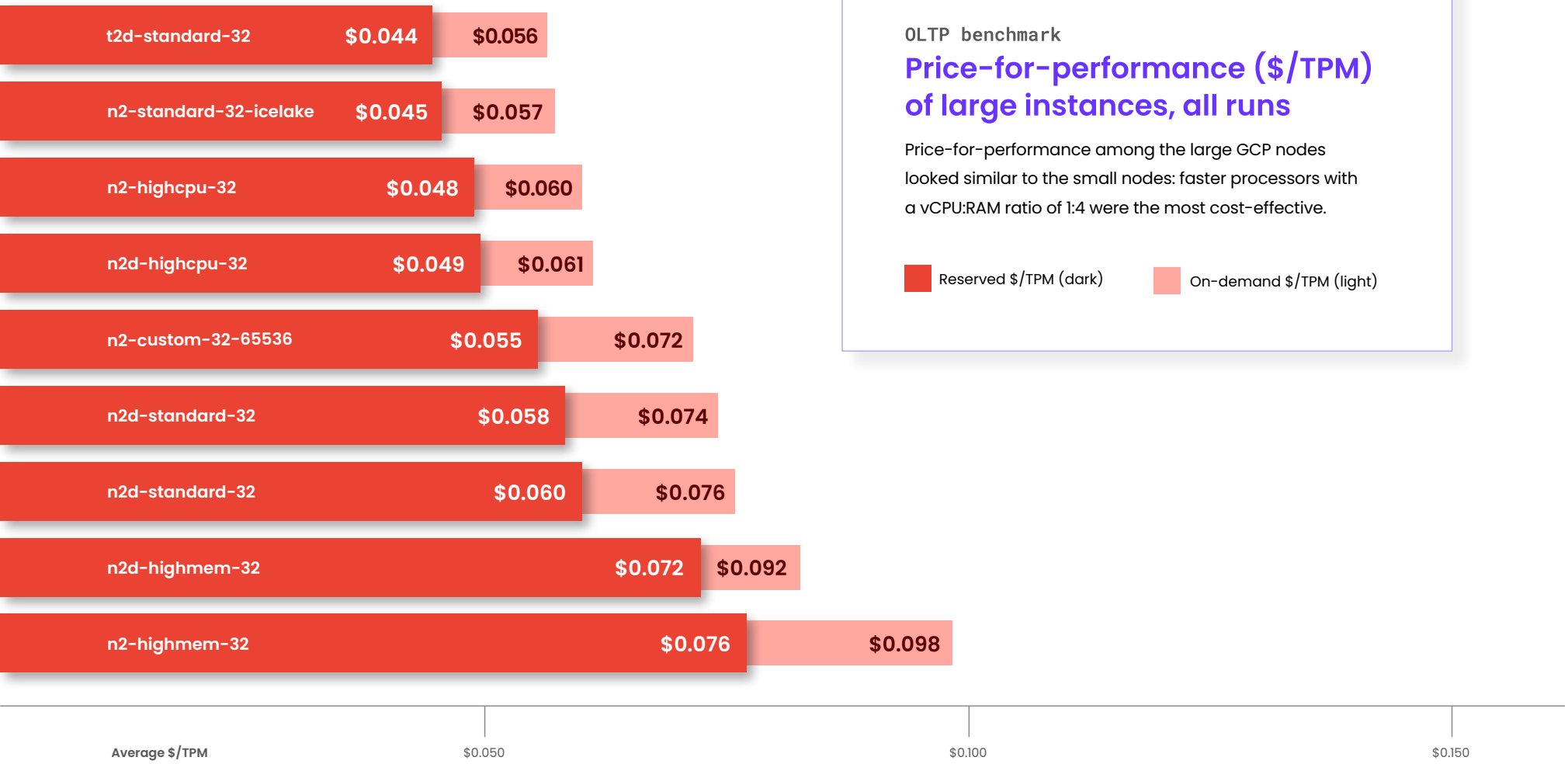
On-demand $/TPM (light)

GCP results

OLTP benchmark

Price-for-performance ($/TPM) of large instances, all runs

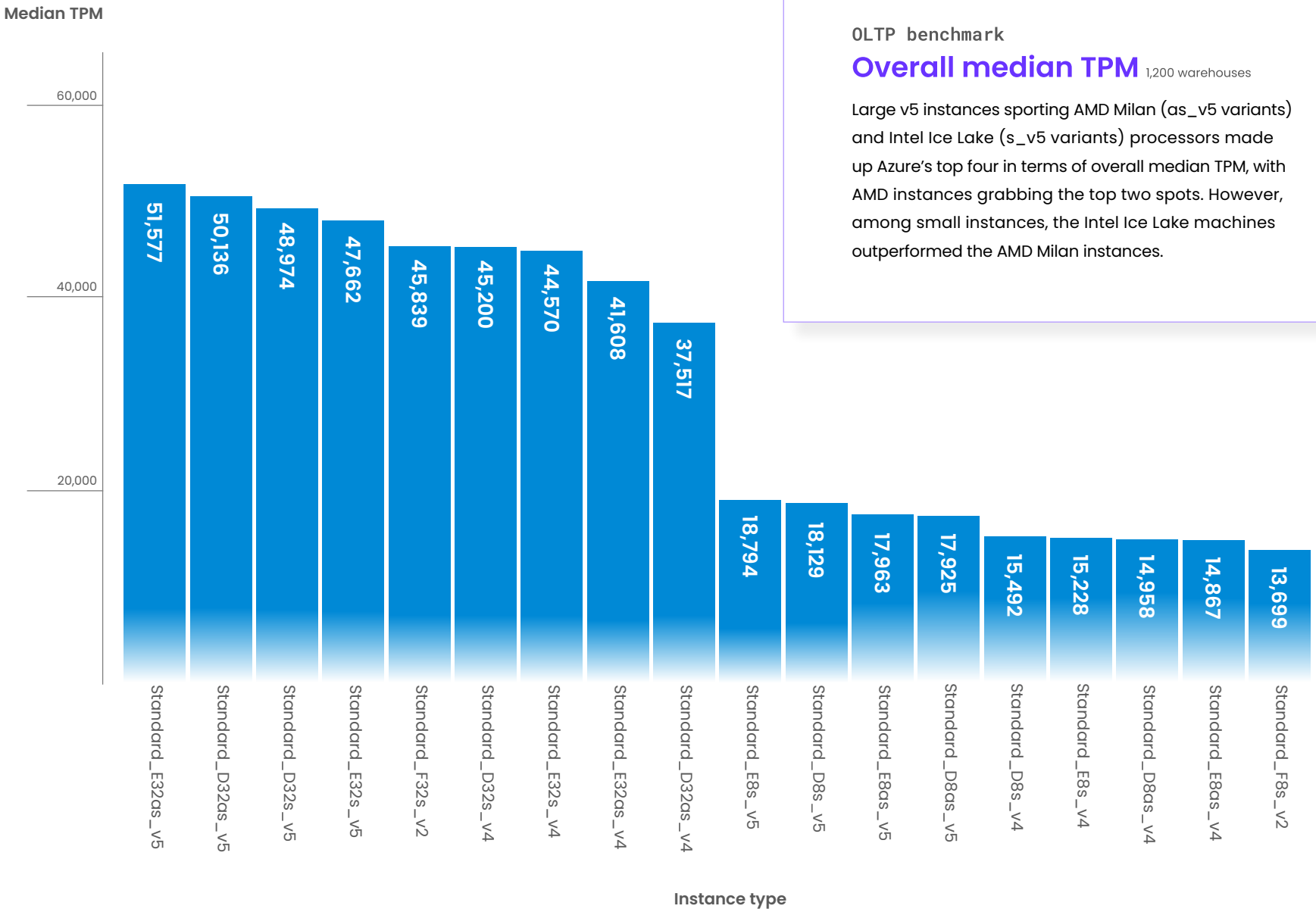
Price-for-performance among the large GCP nodes looked similar to the small nodes: faster processors with a vCPU:RAM ratio of 1:4 were the most cost-effective.



OLTP benchmark

Overall median TPM 1,200 warehouses

Large v5 instances sporting AMD Milan (as_v5 variants) and Intel Ice Lake (s_v5 variants) processors made up Azure's top four in terms of overall median TPM, with AMD instances grabbing the top two spots. However, among small instances, the Intel Ice Lake machines outperformed the AMD Milan instances.

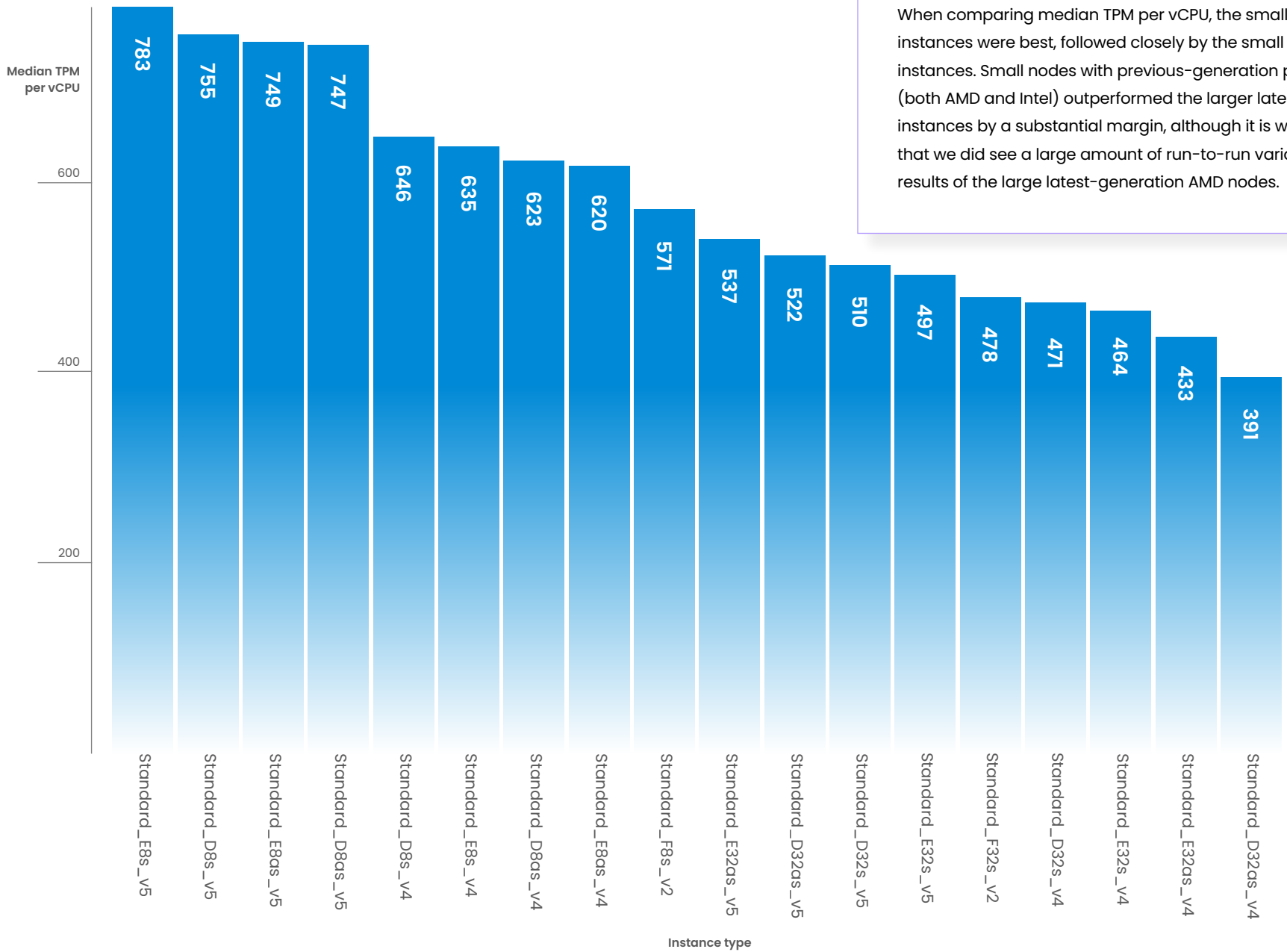


Azure results

OLTP benchmark

Median TPM per vCPU 1,200 warehouses

When comparing median TPM per vCPU, the small Intel Ice Lake instances were best, followed closely by the small AMD Milan instances. Small nodes with previous-generation processors (both AMD and Intel) outperformed the larger latest-generation instances by a substantial margin, although it is worth noting that we did see a large amount of run-to-run variability in the results of the large latest-generation AMD nodes.

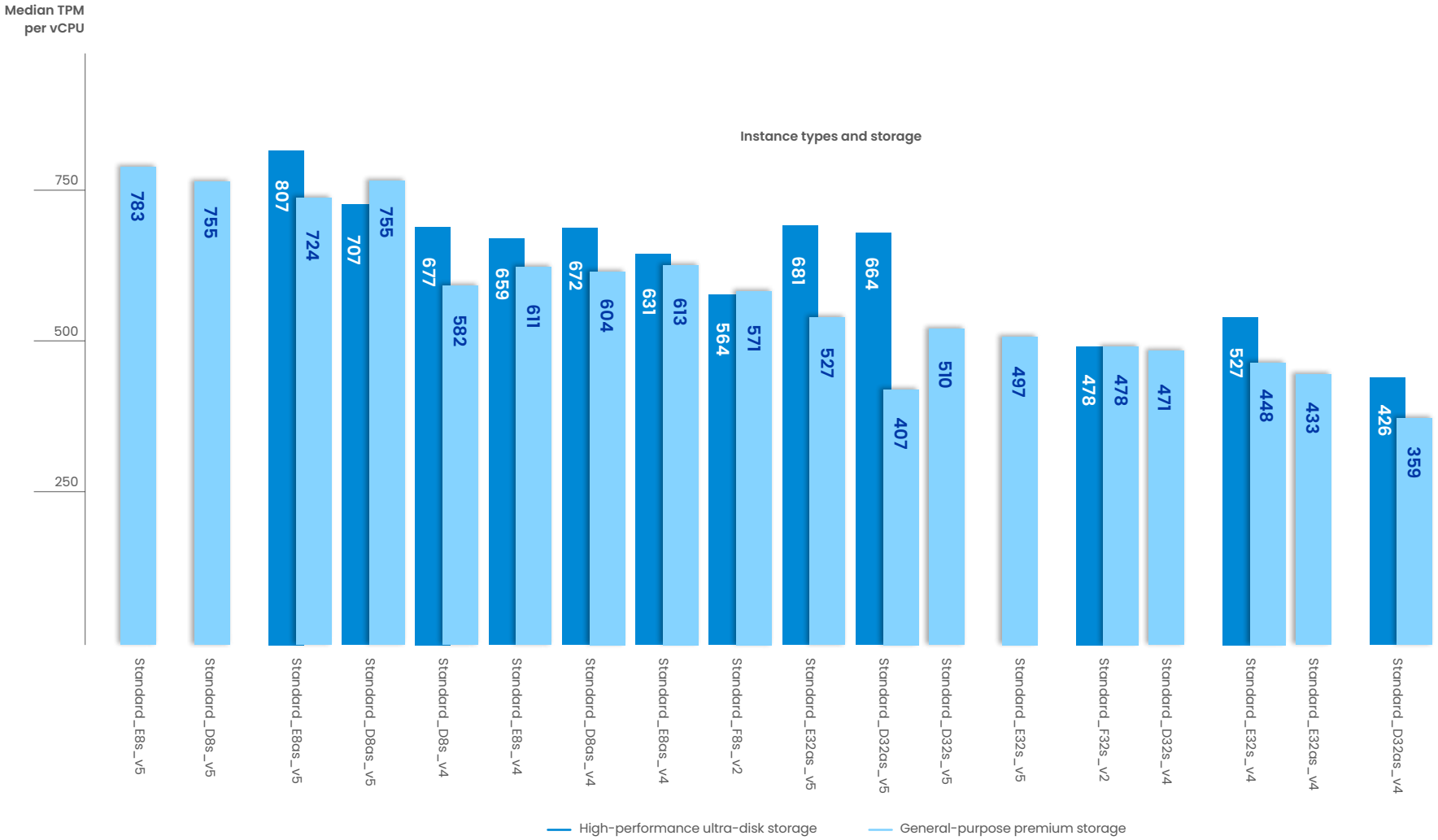


Azure results

OLTP benchmark

Median TPM per vCPU with storage 1,200 warehouses

Among smaller instance types, the general-purpose premium-disk storage performed as well if not better than ultra-disk in many scenarios.
For larger instance types, it may be worth considering ultra-disk depending on your use case.

