Cockroach Labs

2020 Cloud Report

Authors
Paul Bardea, Charlotte Dillon,
Nathan VanBenschoten, & Andy Woods

Table of contents

Background

Last fall, Cockroach Labs introduced its inaugural cloud report focused on benchmarking Amazon Web Services (AWS) and the Google Cloud Platform (GCP). The 2018 Cloud Report was a story of serendipity--we set out on a journey to better understand CockroachDB's performance and ended up discovering a material difference in cloud performance between AWS and GCP. The report generated a large amount of interest from the community and sparked follow-up conversations with all three major cloud providers: AWS, Microsoft Azure (Azure), and GCP.

This year, we've taken our learnings from last year's cloud report to create the 2020 Cloud Report. This report compares the performance of AWS, Azure, and GCP on a series of microbenchmarks and customer-like workloads to help our customers understand the performance tradeoffs present within each cloud and their machine types. We think this report is broadly representative of database performance outside of CockroachDB but should not be understood to provide commentary for all workloads and use cases. For example, machine learning focused users would likely want to use a different set of benchmarks in comparing cloud performance.

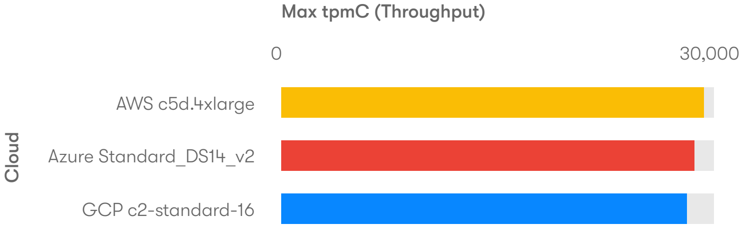
You might be wondering, why the jump from 2018 to 2020? Did we take a year off? We've rebranded the report to focus on the upcoming year. So, like the fashion or automobile industries, we will be reporting our findings as of Fall 2019 for 2020 in the 2020 Cloud Report.

In the 2020 Cloud Report, we've expanded the report by

- Adding Microsoft Azure machines
- Expanding the machine types tested from AWS and GCP
- Open-sourcing a microbenchmarking tool that makes it trivial to reproduce all microbenchmarks
- Completing more than 1,000 benchmark test runs across 26 machine types including CPU, Network Throughput, Network Latency, Storage Read Performance, Storage Write Performance, and TPC-C

[FIG 1]

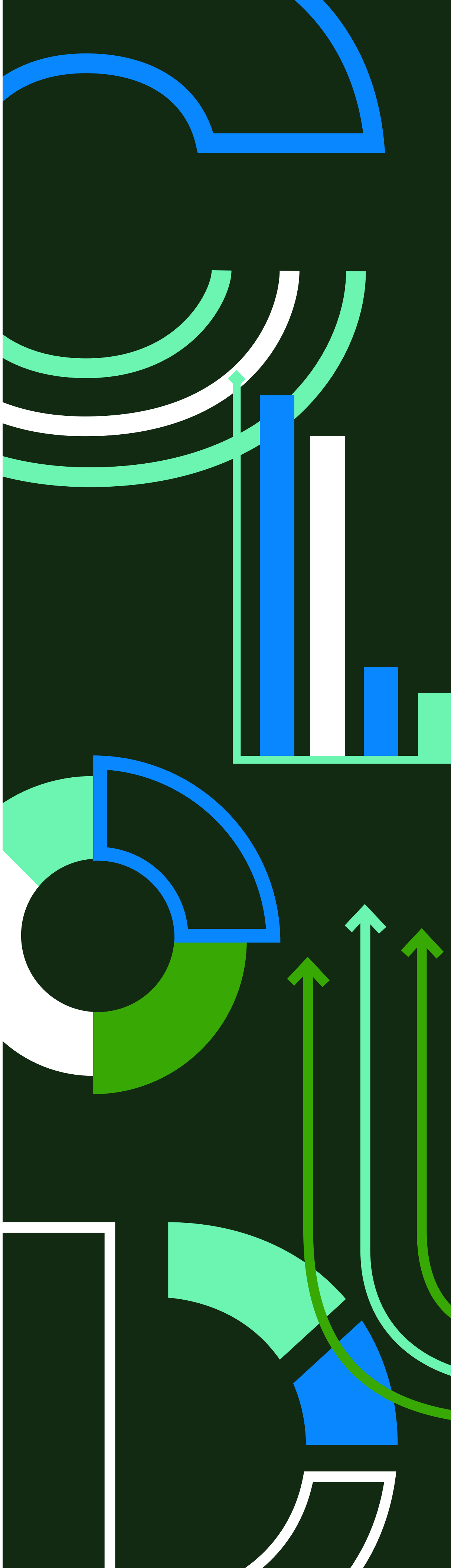
Maximum tpmC Throughput per Cloud



In 2020, we see that GCP has made noticeable improvements in the TPC-C benchmark such that all three clouds fall within the same relative zone for top-end performance.

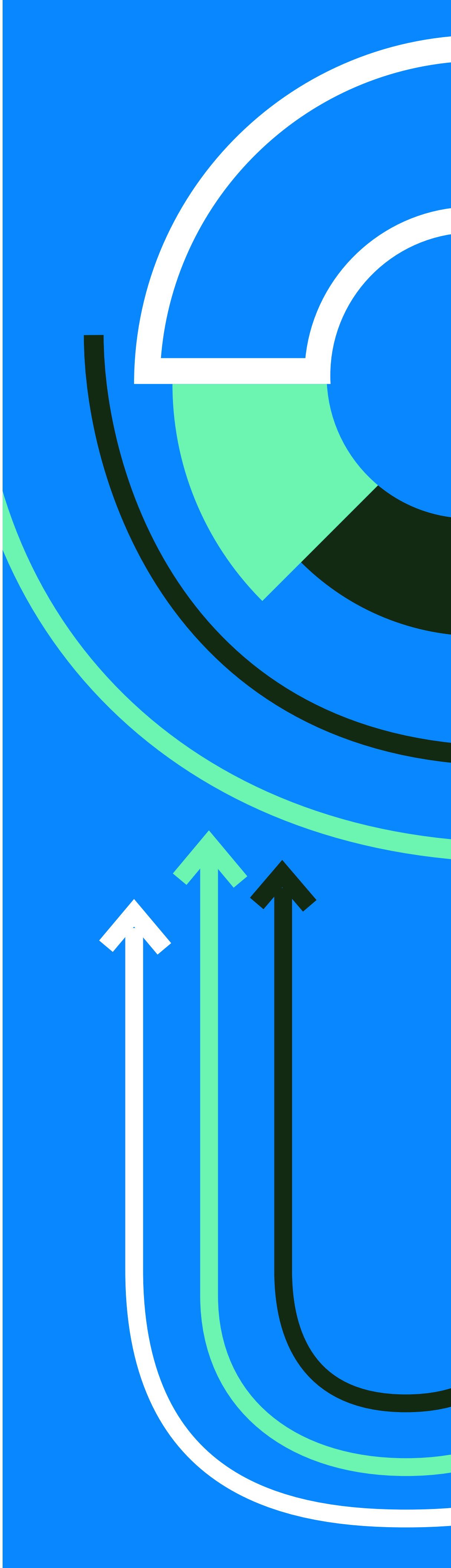
We will discuss these results below but note that this is three-node TPC-C performance.

Reproduction Steps



All reproduction steps can be found in this public repository. These results will always be free and easy to access and we encourage you to review the specific steps used to generate the data in this blog post and report. Note, if you wish to provision nodes exactly the same as we do you can use this link to access the source code for Roachprod, our open-source provisioning system.

Machine Type



We tested the newest and previously top performing machine types available from AWS, Azure, and GCP on three axes: CPU, Network, and Storage I/O. After reviewing the resource-specific microbenchmarks, we took the newest top performing machine types from each cloud and benchmarked them on the industry standard TPC-C benchmark. Note that all TPC-C results are unofficial, and all SSD results were obtained using nobarrier. Please review the chart on the following page for all machine types.

[FIG 2]

Tested Machine Types

Cloud	Machine Type	CPU Platform	Frequency (GHz)	vCPUs	Memory (GiB)	Disk Type
AWS	c5n.4xlarge	Intel Skylake	3.00	16	42	EBS
AWS	c5.4xlarge	Intel Skylake	3.00	16	32	EBS
AWS	c5d.4xlarge	Intel Skylake	3.00	16	32	SSD
AWS	i3.4xlarge	Intel Broadwell	2.30	16	122	SSD
AWS	i3en.6xlarge	Intel Skylake	2.50	24*	192	SSD
AWS	r5d.4xlarge	Intel Skylake	2.50	16	128	SSD
AWS	r5a.4xlarge	AMD EPYC 7000	2.10	16	128	EBS
AWS	r5ad.4xlarge	AMD EPYC 7000	2.10	16	128	SSD
AWS	m5.4xlarge	Intel Skylake	2.50	16	64	EBS
AWS	m5d.4xlarge	Intel Skylake	2.50	16	64	SSD
AWS	m5a.4xlarge	AMD EPYC 7000	2.10	16	64	EBS
AWS	m5ad.4xlarge	AMD EPYC 7000	2.10	16	64	SSD
Azure	Standard_DS5_v2	Intel Broadwell***	2.30	16	56	SSD
Azure	Standard_DS14_v2	Intel Broadwell***	2.30	16	112	SSD
Azure	Standard_F16s_v2	Intel Skylake	2.70	16	32	SSD
Azure	Standard_DS14	Intel Broadwell***	2.30	16	112	SSD
Azure	Standard_E16s_v3	Intel Broadwell***	2.30	16	128	SSD
Azure	Standard_D16s_v3	Intel Broadwell***	2.30	16	64	SSD
Azure	Standard_GS4	Intel Haswell	2.00	16	224	SSD
GCP	n1-standard-16	Intel Skylake**	2.00	16	60	SSD
GCP	n1-highmem-16	Intel Skylake**	2.00	16	104	SSD
GCP	n1-highcpu-16	Intel Skylake**	2.00	16	14.4	SSD
GCP	n2-standard-16	Intel Cascade Lake	2.80	16	64	SSD
GCP	n2-highmem-16	Intel Cascade Lake	2.80	16	128	SSD
GCP	n2-highcpu-16	Intel Cascade Lake	2.80	16	16	SSD
GCP	c2-standard-16	Intel Cascade Lake	3.10	16	64	SSD

*We limited the i3en.6xlarge to 16 CPUs by setting the `--cpu-options` flag that AWS provides to `CoreCount=8,ThreadsPerCore=2`.

**GCP's n1 series offers a variety of CPU platforms (Skylake, Broadwell, Haswell, Ivy Bridge, and Sandy Bridge). We pinned the platform for these tests to Skylake using the `--min-cpu-platform` flag.

***Most Azure machine types offer a variety of CPU platforms (Skylake, Broadwell, and Haswell). All machines we tested on contained Intel Broadwell processors.

Note, we held CPU and Storage as similar as possible across machine types and clouds while allowing memory to match the defaults for these specifications. We chose 16 CPUs as we were most familiar with the performance characteristics of these machine types but note that results could vary by CPU size. We used an ubuntu-1604-xenial-v20190122a OS image across all three clouds. We expect the clouds to choose the best images for providing good performance for their VMs. While it is possible that the various permutations of Linux may impact performance, we did not test this effect.

For each cloud, we ran in the US-central or US-east regions where possible. We expect performance to vary across both availability zone and time of day based upon loads at the various cloud data centers. We did not study the impact of location or time.

Last year, we used a SCSI interface for locally attached SSDs on GCP and an NVMe interface for locally attached SSDs on AWS because these were the defaults. This year, we switched over to using an NVMe interface wherever possible across all clouds. By making this change, we observed a 7% increase in throughput in performance when running TPC-C on GCP.

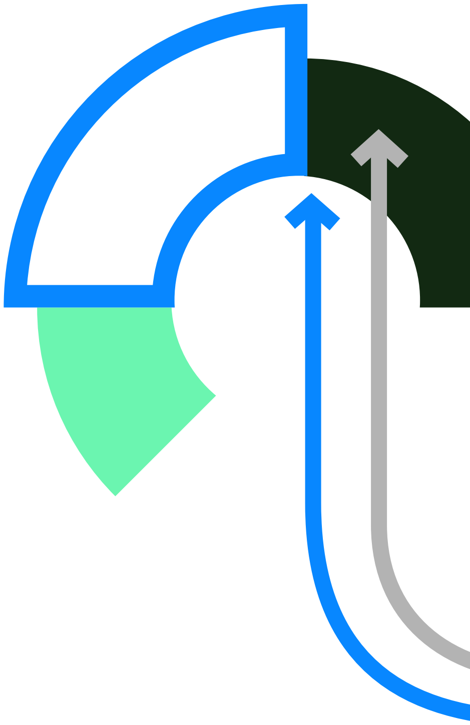
Top Performing Machine Types

Here is a chart outlining the top performing new machine types on each cloud:

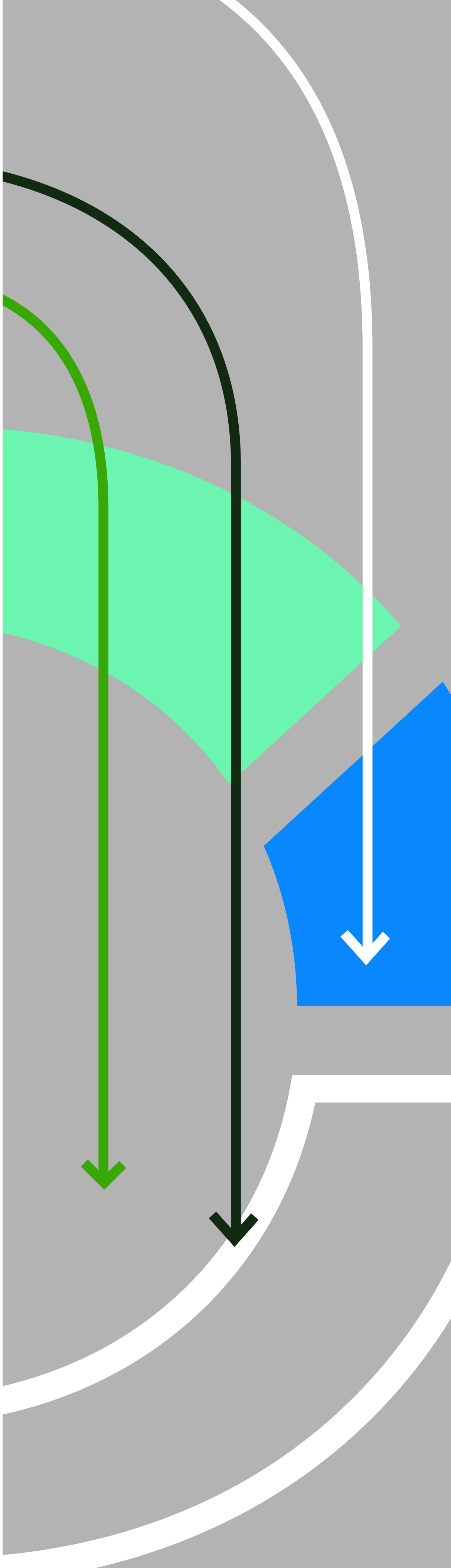
[FIG 3]

Cloud	Machine Type	CPU Platform	Frequency (GHz)	vCPUs	Memory (GiB)	Disk Type	Disk Size (GB)
AWS	c5d.4xlarge	Intel Skylake	3.00	16	32	SSD	1 x 400
AWS	c5n.4xlarge	Intel Skylake	3.00	16	42	EBS	N/A
AWS	i3.4xlarge	Intel Broadwell	2.30	16	122	SSD	2 x 1,900
AWS	i3en.6xlarge	Intel Skylake	2.50	24*	192	SSD	2 x 7,500
Azure	Standard_GS4	Intel Haswell	2.00	16	224	SSD	1 x 448
Azure	Standard_F16s_v2	Intel Platinum 8168	2.70	16	32	SSD	1 x 128
Azure	Standard_DS14_v2	Intel Broadwell	2.30	16	112	SSD	1 x 224
GCP	n2-standard-16	Intel Cascade Lake	2.80	16	64	SSD	2 x 350
GCP	n2-highmem-16	Intel Cascade Lake	2.80	16	128	SSD	2 x 350
GCP	n2-highcpu-16	Intel Cascade Lake	2.80	16	16	SSD	2 x 350
GCP	c2-standard-16	Intel Cascade Lake	3.10	16	64	SSD	2 x 350

*We limited the i3en.6xlarge to 16 CPUs by setting the `--cpu-options` flag that AWS provides to `CoreCount=8,ThreadsPerCore=2`.



Experiments



Unlike last year, we began our experimentation with microbenchmarks instead of TPC-C. We switched the order of the analysis because we wanted to consider more machine types and, in particular, the differences in how the microbenchmarks impact customer workloads. At a macro-level, we evaluated each cloud on a simulated customer workload (TPC-C). We finished the experiments with a focus on customer workloads (and not microbenchmarks) because it most directly simulates real-world customer behavior and provides a holistic picture of performance.

CPU Experiment

We tested a compute performance microbenchmark first as it can have a large impact on performance. CPU is a surface benchmark affected by variables across the hardware (and software) stack. Those variables range from processor microarchitecture to memory hierarchy, to the hypervisor and system kernel running on top of it. CPU is also affected by both the cloud instance types you choose and the CPU platforms your virtual machines (VMs) are placed onto. It can even vary across VMs that claim to use the same CPU.

With so many different variables affecting this layer, there are a lot of different metrics one could look at to benchmark performance, ranging from floating point operations and bit manipulations to measuring the performance of control-flow patterns. For tools that measure all of these, the two we explored most thoroughly are stress-ng and sysbench. Stress-ng is an open-source, third-party tool that anyone can use to benchmark cloud providers. We chose stress-ng over sysbench because it offers more benchmarks and provided more flexible configurations than sysbench.

The output of these tests is a metric referred to as Bogo Ops (bogus operations/second). As the name implies, it's best to avoid putting too much weight into what these Bogo Ops are, but it's a useful metric for comparing across machines.

For this experiment, we chose to test all machine types using stress-ng's `matrix` stressor. This stressor provides a good mix of memory, cache, and floating point operations. We found its behavior to be representative of real workloads like CockroachDB. Because of this, the results we see here have a strong correlation with the results we see later in TPC-C.

This is in contrast with the `cpu` stressor, which steps through its 68 methods in a round-robin fashion, allowing it to be disproportionately affected by changes in some of its slower, less-representative methods like stressing deeply recursive call stacks. GCP and AWS both shared concerns about the `cpu` stressor with us ahead of time and our own experimentation validated those concerns. The results we found using the `cpu` stressor were difficult to explain across CPU platforms and were not useful predictors of TPC-C performance, indicating that it is not a representative benchmark. We therefore decided only to present results using the `matrix` stressor.

About stress-ng

Stress-ng comes packaged with a large suite of stress mechanisms, called "stressors", each one exercising one or more subsystems of a computer. Here are a few, for example

`matrix`
stresses floating point operations, memory, and processor data cache

`sem`
stresses POSIX semaphore operations and rapid context switching between OS threads

`branch`
stresses CPU branch prediction logic.