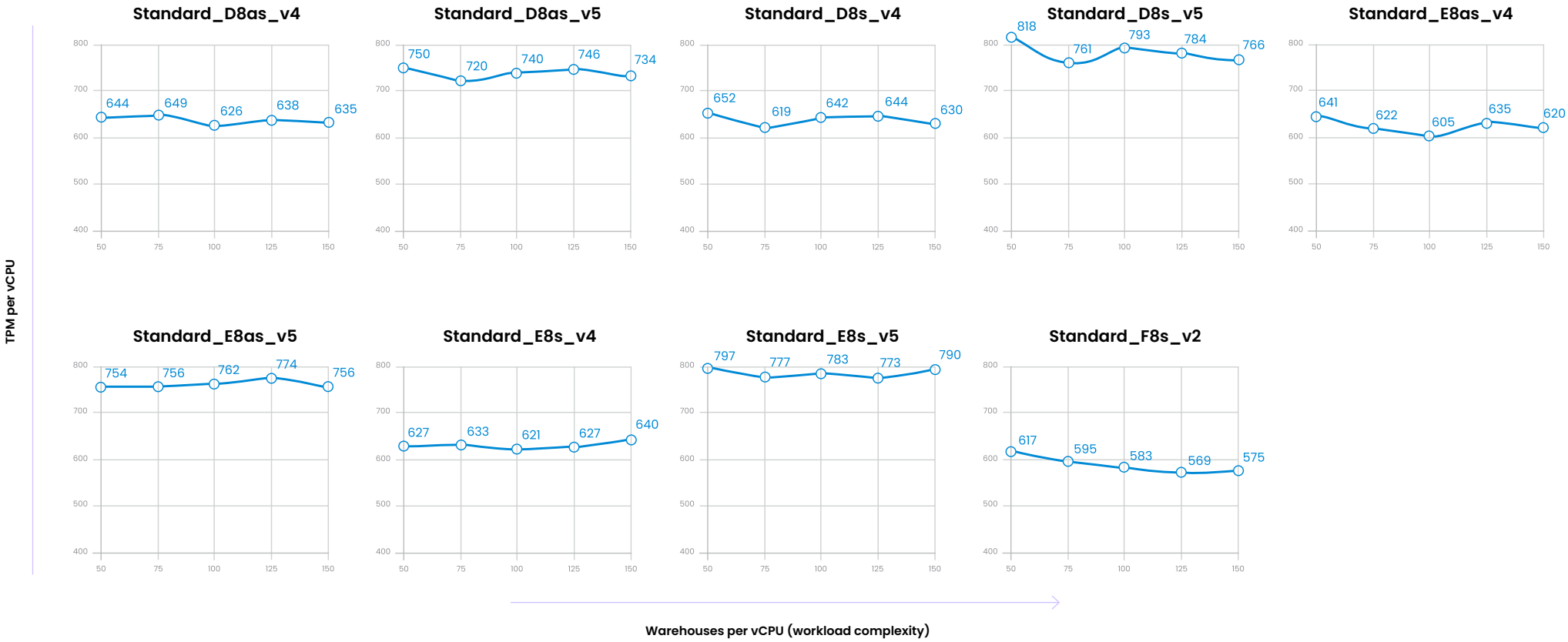


Azure results

These charts compare the per-vCPU TPM of Azure's small and large instance types over a range of workload complexities. In both cases, the latest-generation v5 instances won out, although among small instance types it was Intel-based instances that took the top spot, whereas among large instance types, the v5 AMD instances performed best. Surprisingly, some of the large v4 Intel Cascade Lake instances were very competitive with the v5 Intel Ice Lake instances.

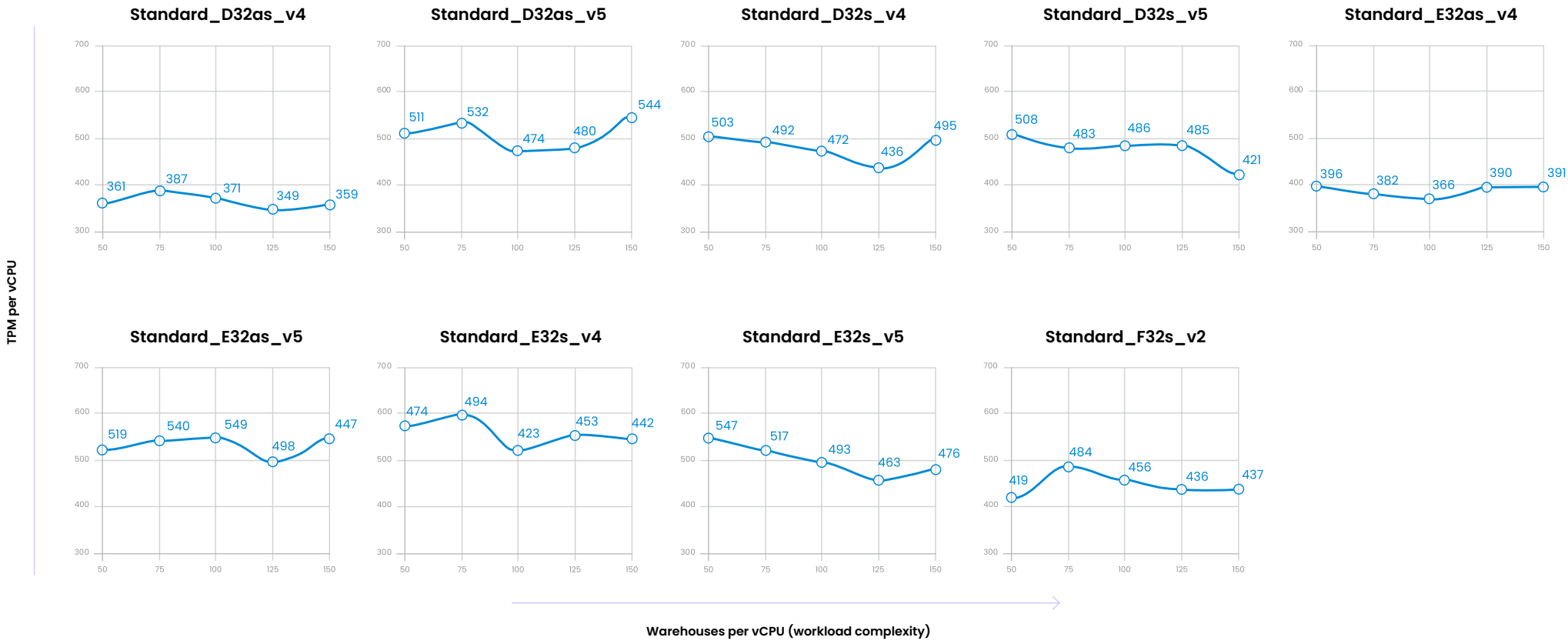
Network benchmark

Small instance TPM per vCPU, increasing workload complexity



Network benchmark

Large instance TPM per vCPU, increasing workload complexity

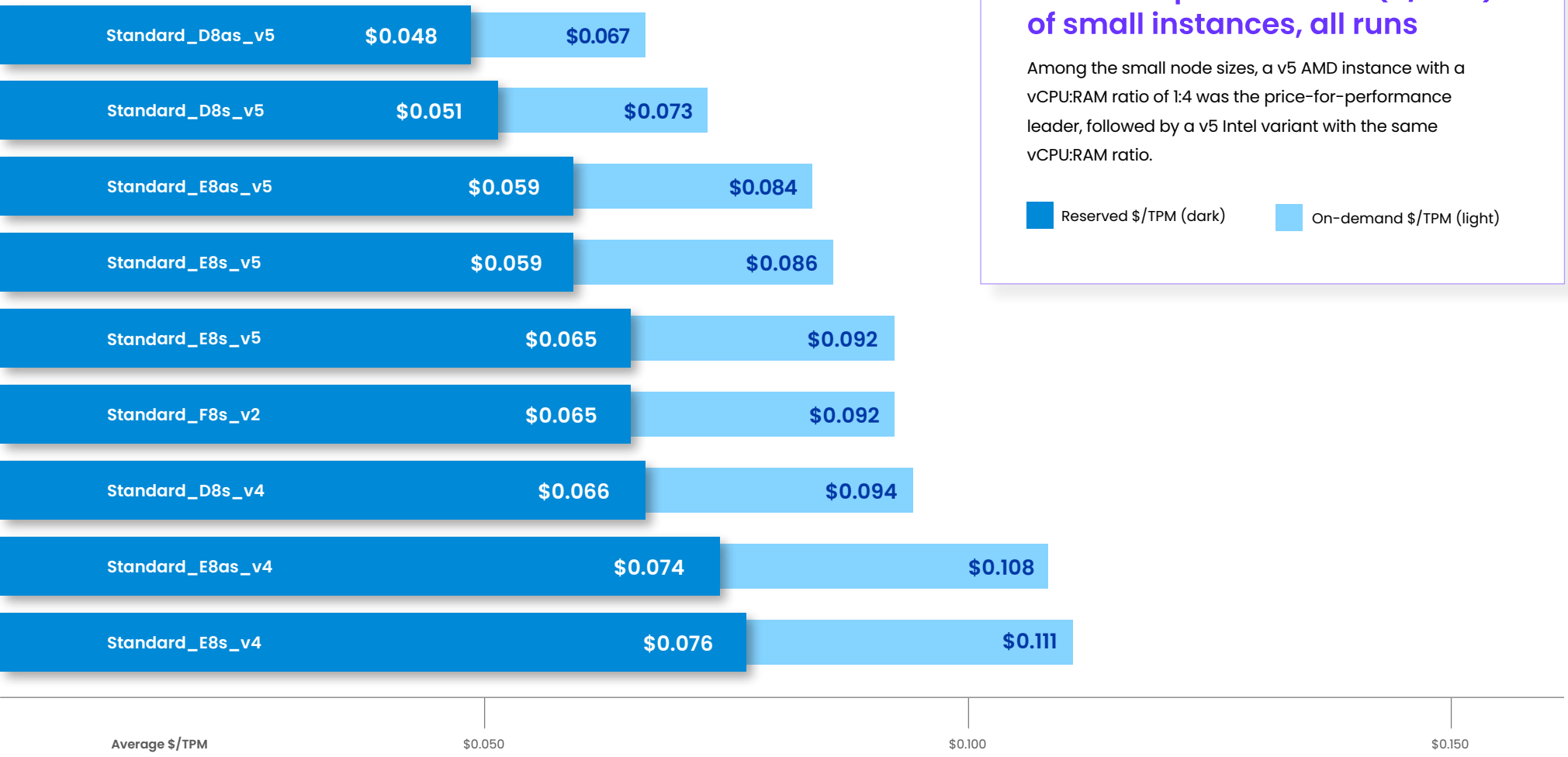


Azure results

OLTP benchmark

Price-for-performance ($/TPM) of small instances, all runs

Among the small node sizes, a v5 AMD instance with a vCPU:RAM ratio of 1:4 was the price-for-performance leader, followed by a v5 Intel variant with the same vCPU:RAM ratio.



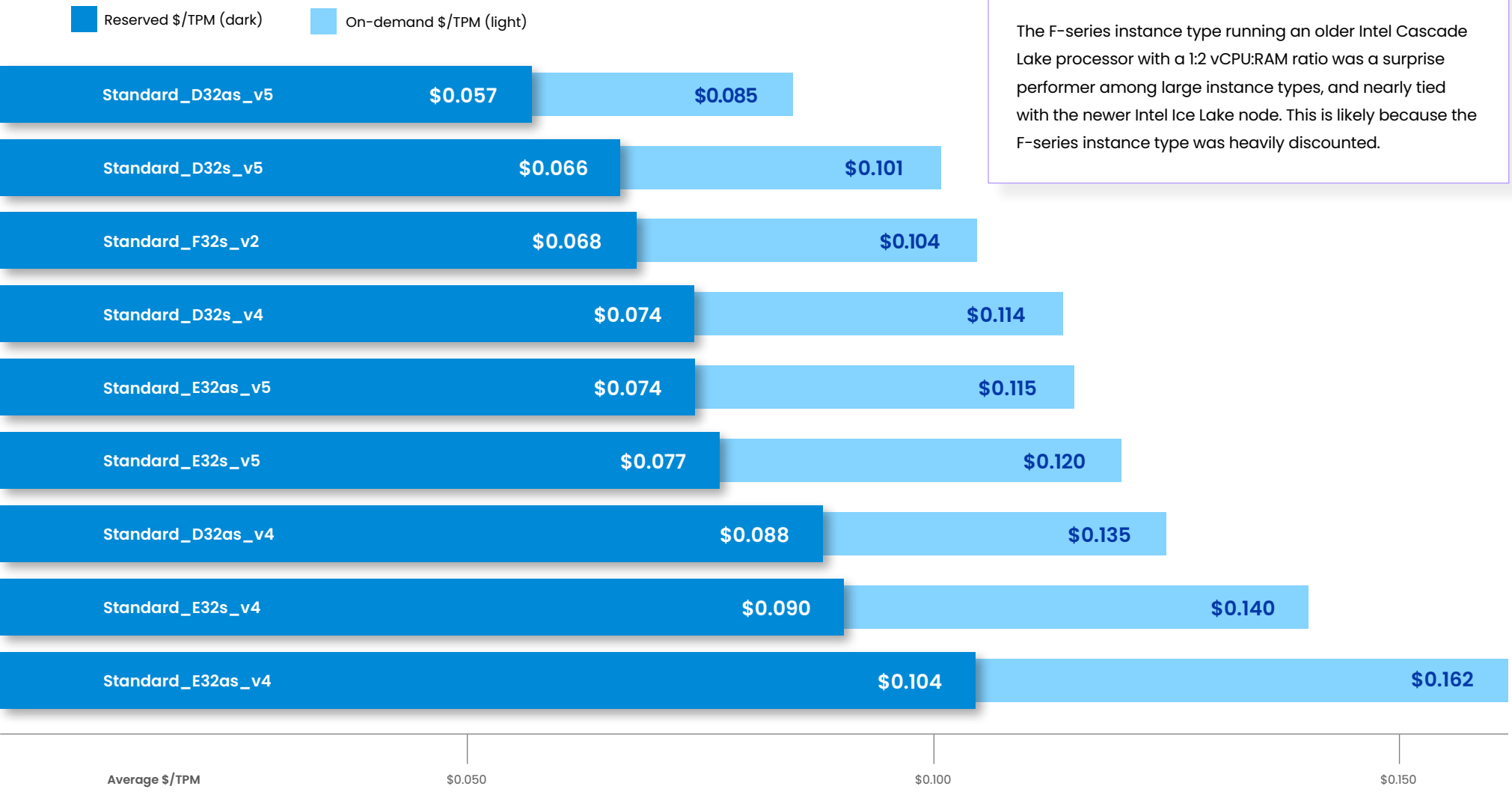
Azure results

OLTP benchmark

Price-for-performance ($/TPM) of large instances, all runs

Among large instance types, a v5 AMD instance with a 1:4 vCPU:RAM ratio again topped the heap for both reserved and on-demand pricing, and was again followed by a v5 Intel variant with the same vCPU:RAM ratio.

The F-series instance type running an older Intel Cascade Lake processor with a 1:2 vCPU:RAM ratio was a surprise performer among large instance types, and nearly tied with the newer Intel Ice Lake node. This is likely because the F-series instance type was heavily discounted.



CPU benchmarks

Methodology

We evaluated each cloud's CPU performance using the CoreMark version 1.0 benchmark.

The CoreMark benchmark can be limited to run on a single vCPU, or it can execute workloads on multiple vCPUs in parallel. We ran CoreMark in both modes ten times and recorded the average number of iterations/second for both the single-core and multi-core results.

Here, we have reported only multi-core results, as we expect the multi-core benchmark to be more representative of real-world performance. In an effort to compare multi-core performance across different instance sizes, we derived a per-vCPU score for each multi-core run. While this does not correlate to the single-core score for a variety of reasons, it allows us to compare multi-core runs where we don't have the same vCPU counts.

About CoreMark

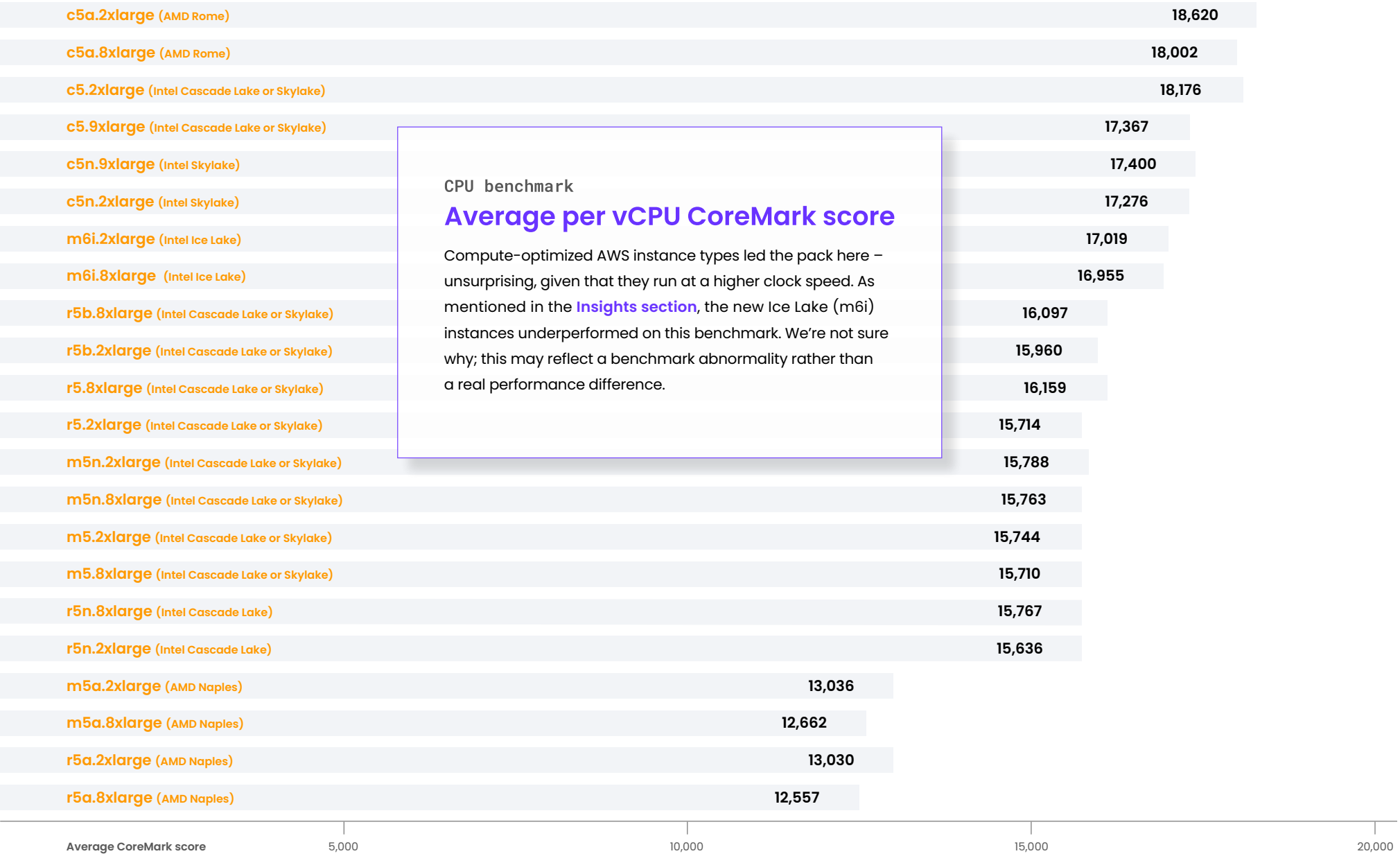
CoreMark is an open source, cloud-agnostic benchmark that evaluates general CPU performance. It executes various workloads that are representative of the types of operations often found in real world applications, including list management, list sort and search, matrix manipulation, and checksum computation. The benchmark then reports the number of operations performed, and produces a single-number score (the higher the score, the better the performance).