`cpu`
stresses CPU through a suite of 68 "methods"

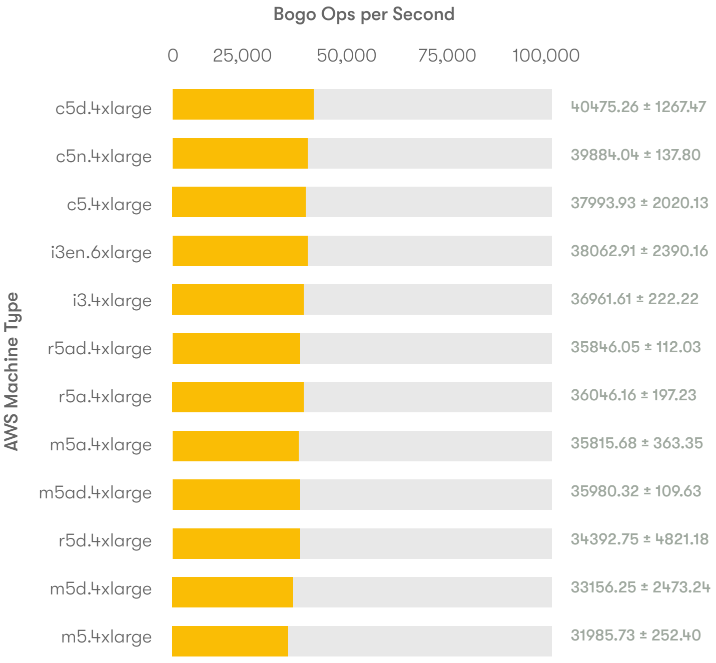
AWS CPU

AWS produced a wide range of bogo ops per second across their many machine types. It's unsurprising that the C series, AWS's compute optimized series, outperformed their other machine types. In general, we see the machine types with higher processor clock frequencies dominating this comparison.

AWS is also the only cloud where we tested AMD machines. Specifically, we tested four different AMD EPYC 7000 series instance types, which each contain an "a" specifier. The CPU benchmarks do not show a significant difference one way or the other when comparing these against the Intel processors.

[FIG 4]

AWS: Average Bogo Ops per Second

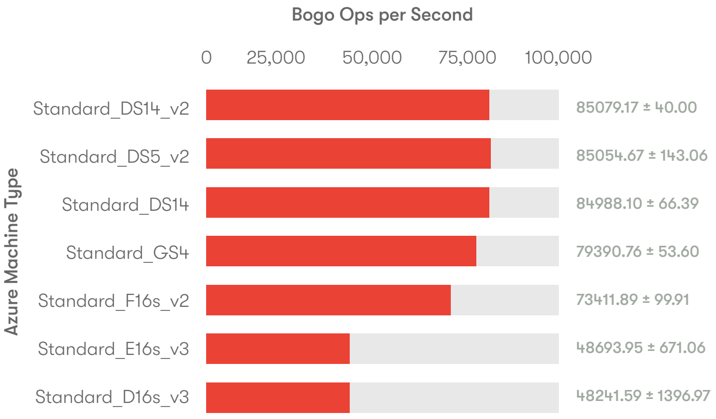


Azure CPU

Azure, like AWS, offers a wide range of performance profiles on Stress-ng. The groupings we see with the Azure machine types were primarily defined by 2 factors: the processor provided and hyperthreading. Most machines were provided with 16 cores and 1 thread per core. The exceptions were the Standard_F16s_v2, Standard_E16s, and the Standard_DS16s_v3. This was the largest factor in influencing the CPU benchmark. Within each of those groups, the score for each machine achieved what was expected based on the frequencies of the CPUs being used.

[FIG 5]

Azure: CPU Average Bogo Ops Per Second

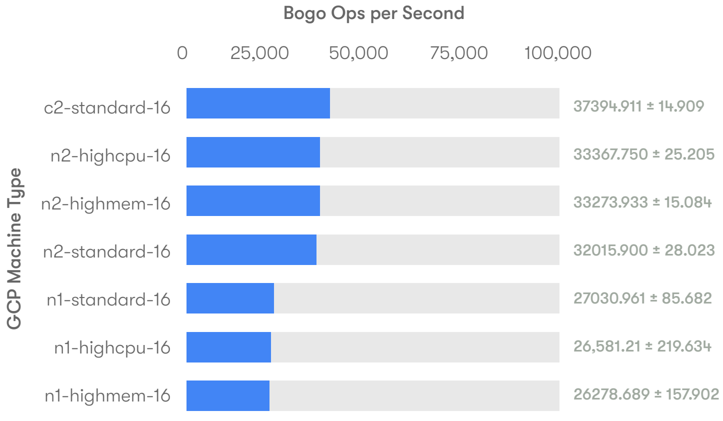


GCP CPU

GCP introduced both the n2 series and the c2 series earlier this year, both of which use the new Intel Cascade Lake Processor. We see from the benchmarks these result in significantly higher performance than the corresponding n1 series instances, which use last generation Intel Skylake processors. Like with AWS, the results here are highly correlated with the processor clock frequency for each instance type.

[FIG 6]

GCP: Average Bogo Ops Per Second



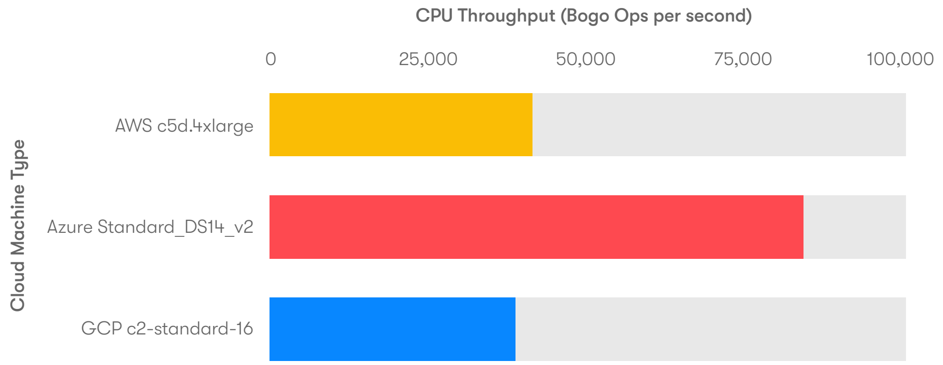
Overall CPU

The best performing Azure machines achieved significantly better results on the CPU microbenchmark. The largest difference between the CPUs on each cloud was that even though all machines have 16 vCPUs, the top performing Azure machines use 16 cores with 1 thread per core. The other clouds use hyperthreading across all instances and use 8 cores with 2 threads per core to achieve 16 vCPUs. The effects of avoiding hyperthreading may have inflated benchmark and is not guaranteed to directly represent performance on other workloads. The other takeaway from this comparison is that these results are highly correlated with the clock frequency of each instance type.
As expected, this appears to be a good indicator of compute performance.

One big omission from this analysis is a price per CPU metric, similar to TPC-C's price per tpmC. We plan to add this in next year's version of the report but chose not to include it in this version since it wasn't a disqualifying metric in the build-up to TPC-C.

[FIG 7]

CPU Throughput: Maximum per Cloud



Network Experiment

We split network benchmark testing into two tests: one for network throughput, and one for network throughput latency. Throughput is the quantity of data being sent and received over a time period. Latency is the time required to transmit a packet across a network. Round-trip latency includes the return time. It's important to note that latency is highly dependent upon your network topology. You can expect much lower latency if you're sending information between VMs within the same availability zone than you can if you have VMs across zones. As a reminder, we ran in the US-central or US-east regions where possible for each cloud.

To test throughput, we used a popular open-source tool called iPerf and latency via another common open-source tool, ping.

Since last year, we've learned that our network tests did not show the true network IO throughput as the network wasn't saturated to max capacity by client traffic. A single client may not drive enough IO to max the network (e.g. underpowered CPU vs. large network capacity). A common practice is to use several clients on different hosts to drive traffic to a single host. Last year, we used a single client and reported numbers from the client side. This leaves a reviewer with uncertainty whether the true max network throughput capacity is measured. This year, we improved our test setup by testing load from multiple clients (i.e., 3) and observing the results from a single destination server.

Another major change from last year is that we increased the periodicity of `ping` by using the `-interval` flag. The high default interval used last year can allow the CPU to go to sleep and therefore impact performance.

About iPerf and ping

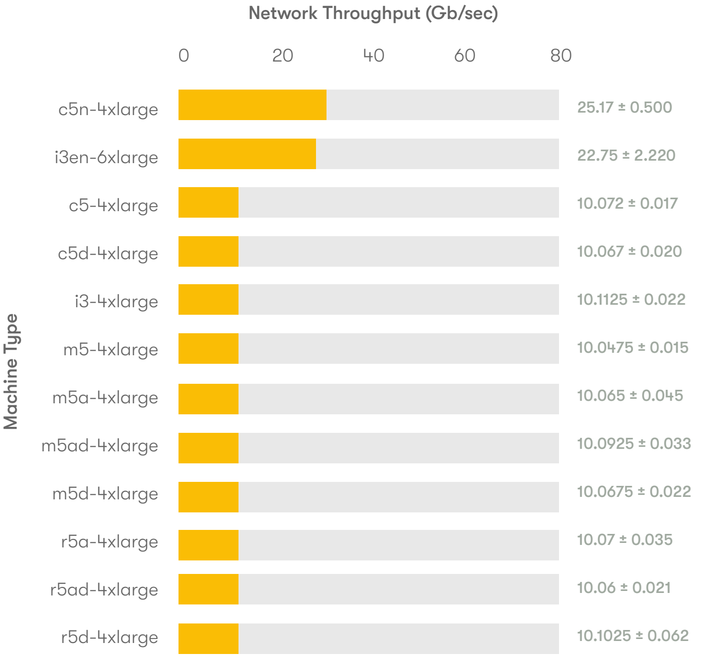
iPerf attempts to connect the client and the server with the data from buffer size via the protocol. iPerf's configurations include a buffer data size (128kb), a protocol, a server, and a client. This test provides a throughput for the network which allows for us to compare the performance of the network on AWS, Azure, and GCP. iPerf supports both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). Pick the right protocol for your use-case, as it can have huge implications on performance.

AWS Network Throughput

AWS is a leader in network performance transparency as they publish their expectations. Other clouds didn't provide their expectations publicly which made it more difficult to sanity check our results. In addition to transparency, we found that the AWS network results reliably matched the specs offered as, for example, we observed the c5n machine type to exactly match their claim of 25 Gbps of peak bandwidth.

[FIG 8]

AWS: Network Throughput Multi

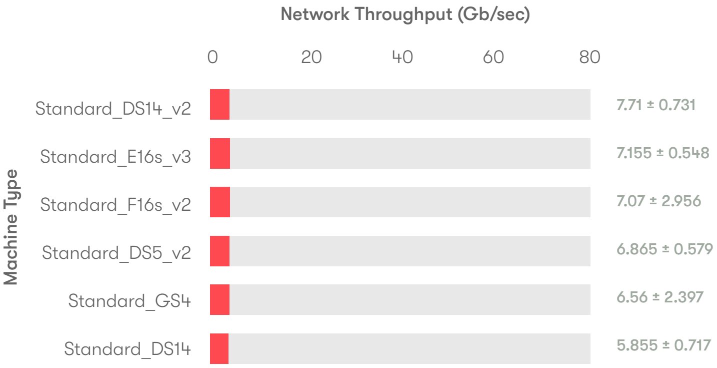


Azure Network Throughput

Azure's top performer on network throughput, the Standard_DS14_v2 clocks in at 8 Gb/sec, 2 Gb/sec below the minimum network throughput offered by AWS. In our testing, the AWS network looks categorically better than Azure's network.

[FIG 9]

Azure: Network Throughput Multi

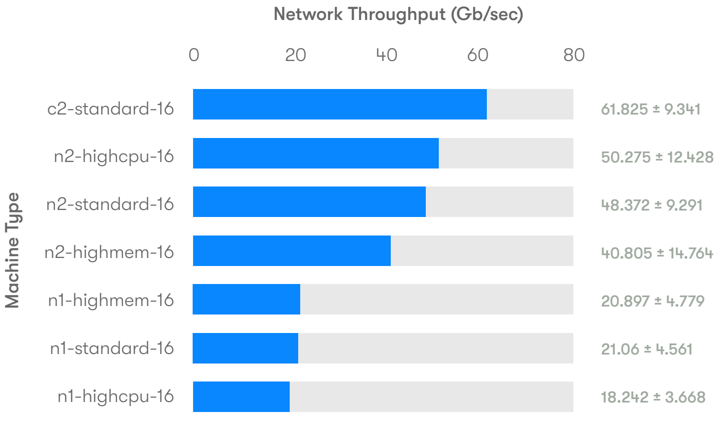


GCP Network Throughput

As discussed above in the machine type section, GCP recommends using `--min-cpu-platform=skylake` for the n1 family of machines, which they believe has an outsized impact on network performance. GCPs bottom machine type, the n1-highcpu-16 is inline with AWS's top performing machines.

[FIG 10]

GCP: Network Throughput

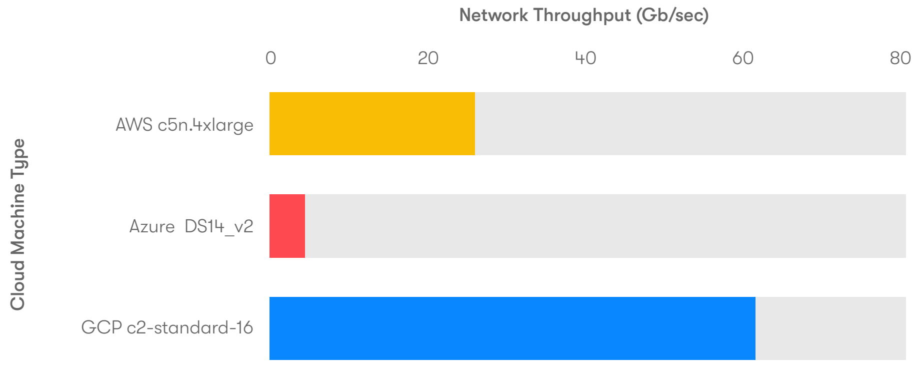


Combined Network Throughput

GCP's network looks much better than either AWS or Azure's networks. Not only do their top performing machines beat each network's top performing machines, but, so to do their bottom performing machines. Even their least performant machine (n1-highcpu-16 in figure 10) is consistent with AWS' maximum network throughput as seen in our tests. This is especially impressive because last year, AWS outperformed GCP in our network tests. It is a credit to GCP that they have improved their network performance and we are left wondering exactly how they accomplished this improvement.

[FIG 11]

Maximum Network Throughput - Multi



Network Latency

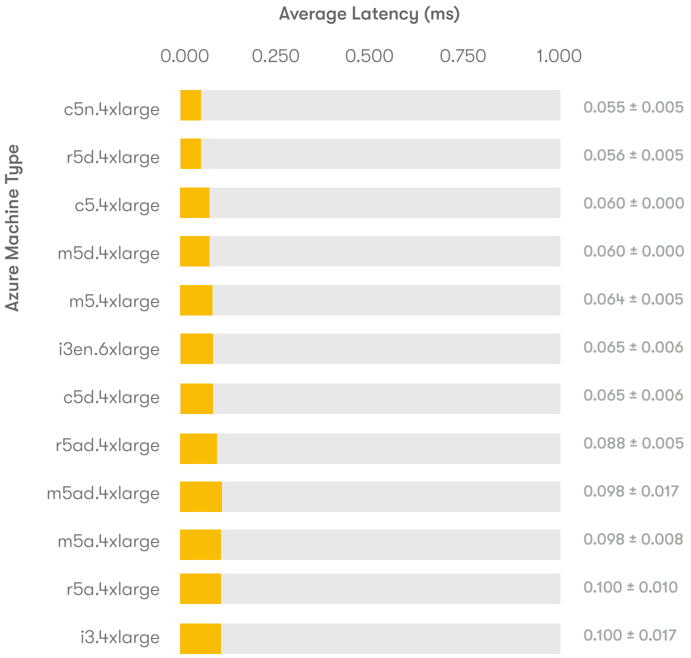
In addition to throughput, we also tested network latency. Without testing for latency we can miss significant delays in service that may be masked by overall performance. For example, latency limits the performance of individual operations.

AWS Network Latency

We see a familiar C5 series entry leading the way for low latency on AWS. AWS network latency is also remarkably consistent with a narrow range. Predictability is preferable because it provides consistent expectations for users.

[FIG 12]

AWS: Network Latency

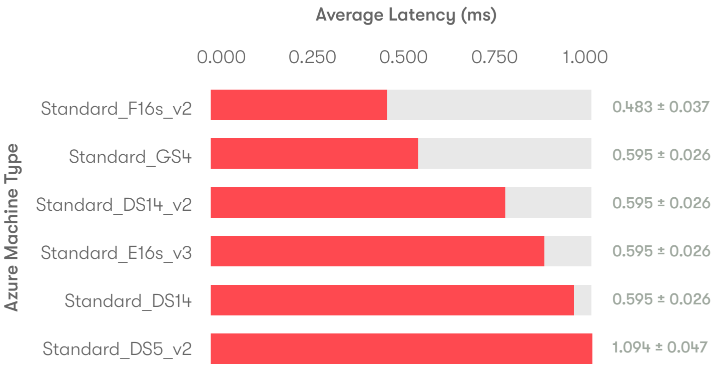


Azure Network Latency

Similarly to CPU, Azure offers a large degree of spread among its various machine types. In addition, Azure Network latency is significantly higher than either GCP or AWS. This matches their comparatively poorer performance on Network throughput.

[FIG 13]

Azure: Average Network Latency

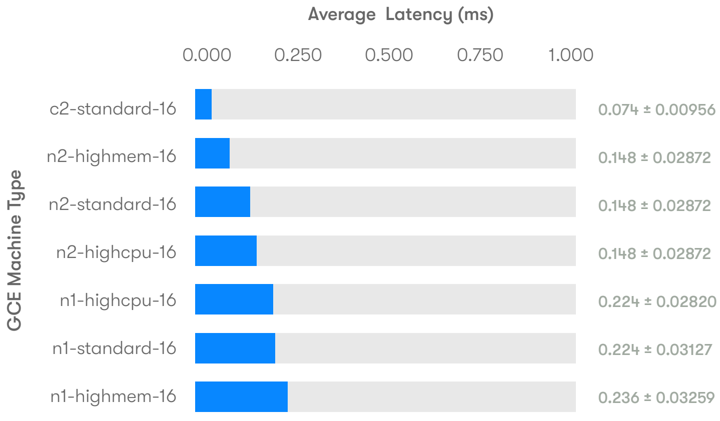


GCP Network Latency

The n1 series offers approximately the same network latency as what we observed in our previous report. However, unlike in CPU, GCP's n2 series dramatically improved its network latency. In addition, the C2 series offers even better network latency. GCP clearly made great strides in its entire network to increase throughput and lower latency at the same time.