AWS results

CPU benchmark

Average per vCPU CoreMark score

Compute-optimized AWS instance types led the pack here – unsurprising, given that they run at a higher clock speed. As mentioned in the Insights section, the new Ice Lake (m6i) instances underperformed on this benchmark. We're not sure why; this may reflect a benchmark abnormality rather than a real performance difference.



GCP results

CPU benchmark

Average per vCPU CoreMark score

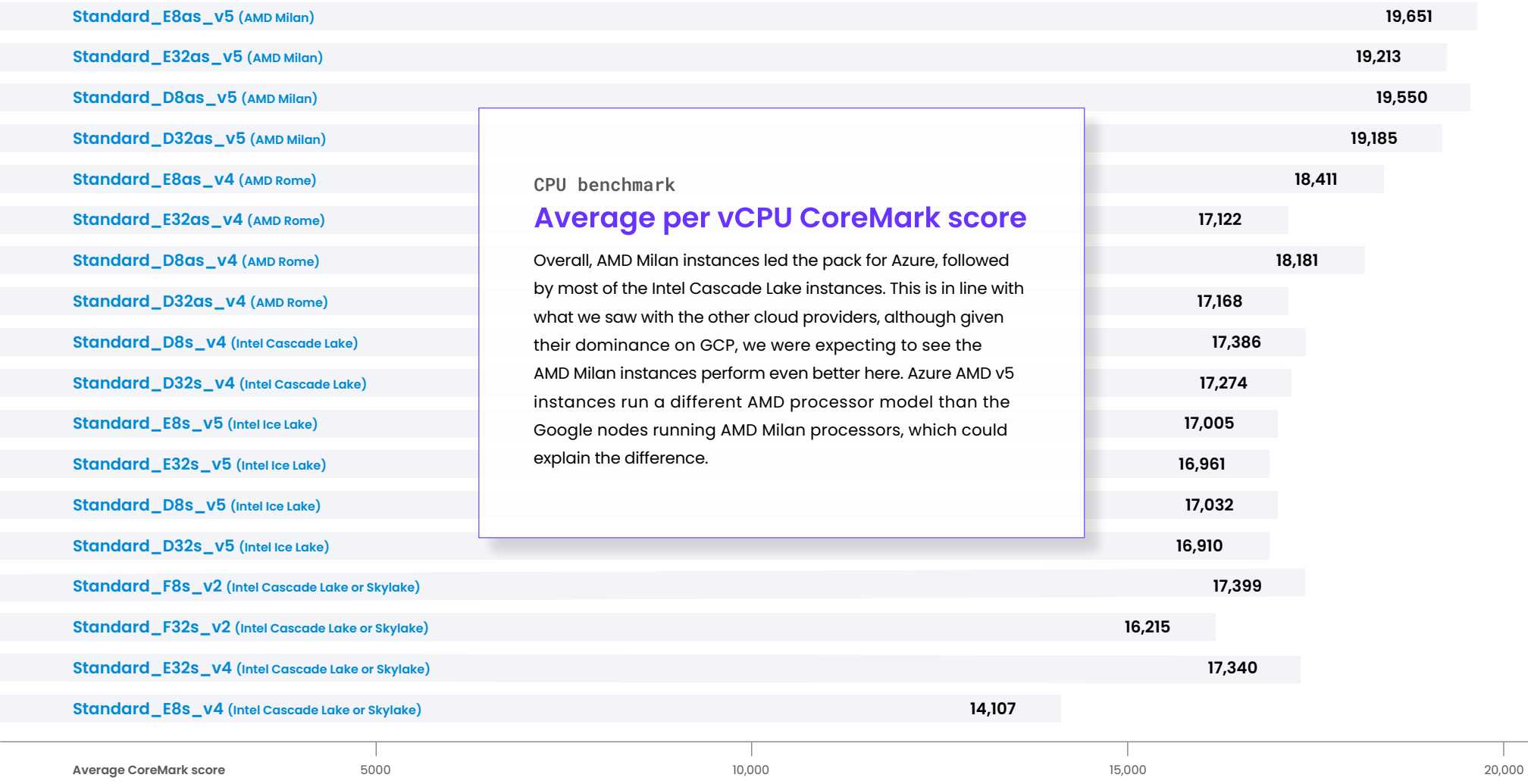
The latest generation AMD instances (t2d) excelled in this benchmark on a per-vCPU basis and overall.

Azure results

CPU benchmark

Average per vCPU CoreMark score

Overall, AMD Milan instances led the pack for Azure, followed by most of the Intel Cascade Lake instances. This is in line with what we saw with the other cloud providers, although given their dominance on GCP, we were expecting to see the AMD Milan instances perform even better here. Azure AMD v5 instances run a different AMD processor model than the Google nodes running AMD Milan processors, which could explain the difference.



Network benchmarks

Methodology

For this benchmark, we focused on node to node (single availability zone) and cross-region latency and throughput, since CockroachDB leverages these capabilities heavily when running in a production environment.

The network benchmark consists of two separate measurements: latency and throughput, implemented in netperf 2.7.1.

In this year's report, we have made numerous improvements[2] to this benchmark to suit the usage characteristics of CockroachDB more closely. These tests were run intra-AZ as well as inter-region, leveraging each cloud provider's Northern Virginia region as the client node and providers' Pacific Northwest US region as the server node.

Specifically, this year we made the following changes:

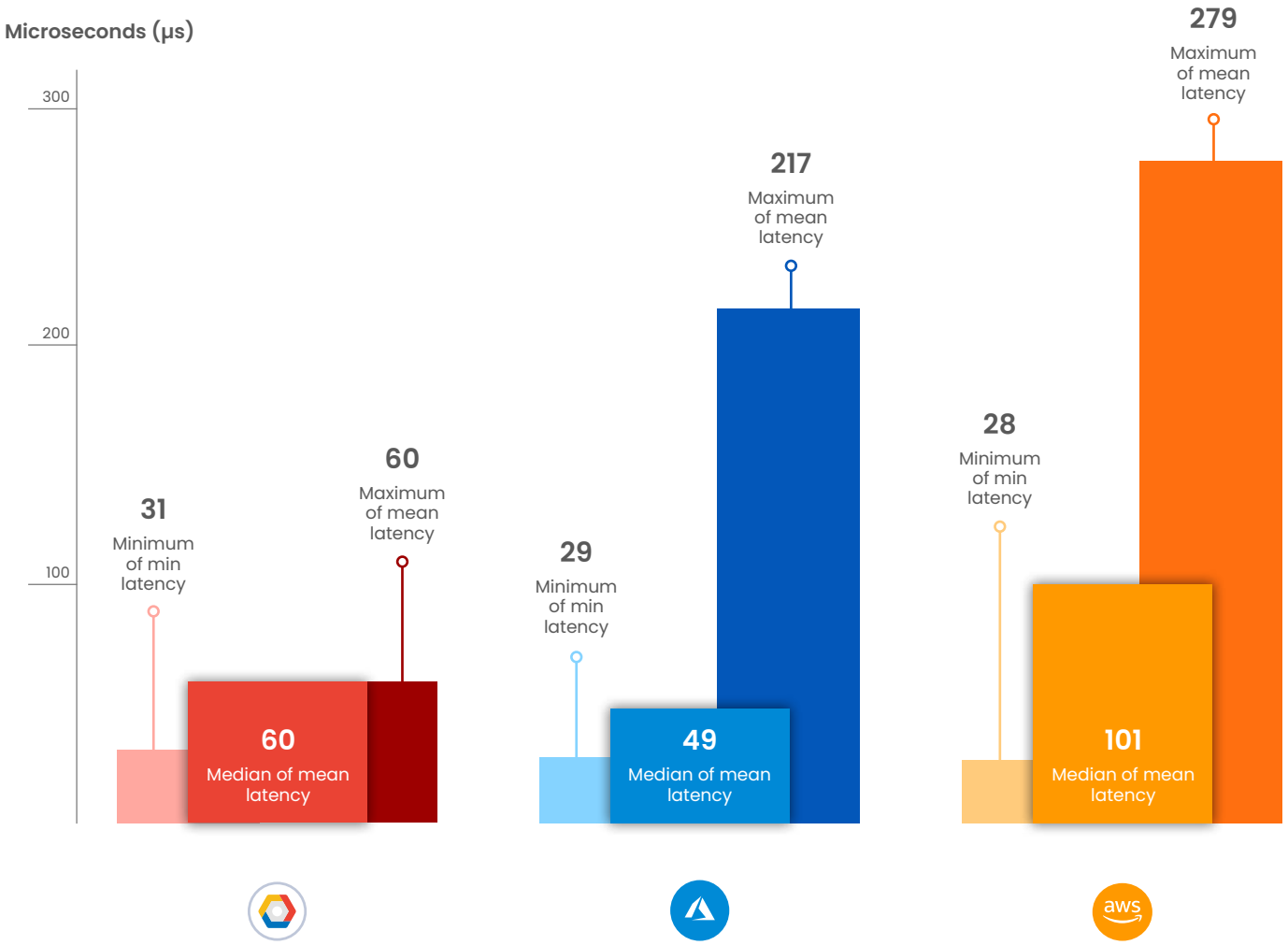
- Implemented a multi-stream method for the throughput test, with a fixed effective TCP socket buffer size of 16MB. This size is consistent with the maximum flow control window size allowed in gRPC, which is the primary node-to-node communication protocol of CockroachDB.
- Incremented the stream count stepwise until the aggregate throughput converged (i.e. stopped increasing) to determine the number of streams for the highest available throughput of a particular configuration.
- Increased the duration of the throughput test from 60 seconds to 12 minutes.
- Ran three sequential runs for each iteration and only collected the results from the median run in the throughput test to try to avoid skew and bursting errors associated with netperf 2.7.1.
- Extended the output of the throughput test to the min/max/avg throughput during the test duration, and also investigated time series plots to illustrate throughput fluctuation.

To assist with measuring cross-region latency, we used publicly-available distances from Dulles, Virginia to Portland, Oregon and Seattle, Washington and entered that into **https://wintelguy.com/wanlat.html** to calculate the theoretical latency floor for our cross-region network tests. We did not change any defaults in the calculator.

[2]In the 2021 Cloud Report, we ran vanilla Netperf TCP_RR and TCP_STREAM tests between two nodes of the same machine type from the same availability zone of the same cloud provider, each test for 60 seconds. We extended the runtime and tuned some parameters to ensure more consistent results in this year's testing.

Latency

There was no significant difference in latency for any of the instance types we tested across all three clouds. Since the variation between runs was greater than the gross latency of our best available runs, we see this as a statistical three-way tie.



Network benchmark

Node to node (Intra-AZ) latency

In our intra-AZ latency testing, the highest overall latency was 279 microseconds (µs), or 0.279 milliseconds, across all of the clouds, and the average latency was 70 µs. The lowest overall latency for each cloud was between 27 and 29 µs. This was without optimizing our nodes into placement groups, which could have pushed the average numbers even lower.

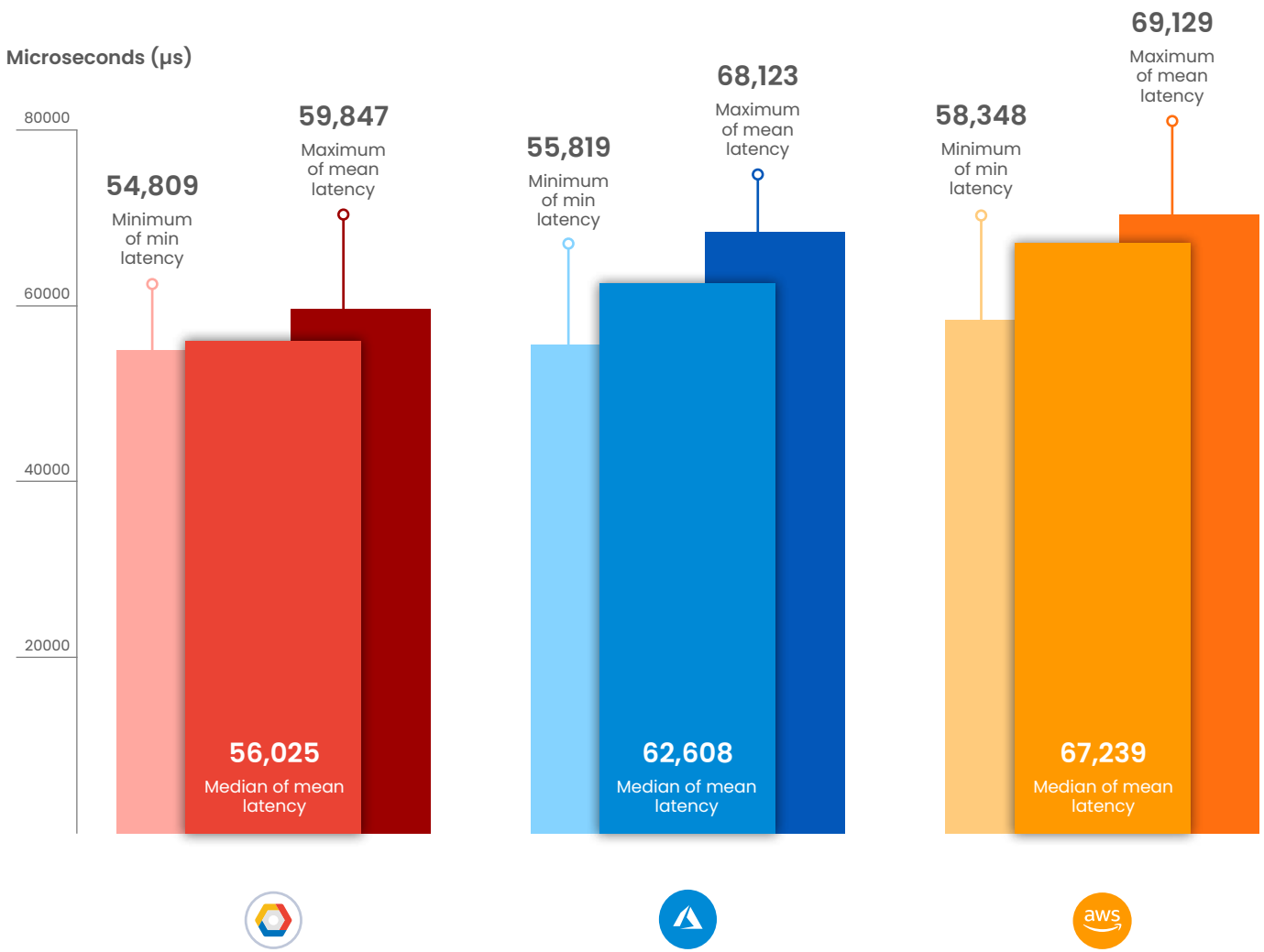
It's also important to note that since we did not set up placement groups to control node placement within an AZ, the variation in performance between nodes could be a measure of physical distance between the instances rather than the provisioned infrastructure.

Ultimately, outside of a true high-performance computing use-case, we don't expect that intra-AZ latency is likely to be an issue on any of the clouds.

Cross-region latency

In our cross-region testing, we measured latency between each provider's Northern Virginia and Pacific northwest regions. Using a **WAN Latency Estimator**, we calculated the best possible latency between Northern Virginia and Portland, Oregon (GCP) at 54 milliseconds (ms), and between Northern Virginia and Seattle, Washington (Azure and AWS) at 53 ms.

Overall, the minimum and average latencies were pretty tightly clustered. Despite being at a slight distance disadvantage, GCP performed best overall with a ~56 ms average latency. Azure had an average latency of ~62 ms, while AWS had an average latency of ~67 ms. (These results hold even when we remove outliers).