[FIG 14]

GCP: Average Network Latency

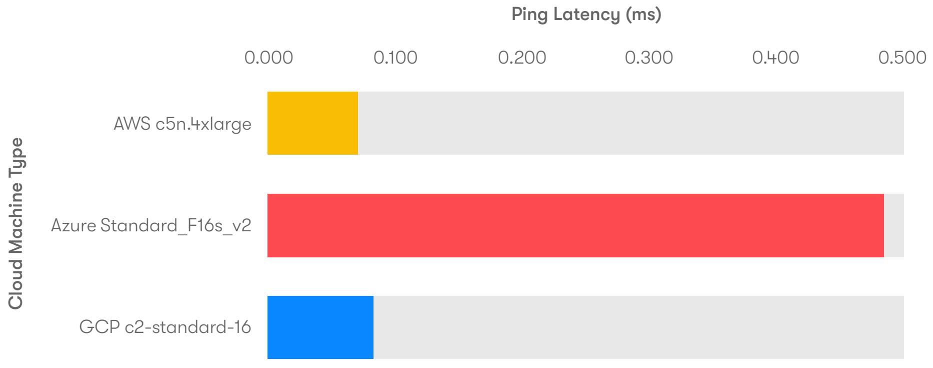


Combined Network Latency

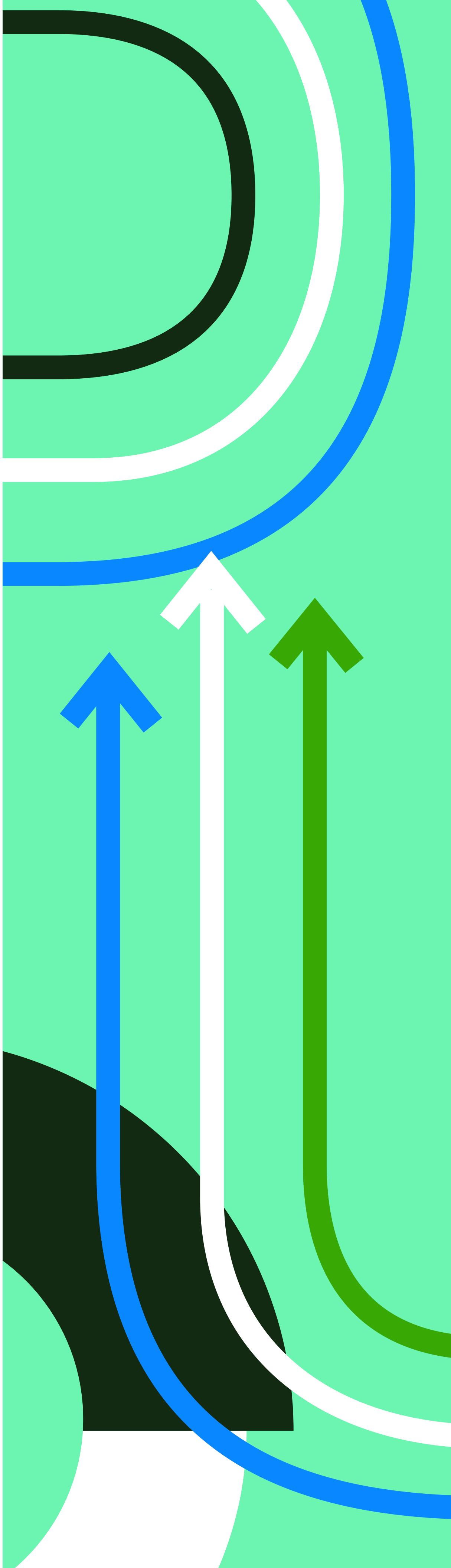
Even the best machine on Azure is more than 5 times worse than on AWS or GCP. GCP dramatically improved its network latency since the last version of this report, but AWS is still king. All of its machine types offer low network latency on average and its top end machines outperform competitors.

[FIG 15]

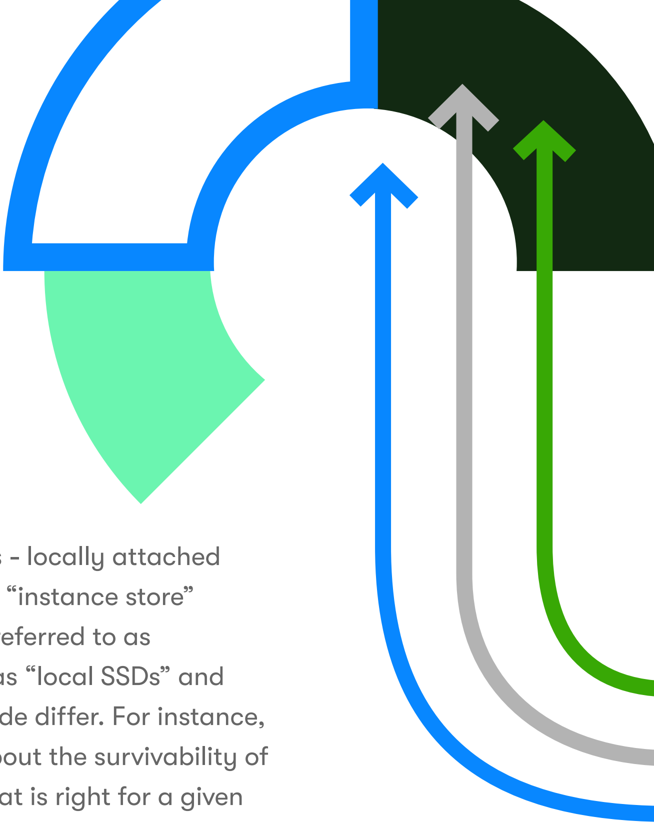
Network Latency: Minimum Average Latency



Storage I/O Experiment



Once again, the importance of storage I/O depends on your application. For CockroachDB, an application that's always reading and writing to persistent storage, this is critical. For other stateless applications, storage performance may not make such a difference. It's important to note that there's a variety of different storage technologies available, from classic spinning hard discs and modern SSDs to network-attached storage and replicated storage. Even when you're running in the cloud, there are a number of choices you can make when provisioning storage.



To start, storage hardware comes in two flavors in these cloud offerings - locally attached storage and network attached storage. In AWS, these are referred to as “instance store” volumes and “elastic-block storage” (EBS) volumes. In Azure, these are referred to as “temporary disks” and “managed disks”. In GCP, these are referred to as “local SSDs” and “persistent disks” (PD). The guarantees that these storage devices provide differ. For instance, network attached storage volumes typically have strong guarantees about the survivability of data across instance lifecycle events. Choosing the flavor of storage that is right for a given application requires taking both the guarantees of the disks and the performance of the disks into account.

As with network benchmarking, storage benchmarks are split into two camps: throughput and latency. These measurements take on a similar meaning, but there are a few subtle details. For one, we’re no longer dealing with symmetric components, so latency is always implicitly defined as round-trip latency (from a user to a storage application and back).

Storage I/O measurements also have another dimension to them that’s critical to acknowledge. Storage devices provide interfaces to read and write data, and these typically have vastly different performance characteristics. We’re going to measure both of these independently to get a holistic picture of storage performance.

We tested I/O using a configuration of sysbench that simulates small writes with frequent syncs for both write and read performance. We ran the sysbench test writing to an SSD to achieve similar results to running a database in production. This test measures throughput based on a fixed set of threads, or the number of items concurrently writing to disk.

About sysbench

Sysbench is an open-source benchmarking tool that’s popular in the database world, since it began as a database-specific benchmarking suite. It’s since evolved into a general purpose benchmarking tool with a lot of flexibility. For our purposes, we’ll be using it as a filesystem level benchmark as we test storage devices.

About Storage I/O Write Performance

At its core, sysbench is measuring how much data it can write from storage per second. We ran the benchmark over a range of concurrency levels, starting at 1 thread (writing alone) moving our way up to 64 threads (writing concurrently). This was really important to test, because storage devices have different performance levels for different levels of concurrency.

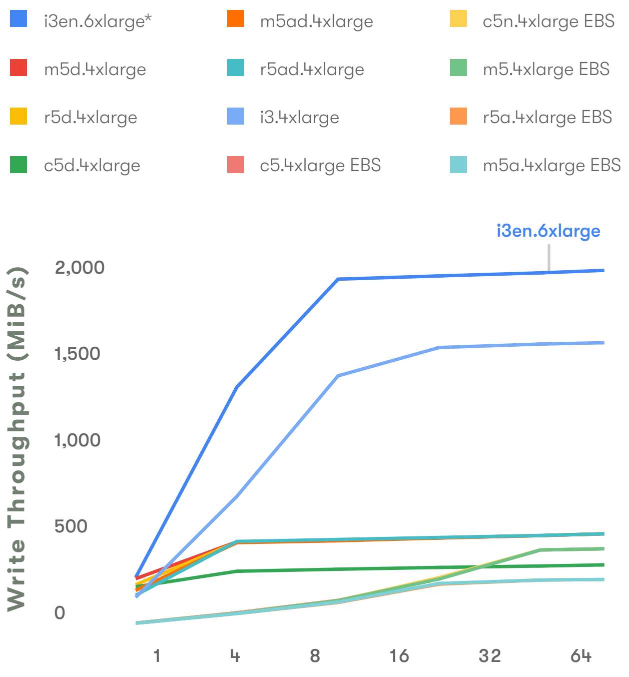
AWS Storage I/O Write Performance

For AWS, most machine types come with a fixed number of SSDs. Other clouds offer more flexibility. As a result, we chose to have all clouds limited to one SSD to make it easy to make comparisons across benchmarks.