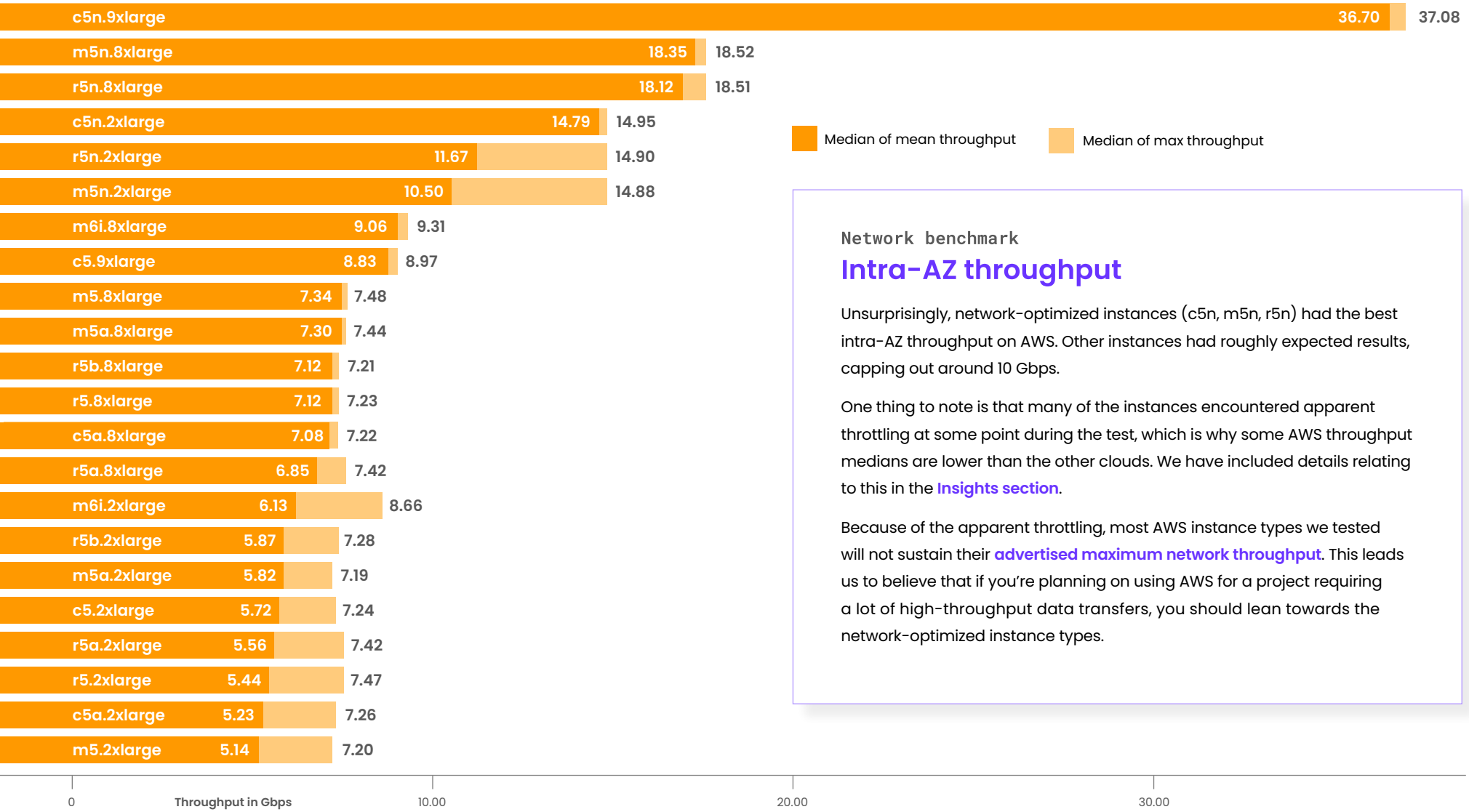


Throughput

Unlike the latency numbers, which were extremely consistent across all the different instance types we tested, throughput varied quite a bit by machine type, so we will break it down on a per-cloud basis.

For the throughput tests, we present both the median result of the mean throughput observed per instance type, and the median result of the max throughput observed per instance type to demonstrate the extent to which throughput varied during runs.

AWS results



Network benchmark

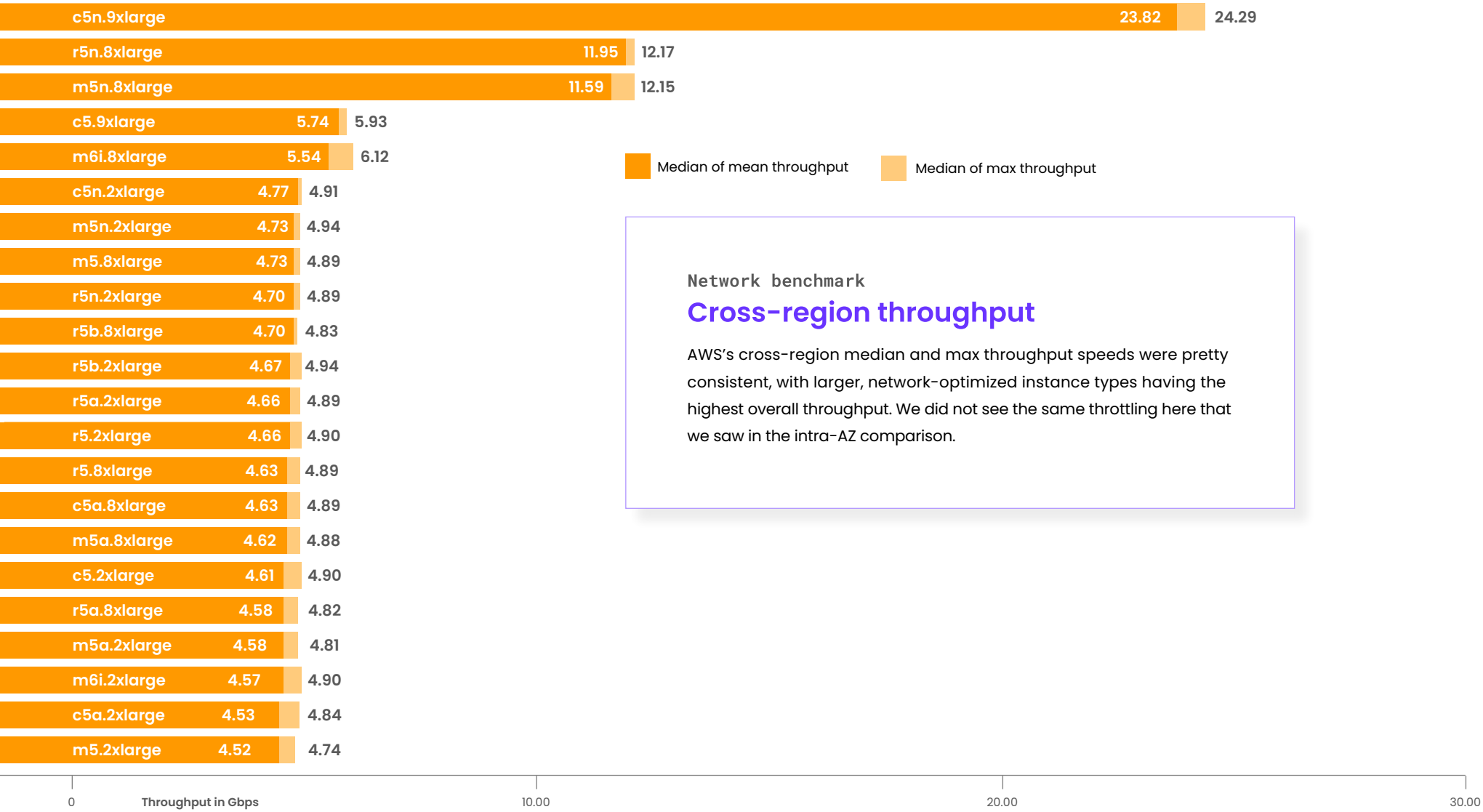
Intra-AZ throughput

Unsurprisingly, network-optimized instances (c5n, m5n, r5n) had the best intra-AZ throughput on AWS. Other instances had roughly expected results, capping out around 10 Gbps.

One thing to note is that many of the instances encountered apparent throttling at some point during the test, which is why some AWS throughput medians are lower than the other clouds. We have included details relating to this in the **Insights section**.

Because of the apparent throttling, most AWS instance types we tested will not sustain their **advertised maximum network throughput**. This leads us to believe that if you're planning on using AWS for a project requiring a lot of high-throughput data transfers, you should lean towards the network-optimized instance types.

AWS results



Network benchmark

Cross-region throughput

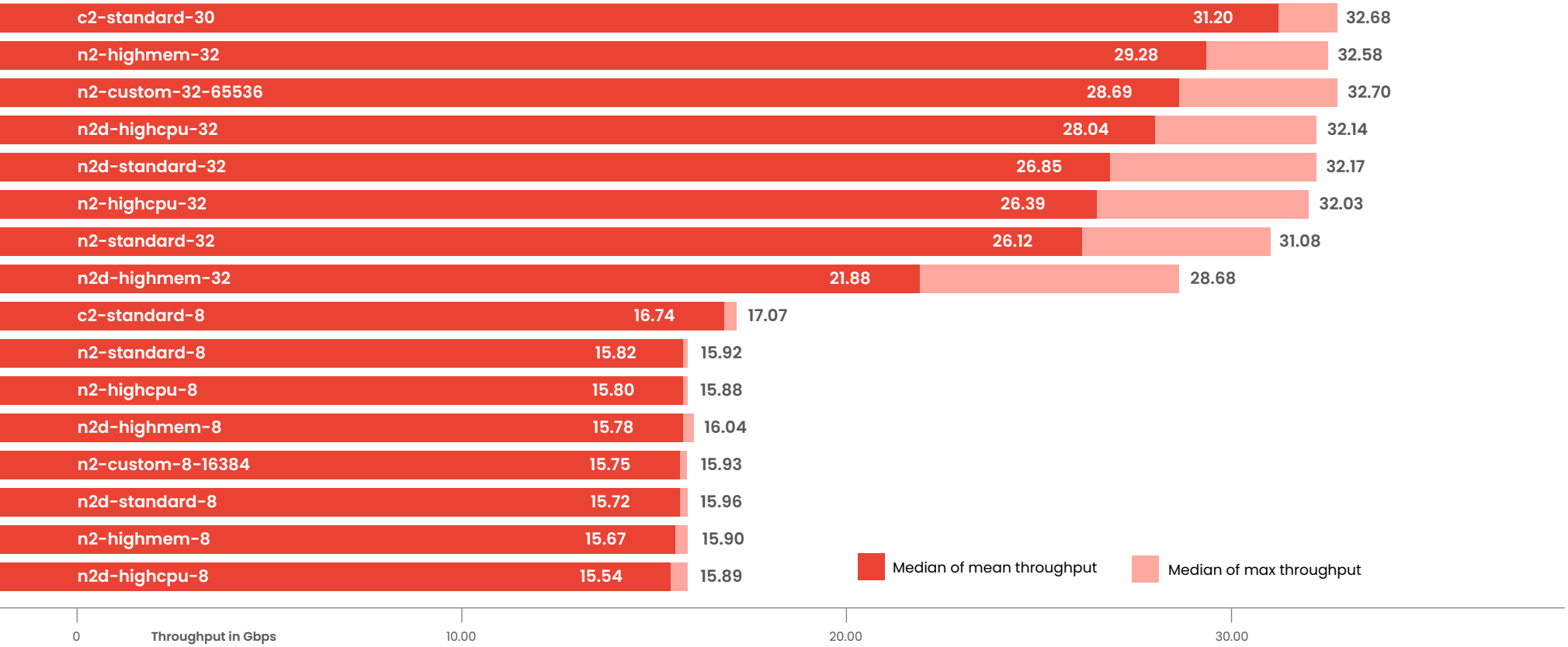
AWS's cross-region median and max throughput speeds were pretty consistent, with larger, network-optimized instance types having the highest overall throughput. We did not see the same throttling here that we saw in the intra-AZ comparison.

GCP results

Network benchmark

Intra-AZ throughput

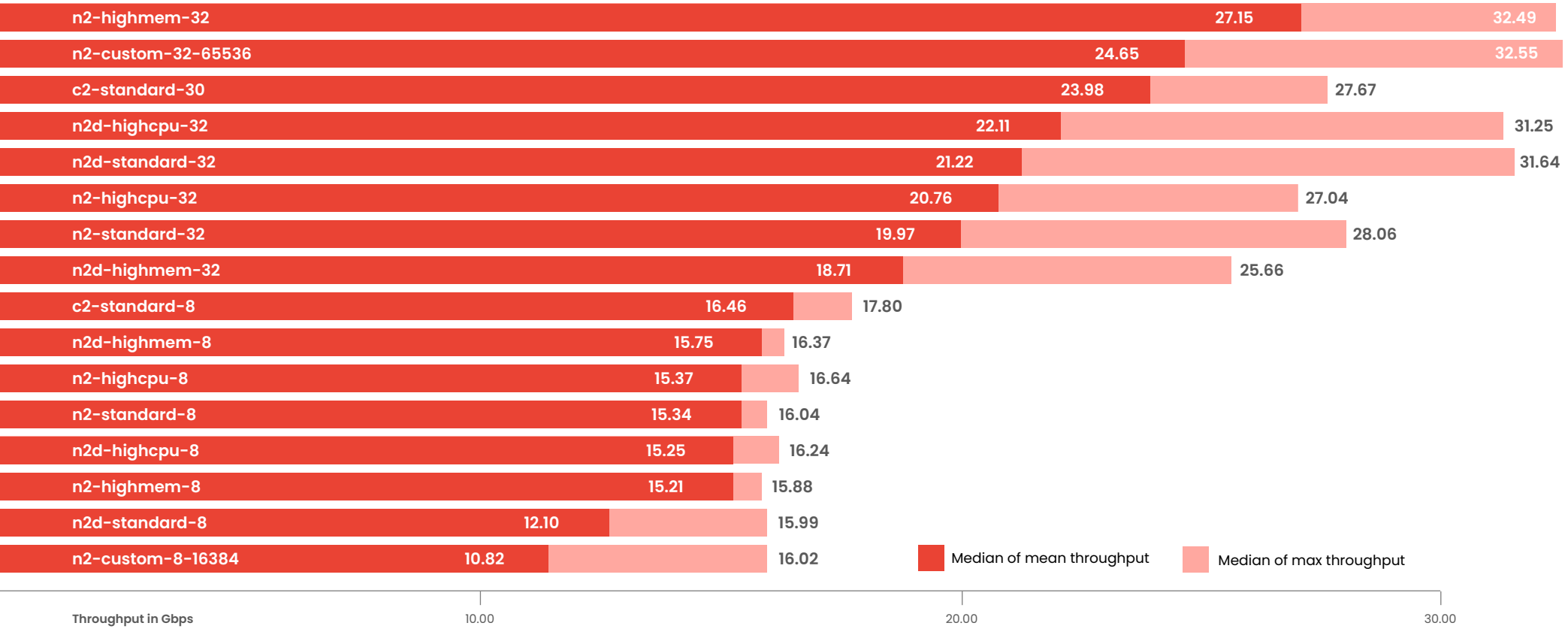
All of GCP's average throughput numbers were roughly in line with their **advertised throughput caps**. However, many of the instance types showed performance bursts that significantly exceeded the throughput numbers GCP advertises.



Network benchmark

Cross-region throughput

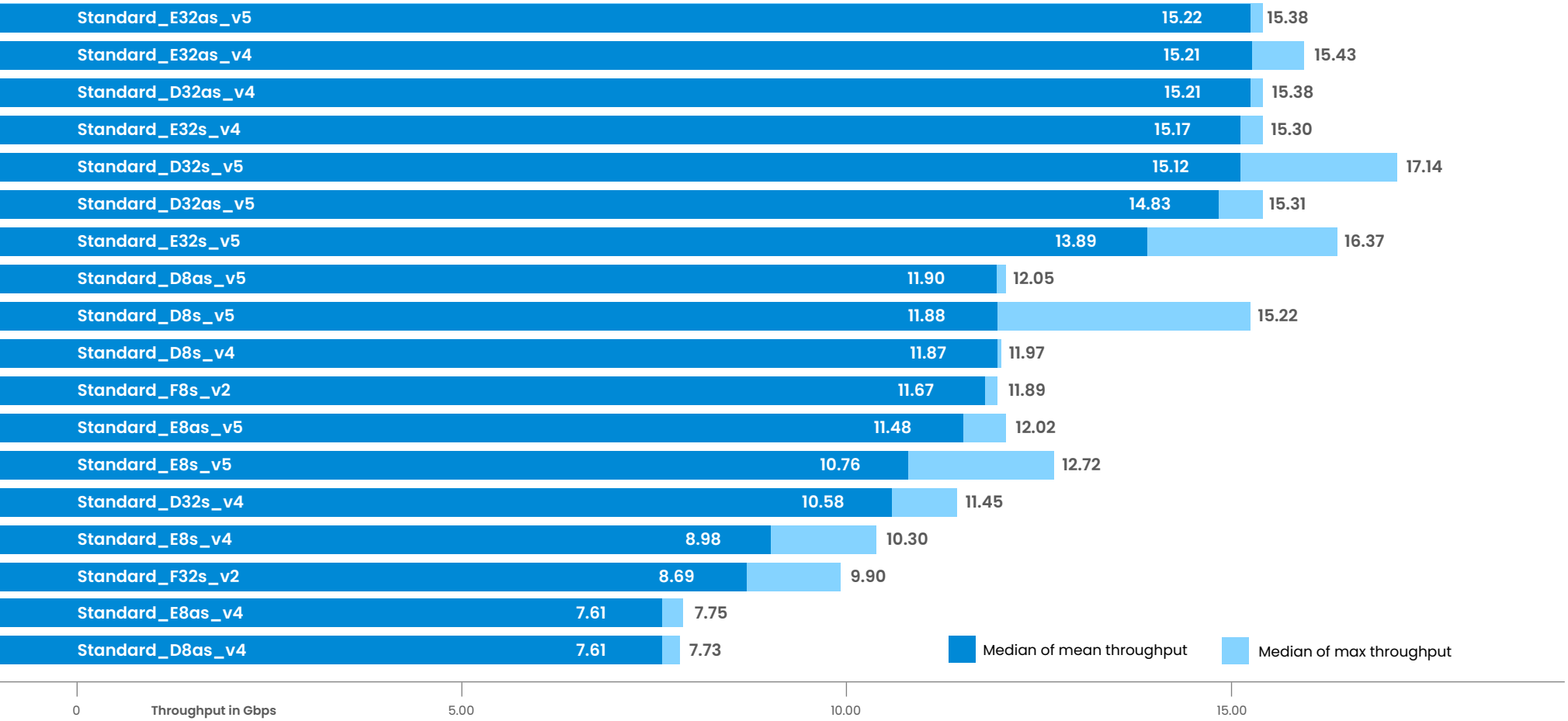
GCP's cross-region throughput was extremely consistent with the intra-AZ testing, with no evidence of mid-run throttling.



Network benchmark

Intra-AZ throughput

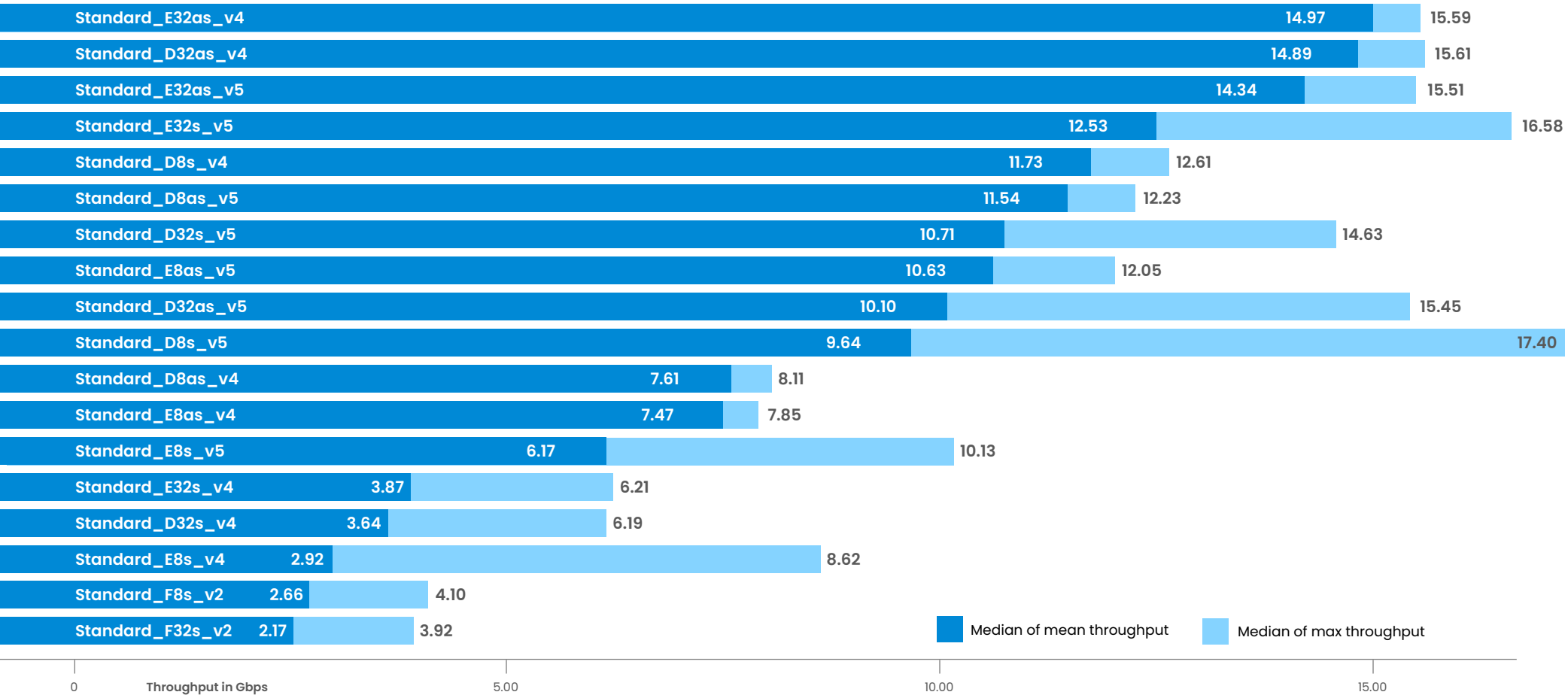
All of the Azure instance types performed as expected, and some of the new v5 instances had single-run bursts way above their advertised throughput speeds.



Network benchmark

Cross-region throughput

Azure's cross-region performance was less consistent than the other providers, but it still roughly hit its promised performance in most cases. There were also some single-run bursts that scored well above advertised throughput speeds.



Storage benchmarks

Methodology

We tested each cloud's storage I/O performance using FIO (Flexible I/O tester) version 3.1, an open-source benchmarking suite designed for measuring storage I/O performance. FIO allowed us to isolate storage performance characteristics by running workloads against the storage volume itself, as opposed to a file system.

Overall, we assessed each cloud's storage I/O performance on durable network-attached storage by analyzing the following metrics:

- Input/output Operations Per Second (IOPS): number of operations per second.
- Throughput (also known as bandwidth): the quantity of data sent and received over a time period.
- Fsync I/O latency: since CockroachDB is a durable application that executes fsync at the end of each write transaction, we added some tests allowing us to measure the write latency impact of fsyncs.

We looked at general-purpose volumes (gp3 for AWS, premium-disk for Azure, and pd-ssd for GCP) as well as high-performance volumes (io2 for AWS, ultra-disk for Azure, and pd-extreme for GCP).

For the smaller instance types we provisioned 1.0 TB volumes, and for the larger instances we provisioned 2.5 TB volumes. This allowed us sufficient space to run identical tests across all platforms, while also accommodating storage platforms where the provisioned IOPS scale per GiB of storage. CockroachDB requires a minimum of 500 IOPS/vCPU for general usage, and can sometimes leverage more than that depending on the workload and specific processor. We wanted to be certain that each instance type had sufficient IOPS to not be I/O bound.

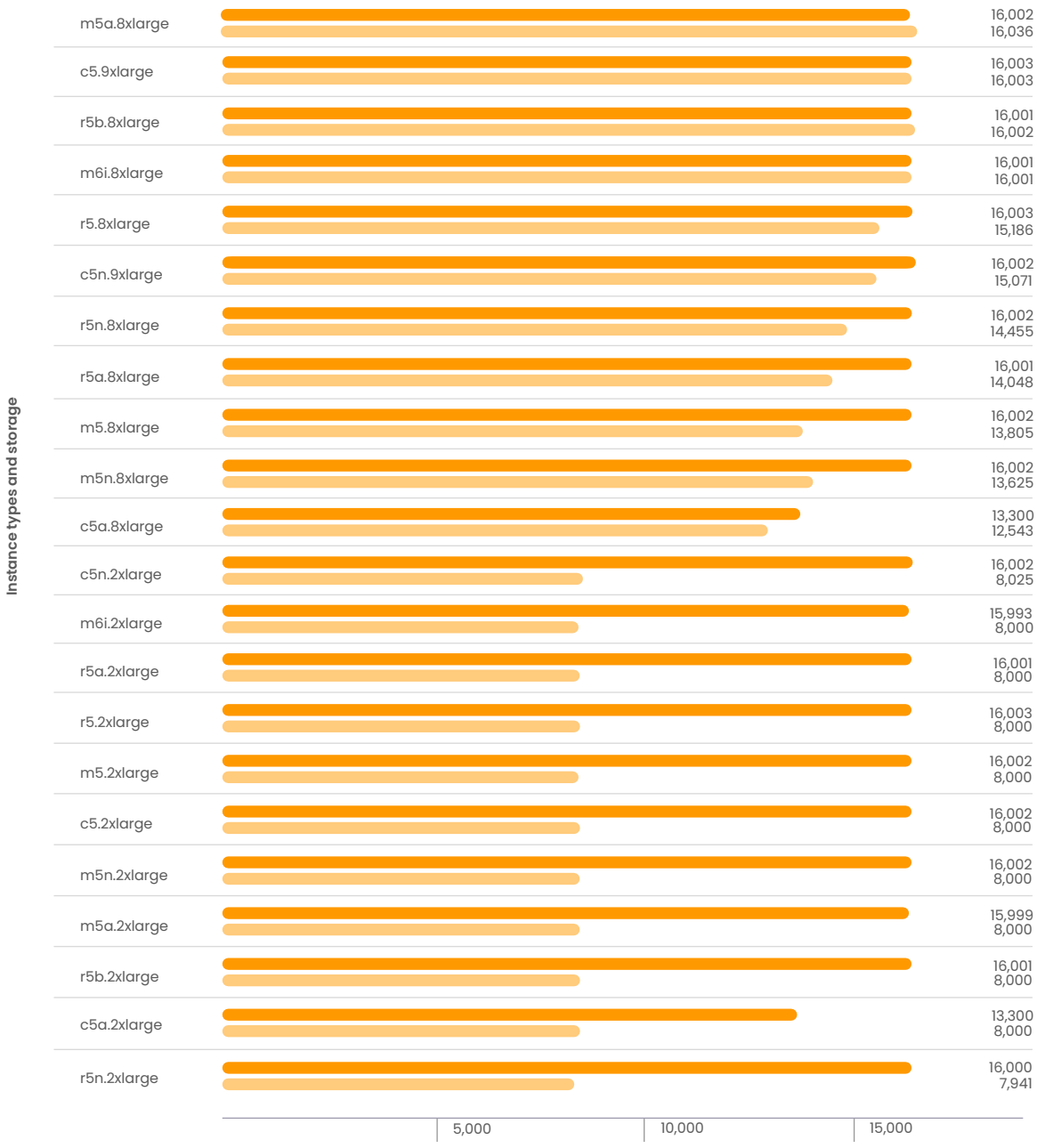
Unlike in past years, we did not test local SSD storage options, as each cloud provider now offers multiple performant persistent-disk options, and for a database we have a strong bias towards durability. Our usage of network-attached storage volumes in CockroachDB Dedicated and Serverless served as our primary motivation for making them the focus of this report.

We have also added some additional testing to better simulate the way CockroachDB interacts with storage – largely, by working with a variety of block sizes and adding some tests where fsyncs are called to determine the performance impact of an fsync on writes and overall IOPS.

AWS results

Understanding the results

Note: General-purpose gp3 storage volumes have 8,000 provisioned IOPS for small instance types and 16,000 provisioned IOPS for large instance types. The high-performance io2 volumes do not have provisioned IOPS.



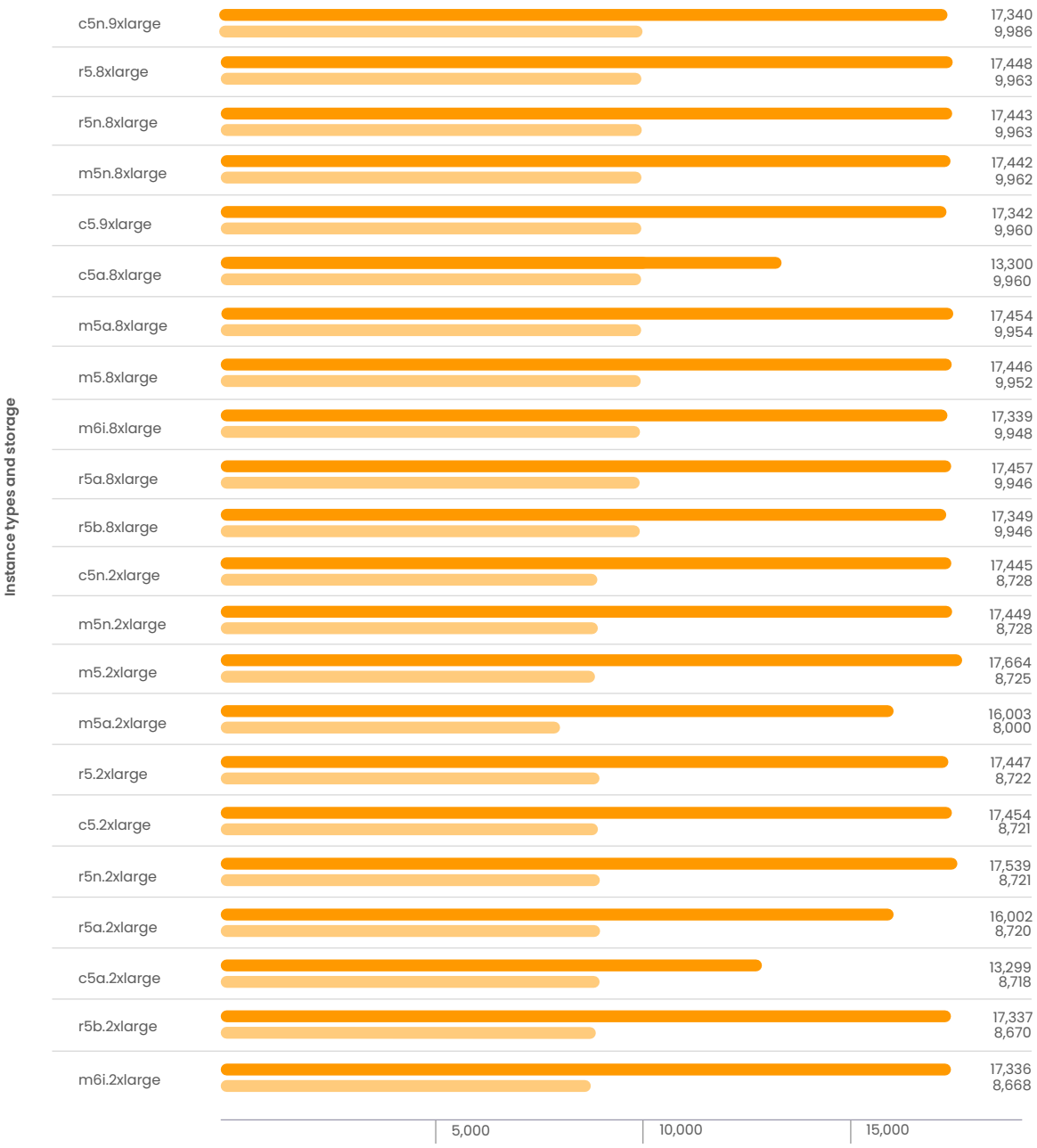
Storage benchmark

Write IOPS with 4k block size and fsync

Looking at write IOPS with a 4k block size and an fsync every 512 KB demonstrated how fsyncs might impact overall write performance. Overall, we got what was provisioned, but we did see some slowdowns on some of the larger nodes using the general-purpose storage.

High-performance io2 volumes General-purpose gp3 volumes

AWS results



Storage benchmark

Read IOPS with OLTP distribution block size

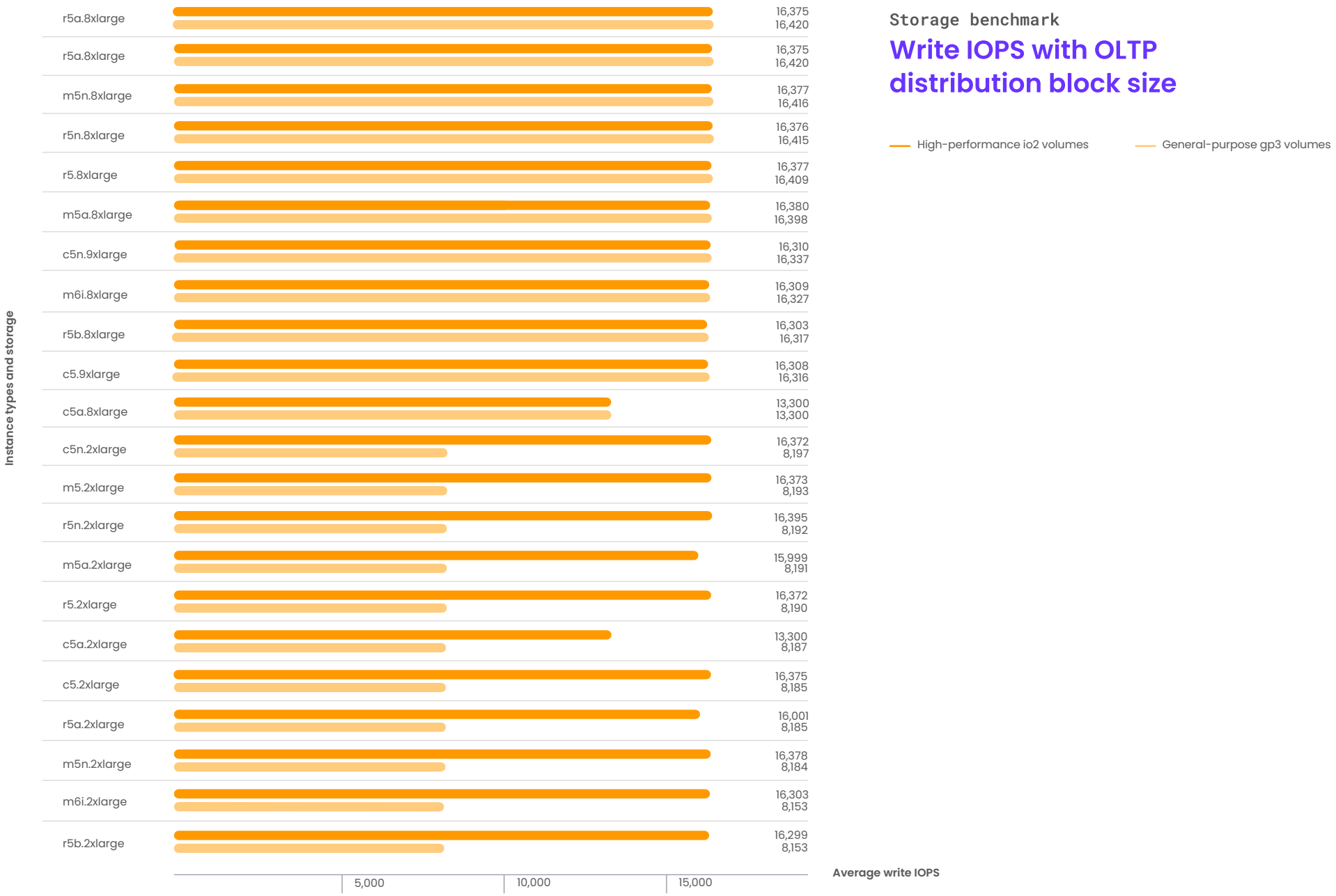
We tested read and write IOPS with a dynamic block size based on the Cockroach Labs Derivative TPC-C nowait workload to demonstrate the optimal read and write I/O performance we think we could see in the OLTP benchmark.