AWS storage optimized machines, the i3 series (e.g., i3 and i3en), offer vastly superior storage write throughput. We also noticed a linear increase in throughput up through 8 threads, whereupon it becomes mostly flat. This is in sharp contrast to the other machine types which don't increase nearly as steeply with the number of threads.

[FIG 16]

Storage: AWS Write Throughput (MiB/s)

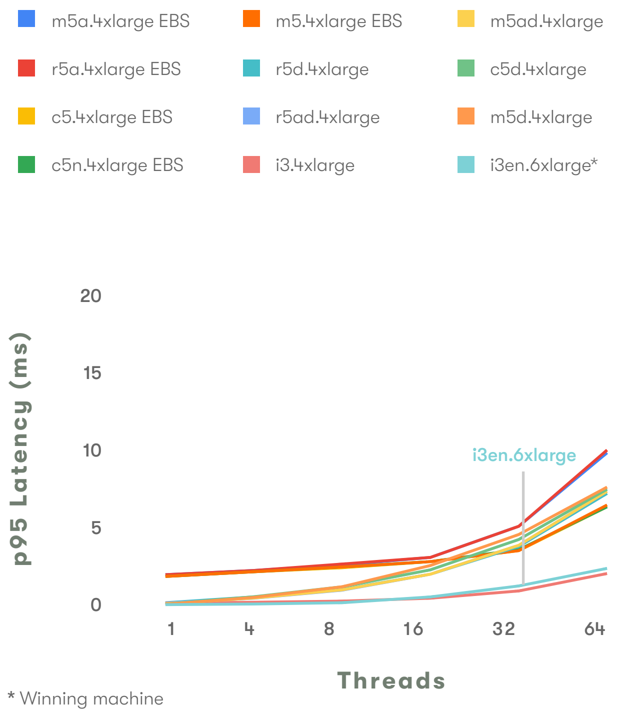


* Winning machine

Similarly to throughput, the i3 series (e.g., i3 and i3en) offer the lowest write latency as well. It’s interesting how all of the machine types increase write latency with the number of threads as contrasted to the shapes of these lines in the write throughput.

[FIG 17]

Storage: AWS p95 Write Latency

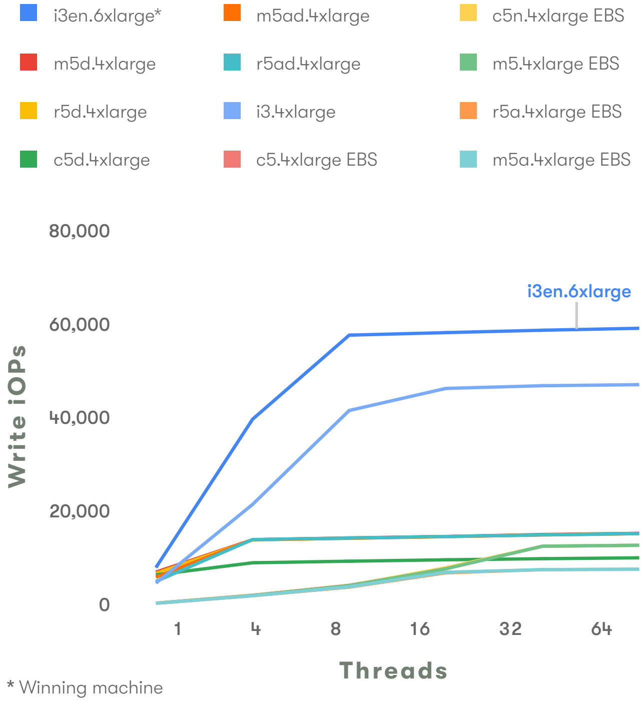


* Winning machine

Unsurprisingly, the i3 series also offers the most amount of IOPS on the write benchmark with similar patterns to the throughput discussed above.

[FIG 18]

Storage: AWS Write iOPS



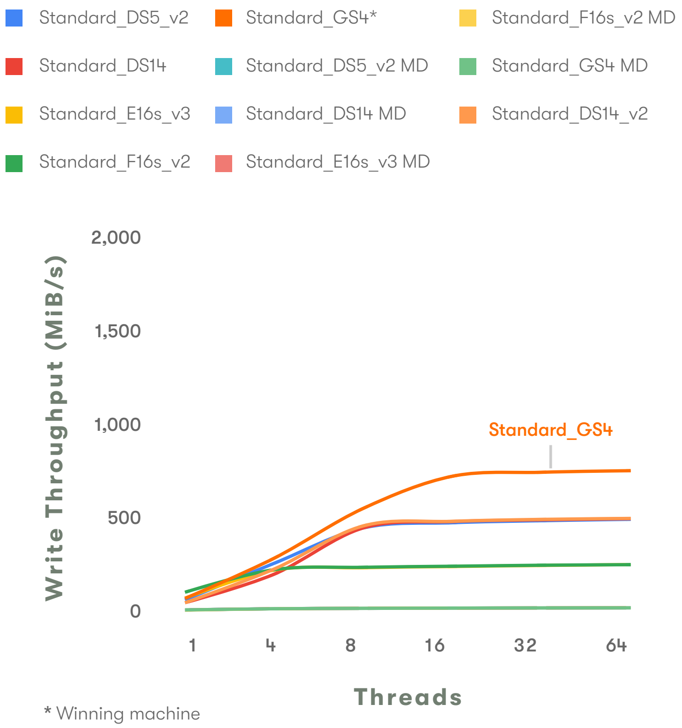
* Winning machine

Azure Storage I/O Write Performance

Azure didn’t offer similar priced and scoped storage instance types when compared to AWS. However, it did offer a similar range of performance in its other machines.

[FIG 19]

Storage: Azure Write Throughput (MiB/s)

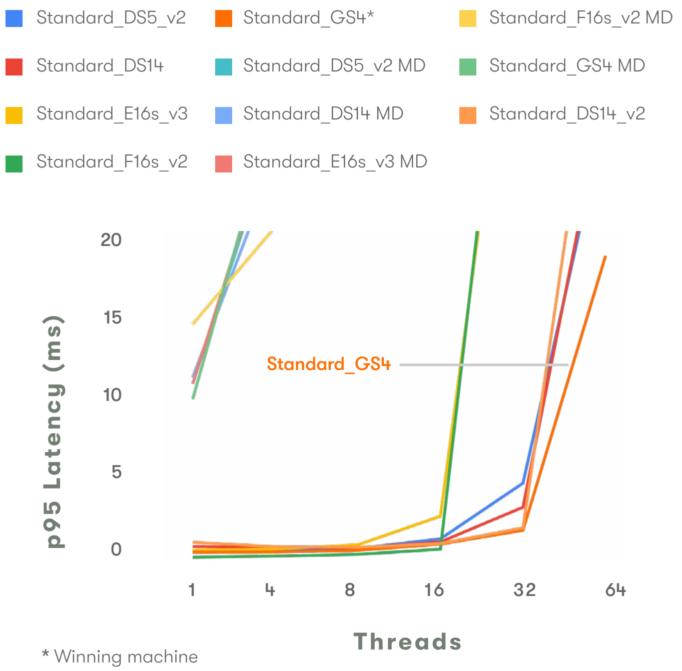


* Winning machine

Initially, Azure demonstrates low latency for threads up to 16. However, after 16 threads, Azure’s latency grows much higher than AWS’s write latency. Note that the MD types quickly fall off the scale of this chart (which we held consistent with AWS and GCP). At 16 threads and higher, the Azure SSDs also outpace the graph’s scale. Azure consistently underperforms AWS and GCP on storage write latency.

[FIG 20]

Storage: Azure p95 Write Latency

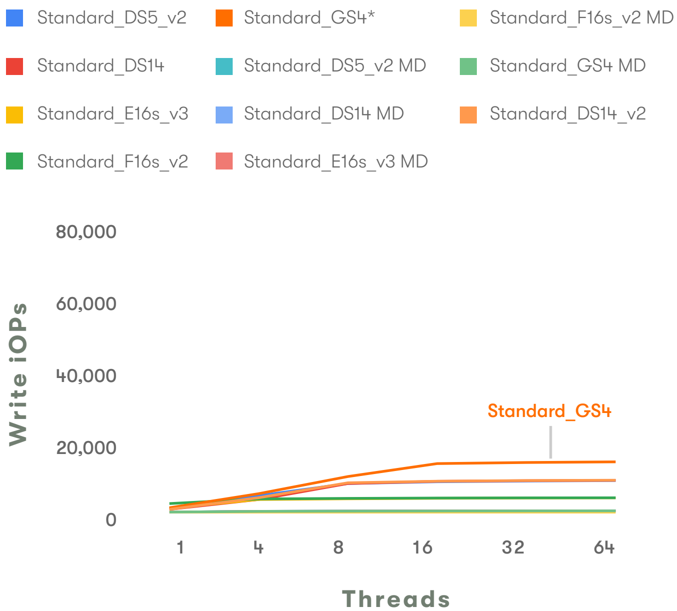


* Winning machine

Similarly to write throughput, the Standard_GS4 outperforms the other Azure machine types in write IOPs. Also like throughput, the spread varies among machine types by a large amount. Note that the managed disks (md) all result in lower IOPs than the SSDs on this benchmark.