High-performance io2 volumes General-purpose gp3 volumes

AWS results

Storage benchmark

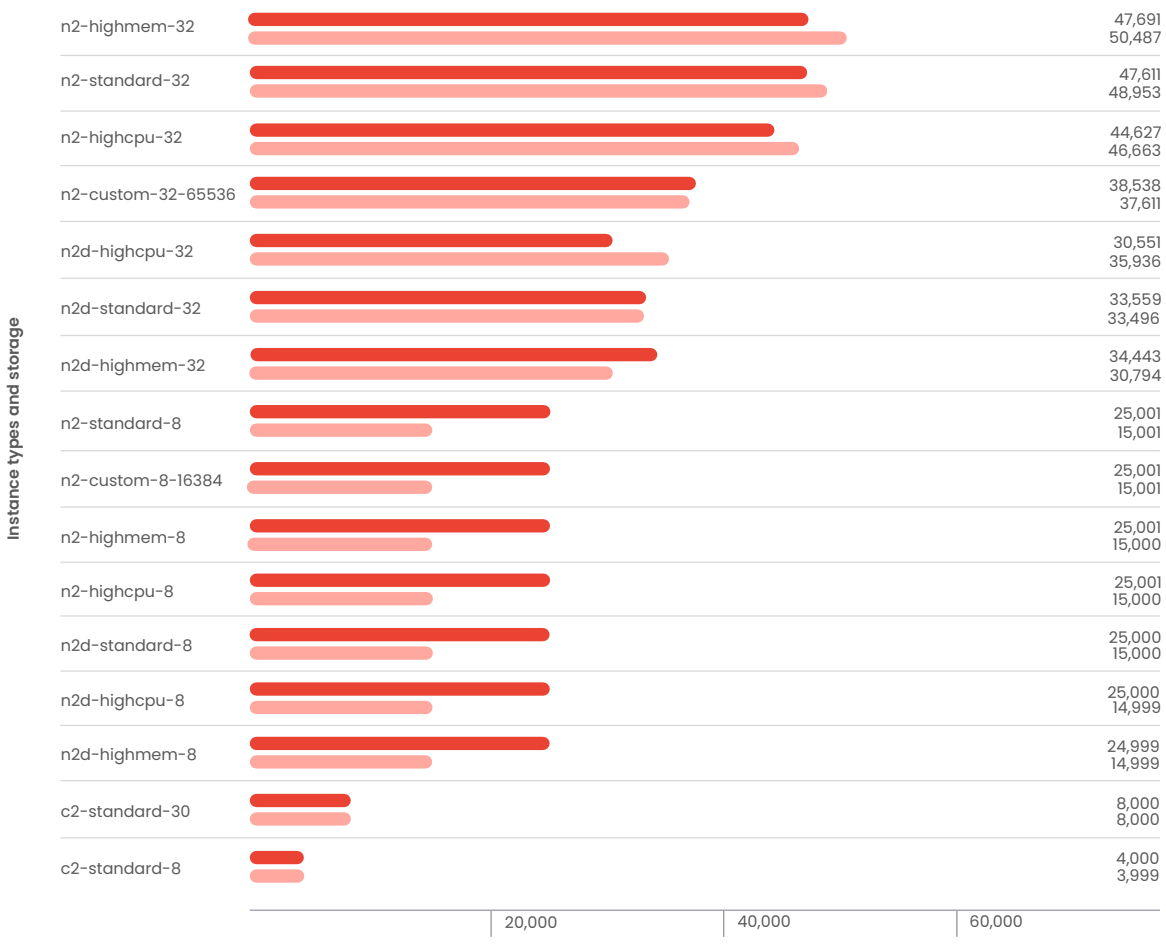
Write IOPS with OLTP distribution block size



GCP results

Understanding the results

GCP scales the available IOPS by the size of the instance.



Storage benchmark

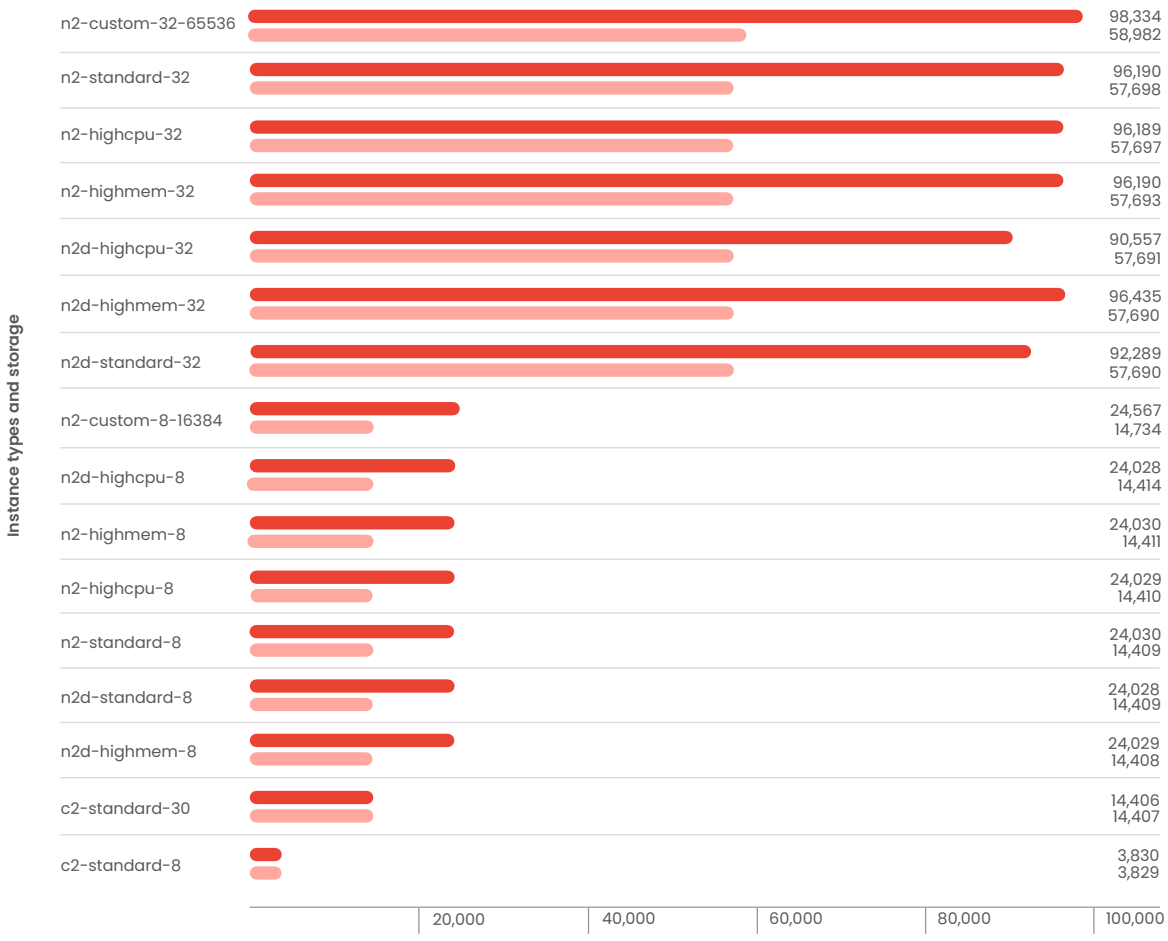
Write IOPS with 4k block size and fsync

Here, we see write IOPS with a 4k block size and an fsync every 512 KB, demonstrating how fsyncs might impact overall write performance. The pd-extreme storage outperformed pd-ssd in general, but we hit instance caps at the top end for all the volumes (64+ vCPUs are required to get optimal performance from GCP volumes), and there were even a few cases where the cheaper pd-ssd volumes outperformed the pd-extreme volumes.

— High-performance pd-extreme volumes

— General-purpose pd-ssd volumes

GCP results



Storage benchmark

Read IOPS with OLTP distribution block size

We tested read and write IOPS with a dynamic block size based on the Cockroach Labs Derivative TPC-C nowait workload to demonstrate the optimal read and write I/O performance we think we could see in the OLTP benchmark. The pd-extreme volumes outperformed pd-ssd across the board.

— High-performance pd-extreme volumes

— General-purpose pd-ssd volumes

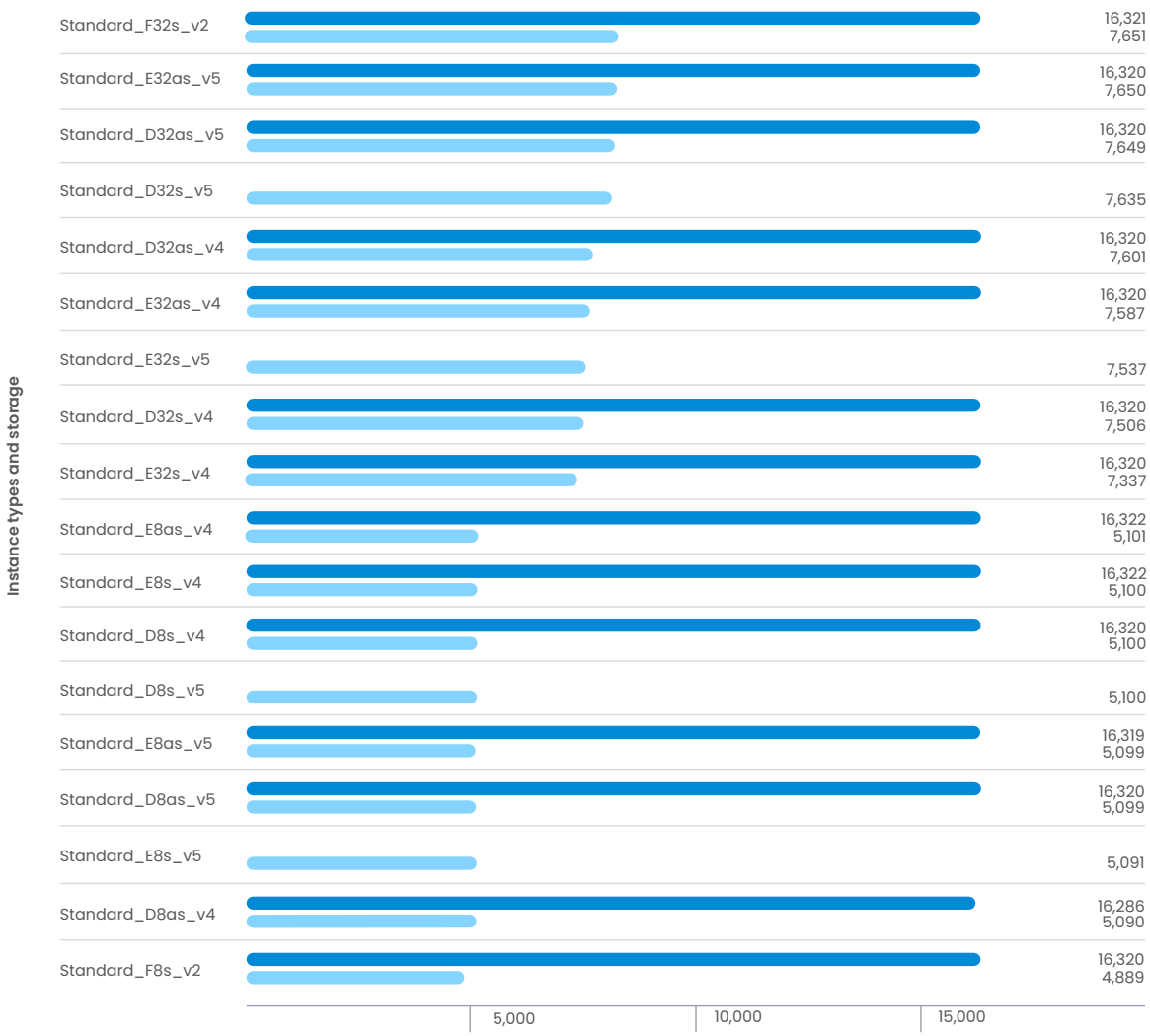
GCP results

Storage benchmark

Write IOPS with OLTP distribution block size

Understanding the results

There are a couple of instance types missing ultra-disk runs. This is because we were unable to get ultra-disk quota for those instance types when we were testing.

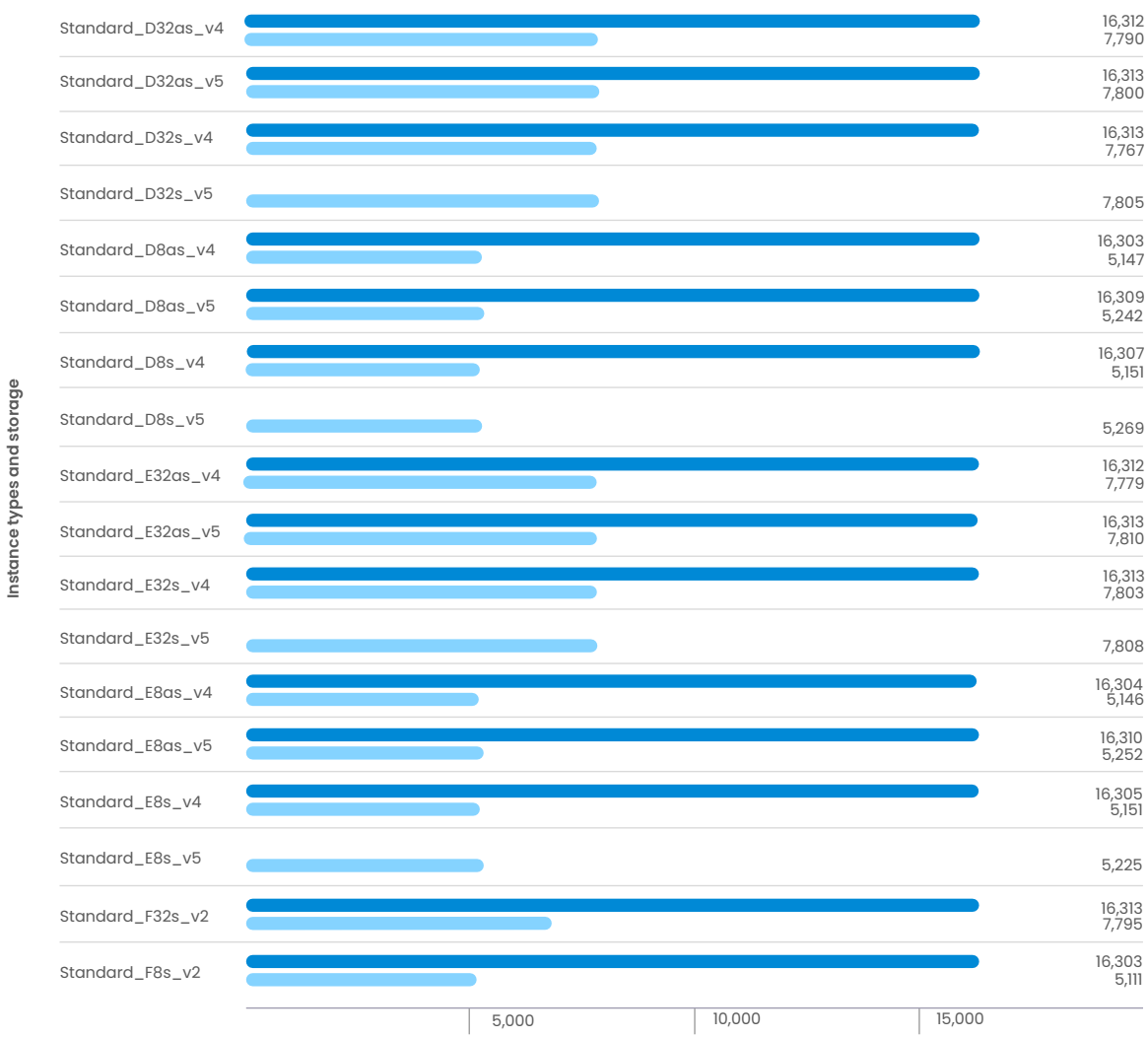


Storage benchmark

Write IOPS with 4k block size and fsync

Here we see write IOPS with a 4k block size and an fsync every 512 KB, which we think demonstrates how fsyncs might impact overall write performance. Ultra-disk outperformed premium-disk across the board.

High-performance volumes General-purpose volumes



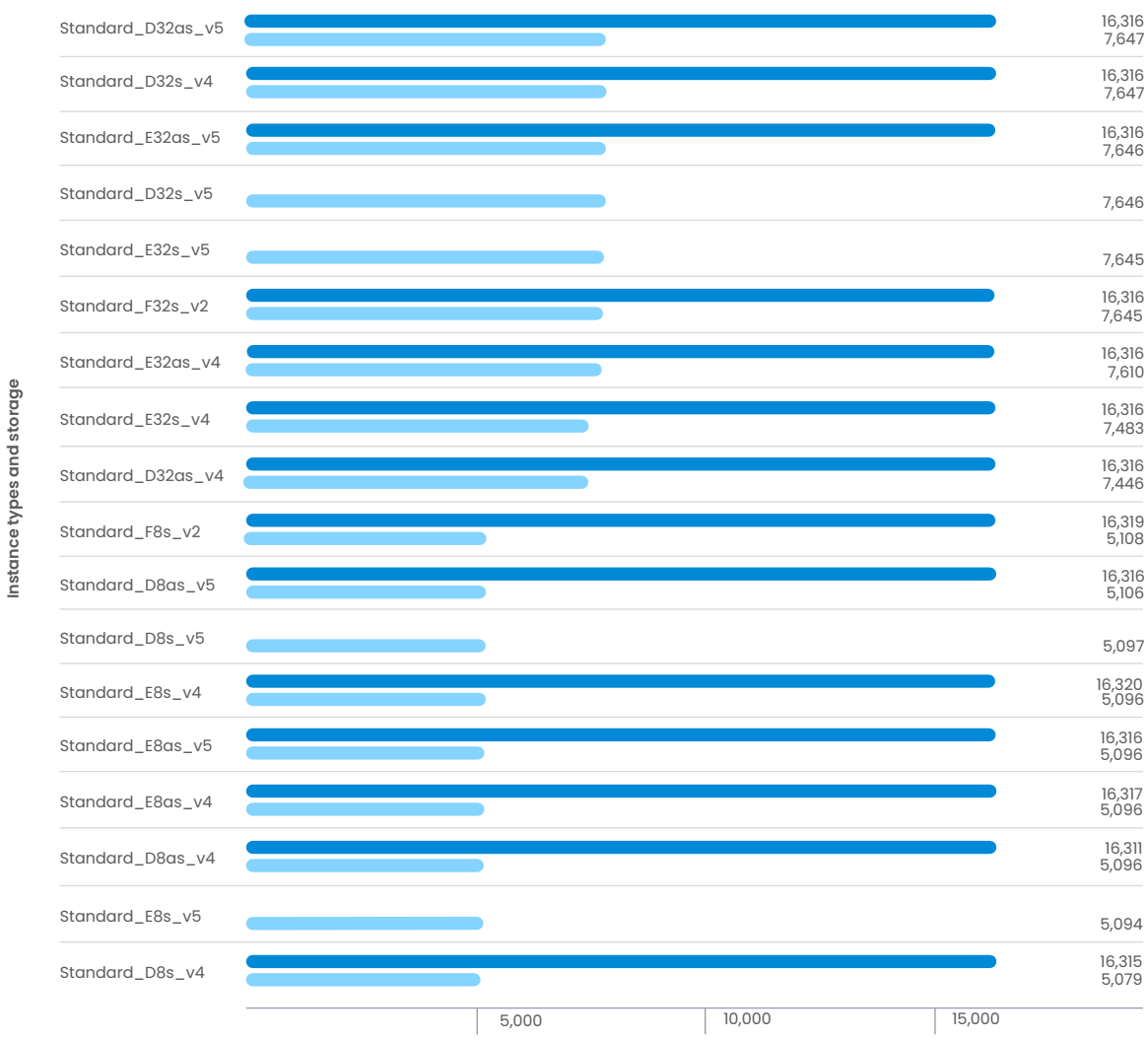
Storage benchmark

Read IOPS with OLTP distribution block size

We tested read and write IOPS with a dynamic block size based on the Cockroach Labs Derivative TPC-C nowait workload to demonstrate the optimal read and write I/O performance we think we could see in the OLTP benchmark. Again, ultra-disk volumes outperformed premium-disk across the board.

High-performance volumes General-purpose volumes

Azure results



Storage benchmark

Write IOPS with OLTP distribution block size

High-performance volumes

General-purpose volumes

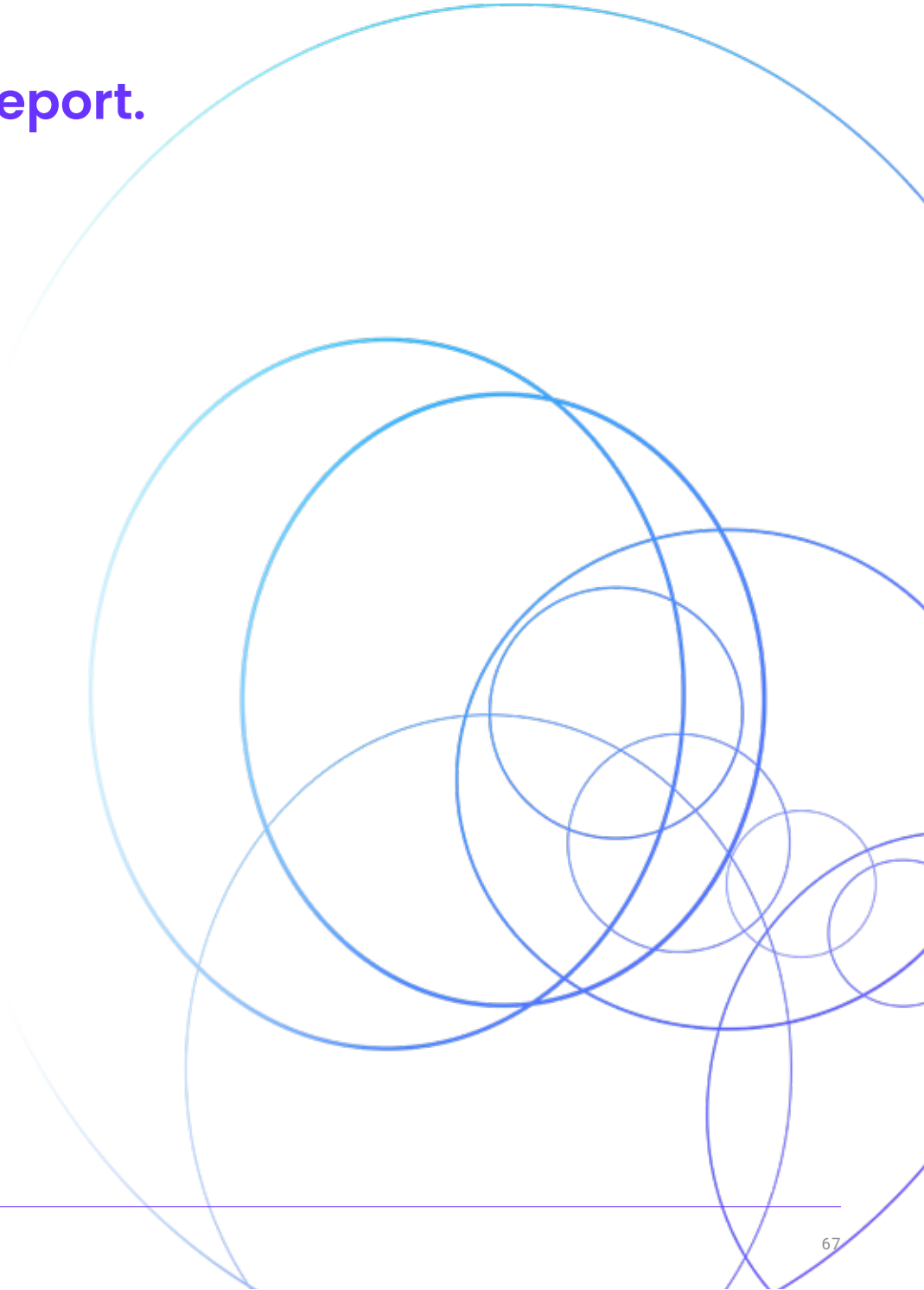
This year, we are not declaring an overall winner of the Cloud Report.

Our testing shows that all three cloud providers offer price-competitive options.

All three clouds can also offer top tier performance for OLTP applications. In terms of per-vCPU TPM, GCP took the top spot, but all three clouds had at least one instance type in the top five, and multiple instance types in the top ten.

However, our testing did reveal important considerations for customers assessing their cloud and instance type options, including:

- It is generally not worth paying for high-performance storage unless your workload specifically requires it.
- Storage and transfer represent a large portion of the total cost of operating a database in the cloud, and in some cases we found that cross-region transfers cost the same as intra-region transfers – getting the availability, latency, and survivability advantages of a multi-region database doesn't always come at a higher cost than single-region.
- For consistency across a variety of workload complexities, we recommend instance types with at least 4 GB of RAM per vCPU.
- Small instance types may outperform larger ones for OLTP applications.



About Cockroach Labs

Cockroach Labs is the creator of CockroachDB, the most highly evolved, cloud-native, distributed SQL database on the planet. Helping companies of all sizes — and the apps they develop — to scale fast, survive failures, and thrive everywhere. CockroachDB is in use at some of the world's largest enterprises across all industries, including some of the most recognized companies in banking, media & entertainment, retail, and technology. Headquartered in New York City, Cockroach Labs is backed by Altimeter, Benchmark, Greenoaks, GV, Firstmark, Index Ventures, Lone Pine, Redpoint Ventures, Sequoia Capital, Tiger Global, and Workbench.

CockroachDB is in use at some of the world's largest enterprises across all industries, including Bose, Comcast, Hard Rock Digital, and some of the largest companies in banking, retail, and media. Headquartered in New York City with offices in Toronto and San Francisco, Cockroach Labs is backed by Greenoaks, Altimeter, BOND, Benchmark, Coatue, FirstMark, GV, Firstmark, Index Ventures, J.P. Morgan, Lone Pine Capital, Redpoint Ventures, Sequoia Capital, Tiger Capital, and Workbench.

This project is open-source and you are welcome to run the benchmarks yourself.

Reproduction steps are fully open source, available in this repo:
https://github.com/cockroachlabs/cloud-report

You may just end up saving money, increasing performance, and learning something new. If you have any questions about the report, we encourage you to get in touch with us via our public Slack channel or learn more via our YouTube channel.

Cockroach Labs

2022 Cloud Report

Appendix

Appendix I: Instance types tested

For instance types that show multiple processor generations (in some tests, we received machines with different processors on different runs), the latest-generation CPU's info is provided.

AWS

Machine Type	vCPU Count	RAM (GB)	Storage Types Tested	Processor Type	Processor, Speed
c5.2xlarge	8	16	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8275CL 3.00GHz(3.90GHz)
c5.9xlarge	36	72	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8275CL 3.00GHz(3.90GHz)
c5a.2xlarge	8	16	ebs-gp3, ebs-io2	AMD Rome (EPYC Gen 2)	AMD EPYC 7R32 2.80Ghz(3.20GHz)
c5a.8xlarge	32	64	ebs-gp3, ebs-io2	AMD Rome (EPYC Gen 2)	AMD EPYC 7R32 2.80Ghz(3.20GHz)
c5n.2xlarge	8	16	ebs-gp3, ebs-io2	Intel Skylake	Intel Xeon Platinum 8124M 3.00GHz(3.50GHz)
c5n.9xlarge	36	72	ebs-gp3, ebs-io2	Intel Skylake	Intel Xeon Platinum 8124M 3.00GHz(3.50GHz)
m5.2xlarge	8	32	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
m5.8xlarge	32	128	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
m5a.2xlarge	8	32	ebs-gp3, ebs-io2	AMD Naples (EPYC Gen 1)	AMD EPYC 7571 2.20GHz(3.00GHz)
m5a.8xlarge	32	128	ebs-gp3	AMD Naples (EPYC Gen 1)	AMD EPYC 7571 2.20GHz(3.00GHz)
m5n.2xlarge	8	32	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz
m5n.8xlarge	32	128	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz
m6i.2xlarge	8	32	ebs-gp3, ebs-io2	Intel Ice Lake	Intel Xeon Platinum 8375C 2.90GHz(3.50GHz)
m6i.8xlarge	32	128	ebs-gp3, ebs-io2	Intel Ice Lake	Intel Xeon Platinum 8375C 2.90GHz(3.50GHz)
r5.2xlarge	8	64	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
r5.8xlarge	32	256	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
r5a.2xlarge	8	64	ebs-gp3, ebs-io2	AMD Naples (EPYC Gen 1)	AMD EPYC 7571 2.20GHz(3.00GHz)
r5a.8xlarge	32	256	ebs-gp3, ebs-io2	AMD Naples (EPYC Gen 1)	AMD EPYC 7571 2.20GHz(3.00GHz)
r5b.2xlarge	8	64	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
r5b.8xlarge	32	256	ebs-gp3, ebs-io2	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
r5n.2xlarge	8	64	ebs-gp3, ebs-io2	Intel Cascade Lake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)
r5n.8xlarge	32	256	ebs-gp3, ebs-io2	Intel Cascade Lake	Intel Xeon Platinum 8259CL 2.50GHz(3.50GHz)

Appendix I: Instance types tested

GCP

Machine Type	vCPU Count	RAM (GB)	Storage Types Tested	Processor Type	Processor, Speed
c2-standard-8	8	32	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 3.10GHz(3.80/3.90GHz)
c2-standard-30	30	120	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 3.10GHz(3.80/3.90GHz)
n2-custom-32-131072	32	64	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 2.80GHz(3.40/3.90GHz)
n2-custom-8-32768	8	16	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 2.80GHz(3.40/3.90GHz)
n2-highcpu-32	32	32	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 2.80GHz(3.40/3.90GHz)
n2-highcpu-8	8	8	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 2.80GHz(3.40/3.90GHz)
n2-highmem-32	32	256	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 2.80GHz(3.40/3.90GHz)
n2-highmem-8	8	64	pd-ssd, pd-extreme	Intel Cascade Lake	Intel Xeon 2.80GHz(3.40/3.90GHz)
n2-standard-32 (-icelake Intel Ice Lake)	32	128	pd-ssd, pd-extreme	Intel Cascade Lake or Intel Ice Lake	Intel Xeon 2.60GHz(3.10/3.40GHz)
n2-standard-8 (-icelake Intel Ice Lake)	8	32	pd-ssd, pd-extreme	Intel Cascade Lake or Intel Ice Lake	Intel Xeon 2.60GHz(3.10/3.40GHz)
n2d-highcpu-32	32	32	pd-ssd, pd-extreme	AMD Rome (EPYC Gen 2) or AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
n2d-highcpu-8	8	8	pd-ssd, pd-extreme	AMD Rome (EPYC Gen 2) or AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
n2d-highmem-32	32	256	pd-ssd, pd-extreme	AMD Rome (EPYC Gen 2) or AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
n2d-highmem-8	8	64	pd-ssd, pd-extreme	AMD Rome (EPYC Gen 2) or AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
n2d-standard-32	32	128	pd-ssd, pd-extreme	AMD Rome (EPYC Gen 2) or AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
n2d-standard-8	8	32	pd-ssd, pd-extreme	AMD Rome (EPYC Gen 2) or AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
t2d-standard-8	8	32	pd-ssd, pd-extreme	AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)
t2d-standard-32	32	128	pd-ssd, pd-extreme	AMD Milan (EPYC Gen 3)	AMD EPYC 7B13 2.45GHz(3.5GHz)

Appendix I: Instance types tested

Azure

Machine Type	vCPU Count	RAM (GB)	Storage Types Tested	Processor Type	Processor, Speed
Standard_D32as_v4	32	128	premium-disk, ultra-disk	AMD Rome (EPYC Gen 2)	AMD EPYC 7452 2.35GHz(3.35GHz)
Standard_D32as_v5	32	128	premium-disk, ultra-disk	AMD Milan (EPYC Gen 3)	AMD EPYC 7763 2.45GHz(3.5GHz)
Standard_D32s_v4	32	128	premium-disk, ultra-disk	Intel Cascade Lake	Intel Xeon Platinum 8272CL 2.60GHz(3.40/3.70GHz)
Standard_D32s_v5	32	128	premium-disk	Intel Ice Lake	Intel Xeon Platinum 8370C 2.80GHz(3.5GHz)
Standard_D8as_v4	8	32	premium-disk, ultra-disk	AMD Rome (EPYC Gen 2)	AMD EPYC 7452 2.35GHz(3.35GHz)
Standard_D8as_v5	8	32	premium-disk, ultra-disk	AMD Milan (EPYC Gen 3)	AMD EPYC 7763 2.45GHz(3.5GHz)
Standard_D8s_v4	8	32	premium-disk, ultra-disk	Intel Cascade Lake	Intel Xeon Platinum 8272CL 2.60GHz(3.40/3.70GHz)
Standard_D8s_v5	8	32	premium-disk	Intel Ice Lake	Intel Xeon Platinum 8370C 2.80GHz(3.5GHz)
Standard_E32as_v4	32	256	premium-disk, ultra-disk	AMD Rome (EPYC Gen 2)	AMD EPYC 7452 2.35GHz(3.35GHz)
Standard_E32as_v5	32	256	premium-disk, ultra-disk	AMD Milan (EPYC Gen 3)	AMD EPYC 7763 2.45GHz(3.5GHz)
Standard_E32s_v4	32	256	premium-disk, ultra-disk	Intel Cascade Lake	Intel Xeon Platinum 8272CL 2.60GHz(3.40/3.70GHz)
Standard_E32s_v5	32	256	premium-disk	Intel Ice Lake	Intel Xeon Platinum 8370C 2.80GHz(3.5GHz)
Standard_E8as_v4	8	64	premium-disk, ultra-disk	AMD Rome (EPYC Gen 2)	AMD EPYC 7452 2.35GHz(3.35GHz)
Standard_E8as_v5	8	64	premium-disk, ultra-disk	AMD Milan (EPYC Gen 3)	AMD EPYC 7763 2.45GHz(3.5GHz)
Standard_E8s_v4	8	64	premium-disk, ultra-disk	Intel Cascade Lake	Intel Xeon Platinum 8272CL 2.60GHz(3.40/3.70GHz)
Standard_E8s_v5	8	64	premium-disk	Intel Ice Lake	Intel Xeon Platinum 8370C 2.80GHz(3.5GHz)
Standard_F32s_v2	32	64	premium-disk, ultra-disk	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8272CL 2.60GHz(3.40/3.70GHz)
Standard_F8s_v2	8	16	premium-disk, ultra-disk	Intel Cascade Lake or Skylake	Intel Xeon Platinum 8272CL 2.60GHz(3.40/3.70GHz)

Appendix II: OLTP benchmark - small vs. large node investigation

To investigate the performance discrepancy between smaller and larger nodes, we performed experiments including:

- Looking at specific CPU/Instance information to control for bias.
- Sample runs looking at processor frequency (Turbo boost).
- Reconsidering some of the parameters of our scaled-up test.
- Designing a test that had the same vCPU count between the large and small nodes (sixteen 8 vCPU nodes compared against four 32 vCPU nodes at the same warehouse count).