[FIG 21]

Storage: Azure Write iOPS



* Winning machine

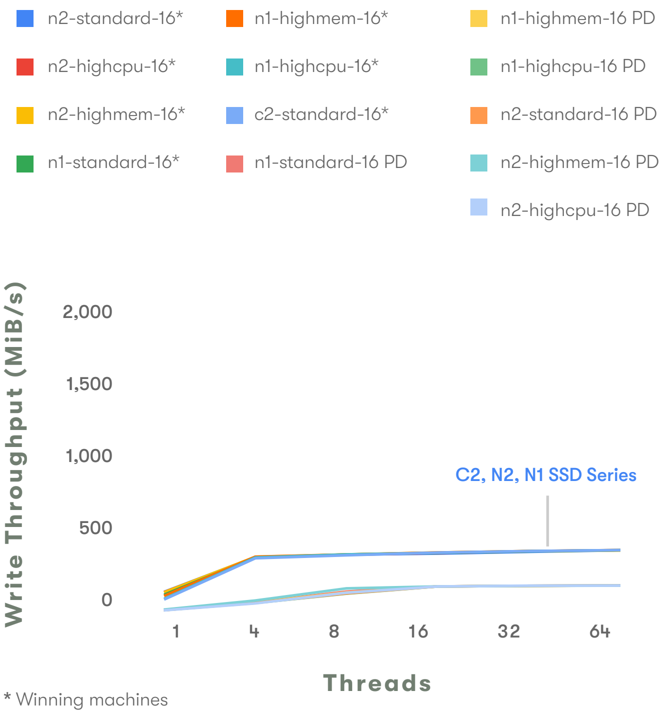
GCP Storage I/O Write Performance

GCP doesn't have a storage-optimized instance, but local SSD can be attached to most VMs with either NVMe or SCSI interfaces. We like the flexibility GCP provides in allowing users the ability to configure the number of SSDs attached to a single host. Other clouds don't provide this same flexibility. For the tests below (and throughout this report), we only used one SSD (despite two being the default for some instance types) to be able to make cross cloud comparisons. These micro-benchmarks didn't take cost into account (but will likely do so next year) so it doesn't penalize GCP that we are only using one disk for this analysis.

Unsurprisingly, SSDs outperformed the persistent disk (PDs) in storage throughput benchmarks. We can clearly see that all SSDs outperform all PDs.

[FIG 22]

Storage: GCP SSD Write Throughput (miB/s)

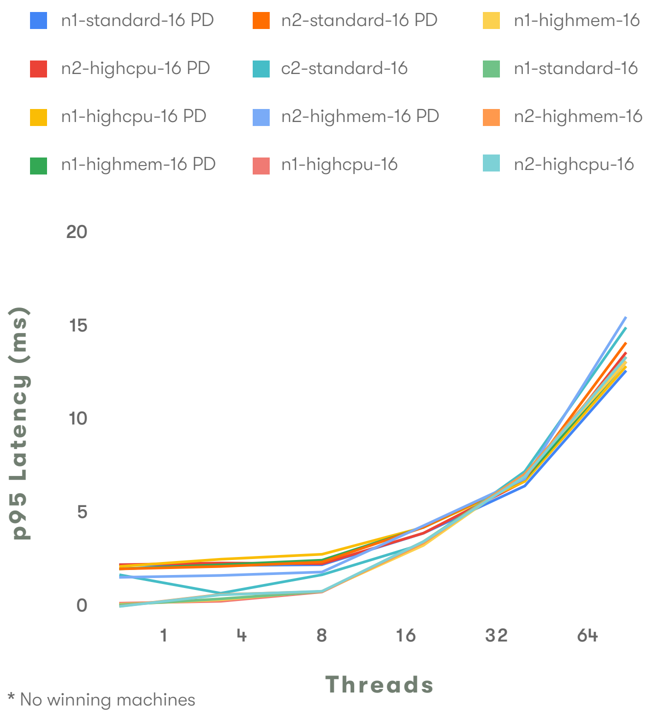


* Winning machines

It’s also interesting to note how closely each machine’s curve resembles the other machines. There doesn’t appear to be a strong difference in storage profiles by machine type in GCP when accounting for SSD or PD. Latency didn’t seem to be impacted by the delineation between SSDs and PDs. In fact, there is little variance across any of the machine types for write latency.

[FIG 23]

Storage: GCP p95 Write Latency

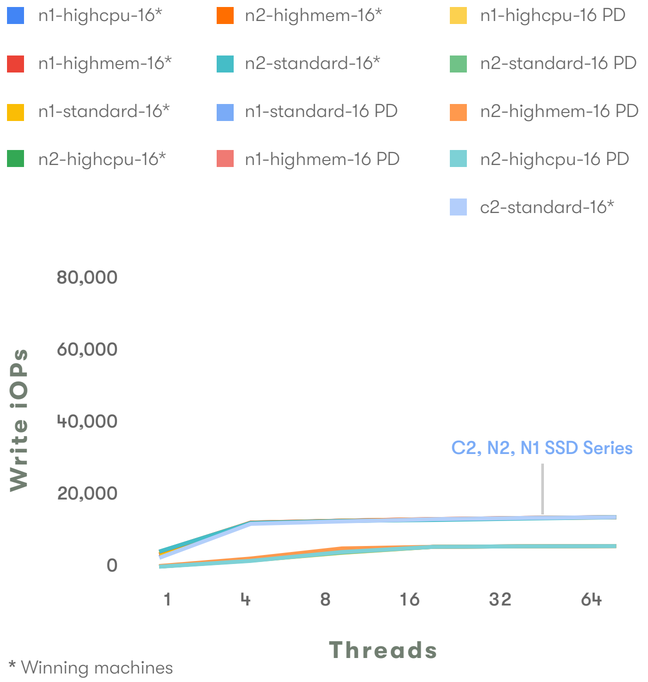


* No winning machines

Unsurprisingly, IOPS mirrors the throughput distribution. All SSDs outperform all PDs. And, similarly to above, the results appear nearly identical by machine type.

[FIG 24]

Storage: GCP Write IOPS



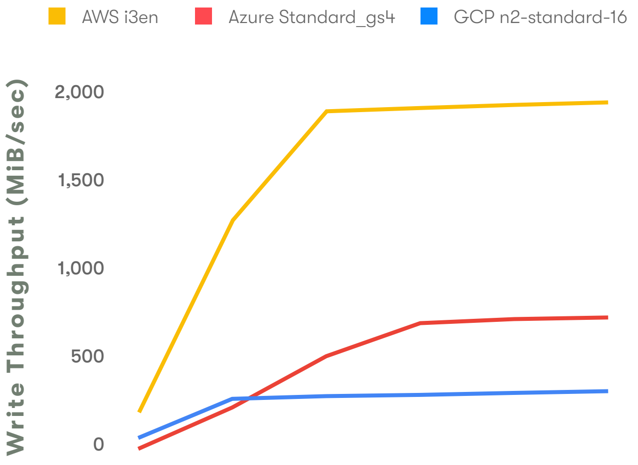
* Winning machines

Combined Storage Write Performance

After comparing all three clouds top performing machines, AWS offers superior write storage performance with the i3en machine type.

[FIG 25]

Storage: Maximum Write Throughput

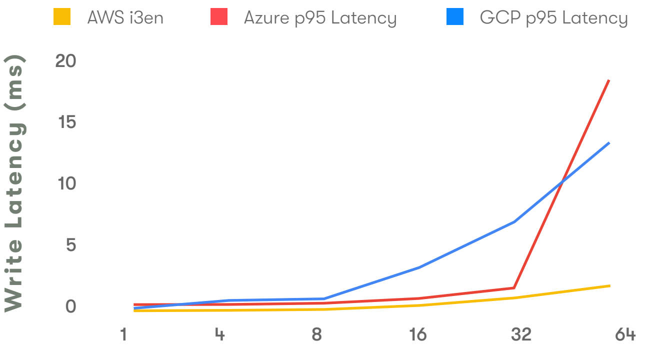


Threads

Both AWS and GCP appear to hit a bottleneck at 4 threads while Azure contains to increase write iOPs until 16 threads. For applications with more threads, Azure write iOPs really shine through after falling behind initially on smaller thread sizes.

[FIG 26]

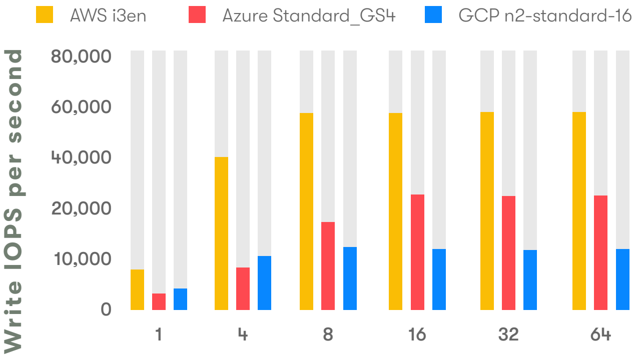
Storage: Write p95 Latency



Threads

[FIG 27]

Storage: Maximum Write iOPs



Threads

Cockroach Labs

About Storage I/O Read Performance

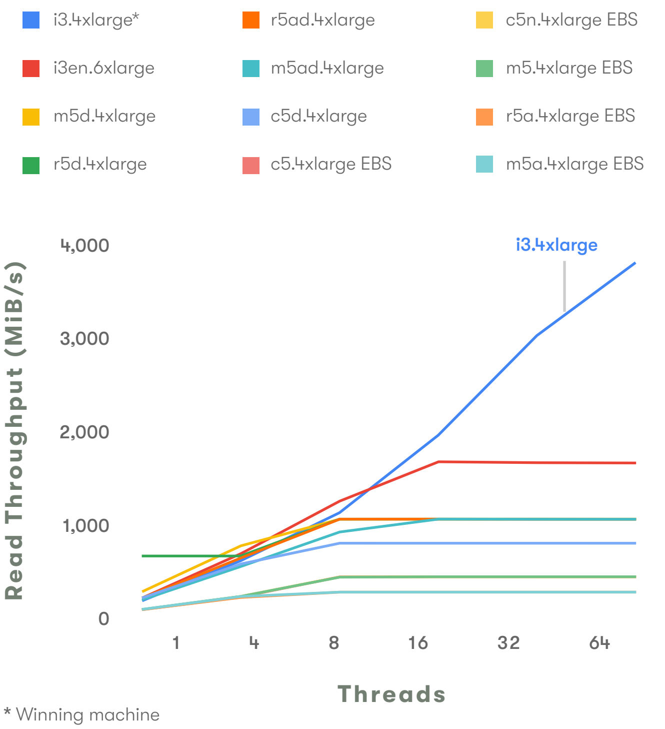
Similar to write performance, sysbench also measures how much data it can read from storage per second. We ran the benchmark over a range of concurrency levels, starting at 1 thread (reading alone) moving our way up to 64 threads (reading concurrently). This was really interesting to test, because different storage devices support different levels of concurrency.