We started by considering whether or not it was appropriate to use the same benchmark harness configuration for both the smaller and larger instance shapes. Our initial hypothesis was that we were artificially limiting the performance of the larger nodes. We did several tests changing individual load-generator parameters, and the only one that had a material impact on the performance of the database was changing the overall size of the connection pool at the load generator.

While our production guidance states that we recommend four active connections per vCPU, we found that we could run a slightly higher number of active connections per vCPU (either 5 or 6) in certain cases without pushing the database cluster into a potentially unstable state, and we saw a corresponding slight increase in overall throughput. However, some instance types performed worse than with the original parameters, and both the small and large instance sizes for a particular instance type seemed to benefit from tuning the size of the connection pool. We determined that this was thus probably not the cause of the small/large instance type performance discrepancy.

Next, we analyzed the detailed CPU info we collected, using lscpu and cpufetch to determine the NUMA node count for each run. The NUMA node count describes the number of physical processors a particular virtual machine is running across, and running across multiple physical processors has a material impact on the likelihood of a processor cache miss (among other things). We trusted that the hypervisors were reporting that information honestly, but that may not be the case.

Overall, the gap for most AMD-based processors closed almost immediately when we controlled for NUMA nodes – in other words, when we only considered runs where each instance showed all vCPUs running across a single NUMA node. When we did this, the performance gap dropped from 22% to 1%, which is smaller than our margin of error. While there was still some variability in runs, the best AMD runs on the big instances showed roughly the same per-vCPU performance as the smaller instances after controlling for NUMA nodes.

For Intel-based processors the results were less dramatic, with the gap narrowing from 16% to 12%. This led us to follow-up experiments to explain the remaining gap between the small and large instance sizes for Intel-based instance types.

We looked into variable CPU frequency technologies (Turbo Boost for Intel, Turbo Core for AMD). Our first hypothesis was the difference between how variable processor speeds work in Intel and AMD based processors could explain why we were still seeing the difference in large/small node performance for Intel-based instance types even after controlling for NUMA nodes.

AMD's variable frequency technology (Turbo Core) is focused on multi-core performance, allowing all cores on a socket to scale their frequencies concurrently as long as the thermal load of the processor does not get too high. Intel's variable frequency technology (Turbo Boost) is more focused on single core performance, allowing a limited number of cores per processor to burst to much higher frequencies at the cost of other cores not running at quite as high a frequency. Our supposition was that smaller Intel instances were getting an outsized benefit from Intel's Turbo Boost, which could explain the remaining performance differences.

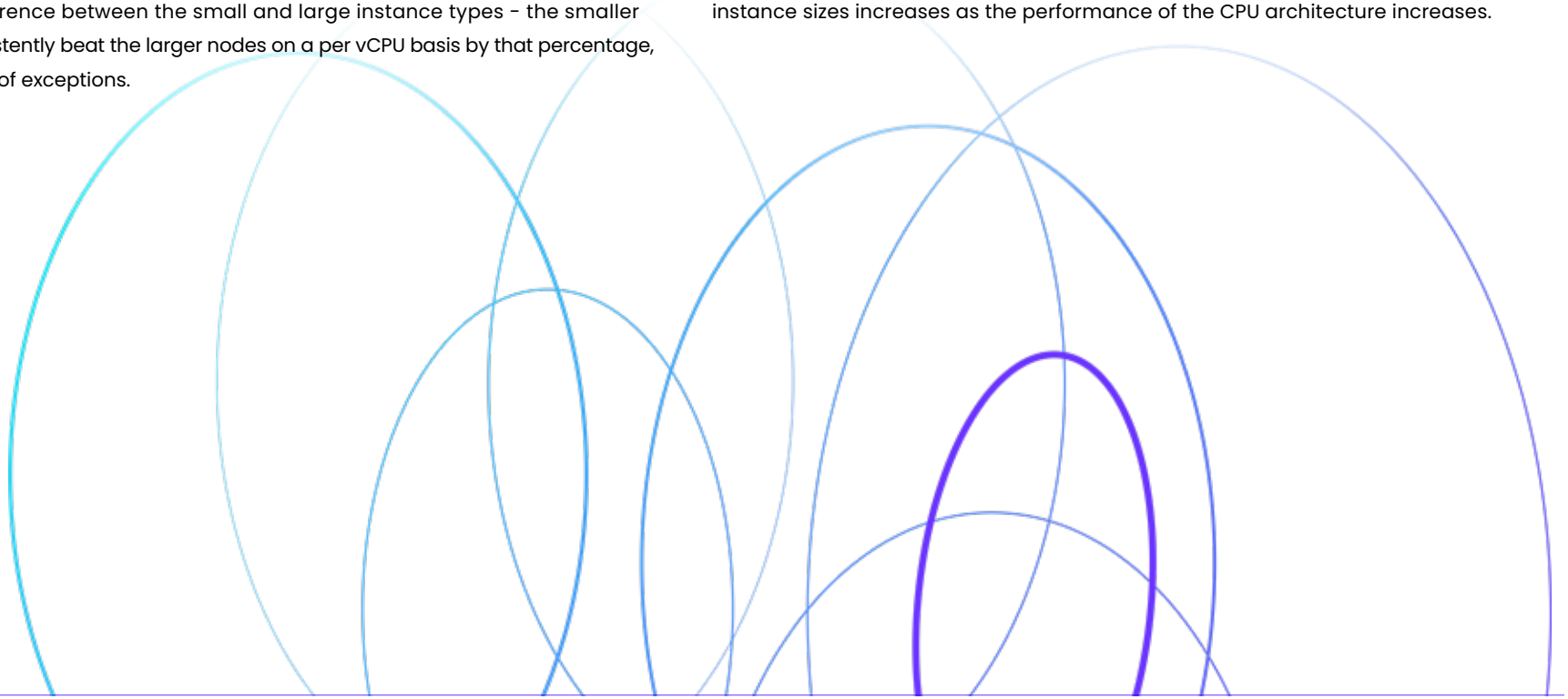
However, when we collected the average per-core frequency every 30 seconds on both small and large instance sizes for a select number of instance types, we did not see a substantial difference between the big and small samples. The average frequency per core for the larger instances was 47 MHz lower than the average frequency for the smaller instances. This works out to an advantage of roughly 1.5% to the smaller nodes – not enough to explain the entire performance discrepancy between large and small nodes on Intel machines.

Pulling in data from the CPU micro-benchmark, we can account for another 2-3% performance difference between the small and large instance types - the smaller nodes pretty consistently beat the larger nodes on a per vCPU basis by that percentage, with only a couple of exceptions.

To confirm these results, and to make an overall recommendation between fewer large instances and more small instances, we ran a handful of tests with identical core counts. We did this by comparing 16 small nodes to 4 large nodes for a handful of instance types and measuring overall performance, with the fastest overall Intel instance types running Intel Ice Lake processors and fastest GCP and Azure instances running AMD Milan (EPYC Gen 3) processors. We also ran a single Cascade Lake scale-up test for comparison purposes.

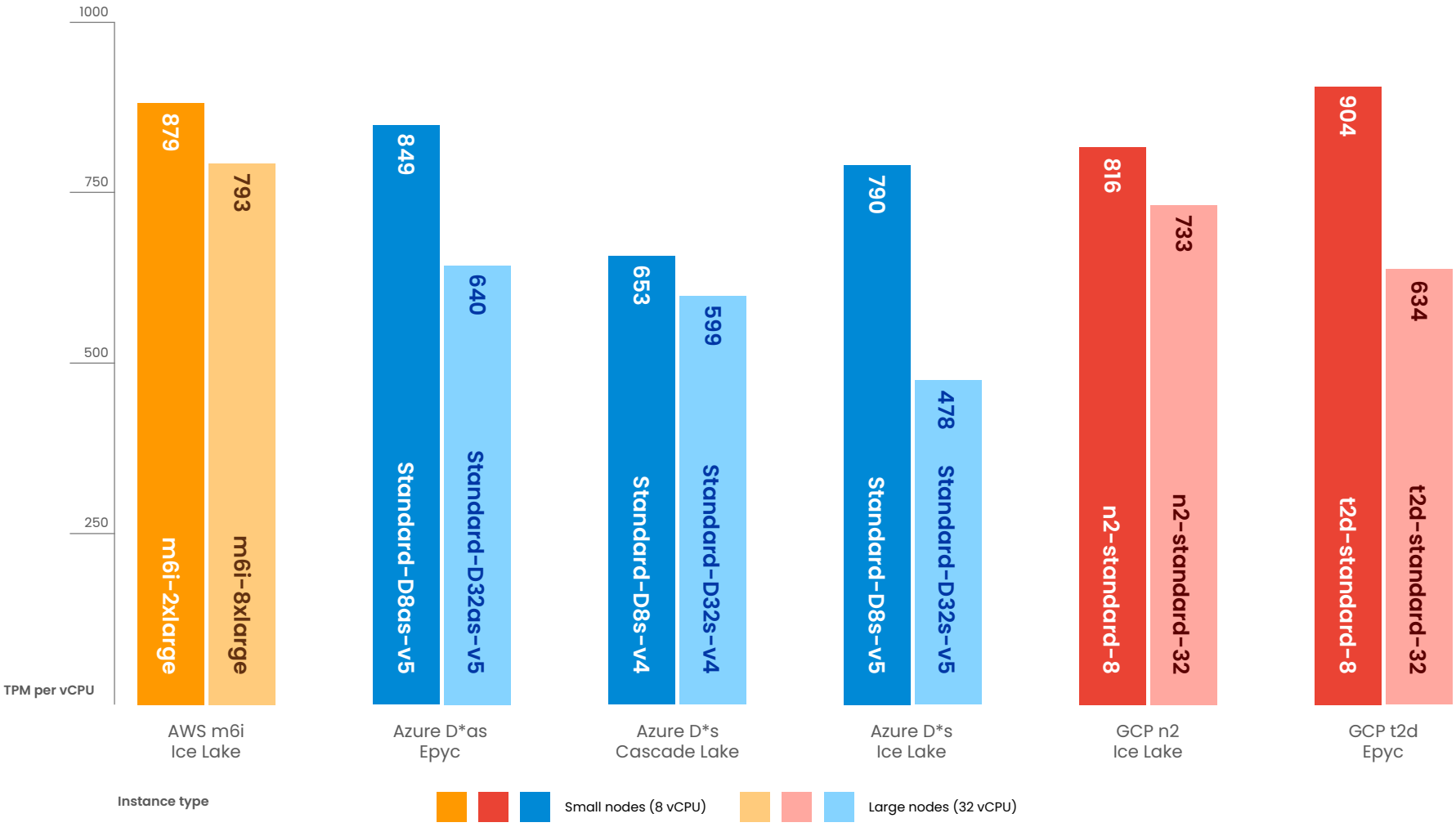
All tests were run using the highest-tier storage available for that instance type, using the same parameters as the flat 1,200 warehouse tests from our wide-scale analysis, with the exception of Azure's standard_D*s_v5. We were not able to get ultra-disks for that configuration, so the 32-core machine was likely I/O bound – in retrospect, we should have provisioned 8 TB volumes to accommodate this. Unfortunately, we didn't have time for multiple runs, so this was a single-run benchmark.

The small instances all outperformed the large instances even after we scaled up to run the exact same number of cores per cluster. In some cases, the performance discrepancy was greater than 30%. Oddly, the difference between the small and large instance sizes increases as the performance of the CPU architecture increases.



OLTP benchmark

Large vs. small node OLTP performance (TPM per vCPU) with identical core counts



Ultimately, we're still not certain of the cause of the discrepancy. It could be because of host oversubscription (we are pushing the database close to max load here), some limitation of the configuration we chose (perhaps we needed to have separate Linux kernel parameters for the small and large instance sizes), some limitation of the database parameters (perhaps we needed to set different database configuration defaults for large nodes than we did for small nodes), or other scalability issues in Go or CockroachDB.

Appendix III: Cloud terminology

Availability Zones or Zones: An Availability Zone (AZ) or Zone is an isolated location— often a data center — within a region. A region is made up of multiple AZs.

Regions: Regions are separate geographic areas defined by the cloud providers (e.g. among many others, GCP has regions called us-east1 and us-west1 located on the US East Coast and West Coast, respectively). The closer a region is to the end user or application, the better the network latency.

gp3 volumes: AWS's general-purpose storage volumes.

io2 volumes: AWS's high-performance storage volumes.

pd-ssd (SSD persistent disks): GCP's general-purpose storage volumes.

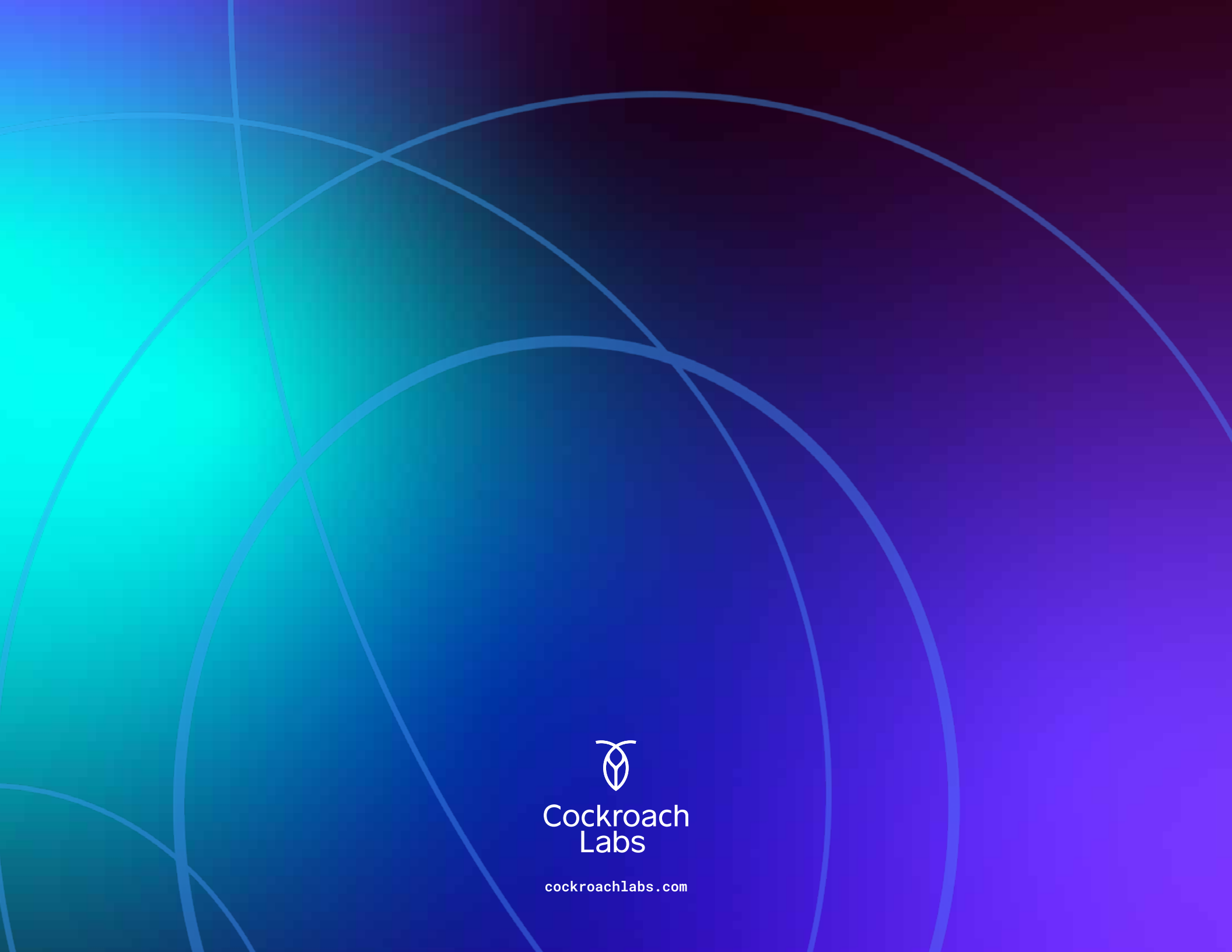
pd-extreme: GCP's high-performance storage volumes.

Placement Policy / Placement Group: Rules designating where to place machines within an availability zone.

Premium-disk: Azure's general-purpose storage volumes.

Ultra-disk: Azure's high-performance storage volumes.

Volume: A unit of data storage, synonymous with "disk" in the context of cloud data storage.


Cockroach
Labs
cockroachlabs.com