AWS Storage I/O Read Performance

Unsurprisingly, AWS's storage optimized machines, the i3 series, again outperformed their other machine types. We did find it surprising that the i3en series underperformed the older i3 series on the storage read benchmarks (e.g., throughput, and latency as shown in FIG. 28).

[FIG 28]

Storage: AWS Read Throughput (MiB/s)

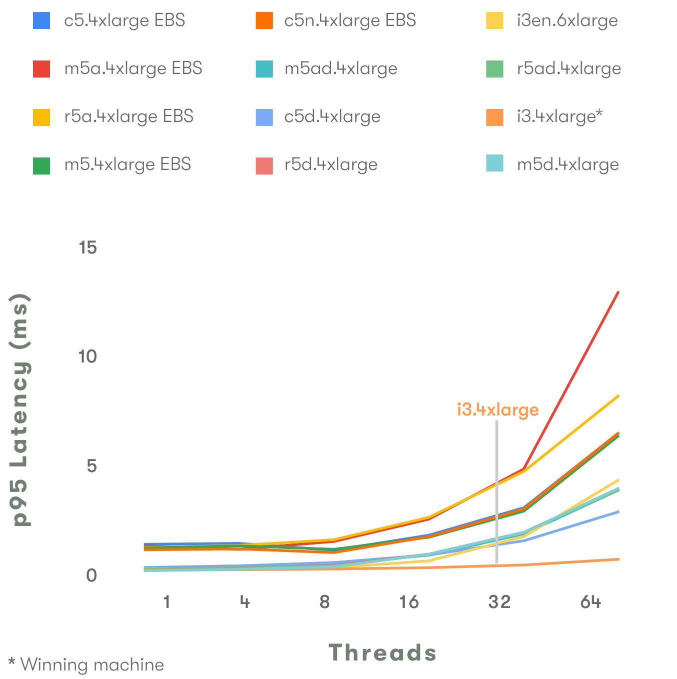


* Winning machine

In fact, the i3en is only middle of the pack as we review the p95 read latency. The c5d becomes the second best machine after the default i3 entry.

[FIG 29]

Storage: AWS p95 Read Latency

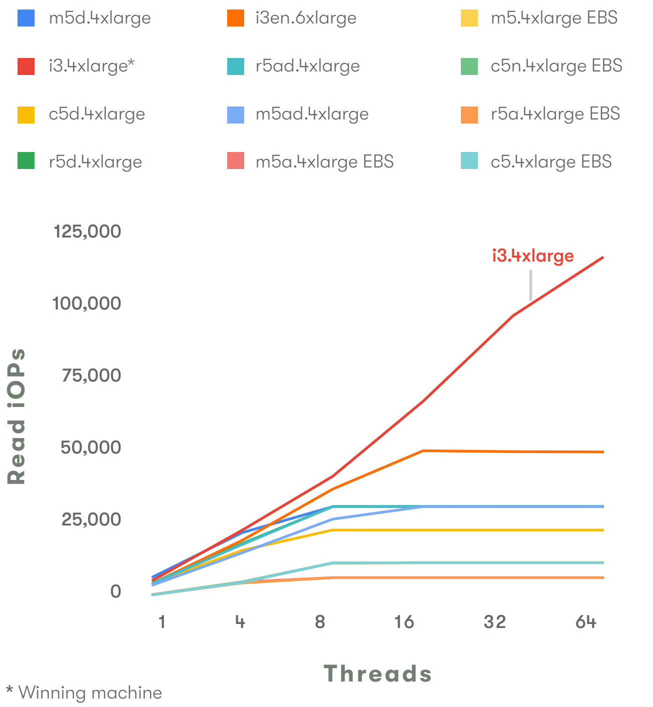


* Winning machine

Again, like on the write benchmarks, read IOPS mirrors read throughput. The main takeaway is that AWS's storage optimized machines live up to their billing as strong choices when optimizing for storage performance.

[FIG 30]

Storage: AWS Read iOPs



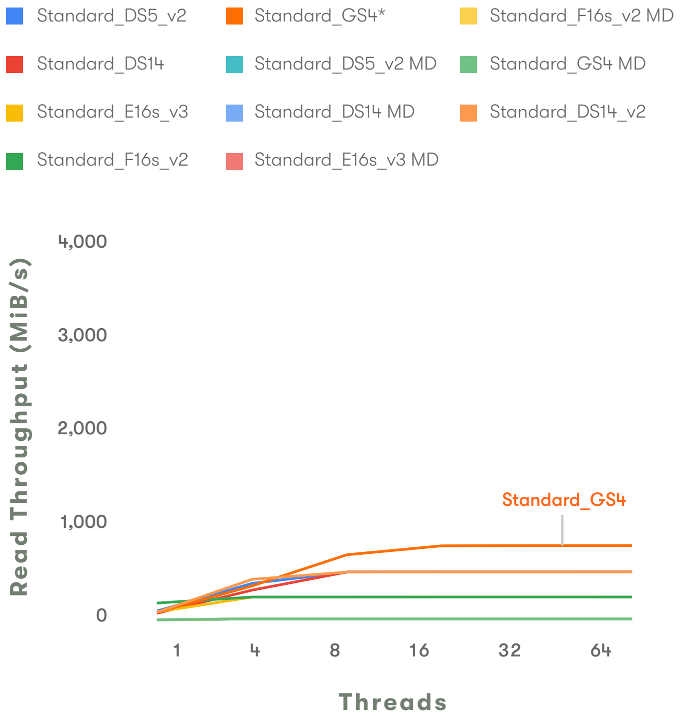
* Winning machine

Azure Storage I/O Read Performance

Azure's read throughput is similar to their write throughput. It's in the middle of the AWS spread but, even when excluding the storage optimized instances, can't reliably outperform AWS.

[FIG 31]

Storage: Azure Read Throughput (MiB/s)

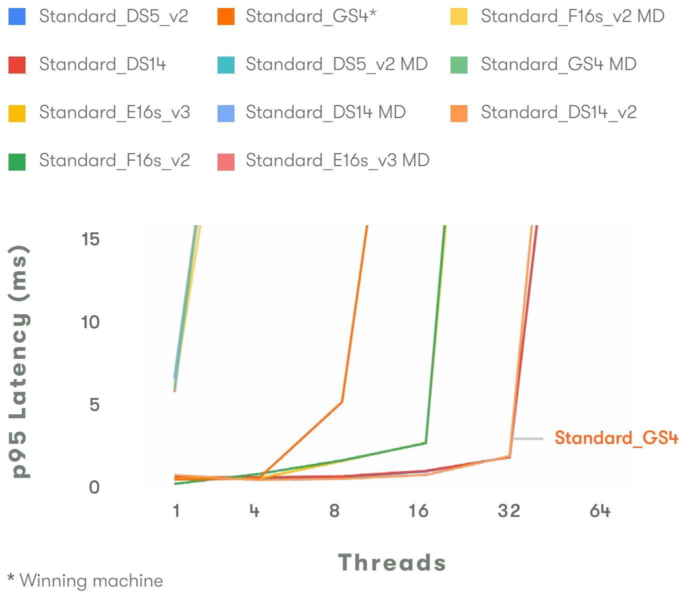


* Winning machine

Azure's read latency is extremely variable. It jumps quickly as threads increase. The standard_DS14 and Standard_DS5_v2 offer the best latency profile as they make it to 32 threads before sharply rising.

[FIG 32]

Storage: Azure p95 Average Read Latency



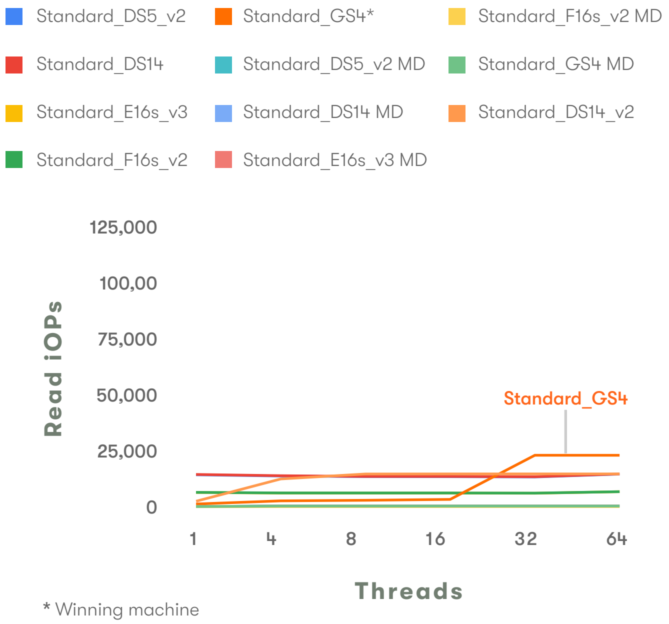
* Winning machine

Similarly to write latency, Azures MDs (and later SSDs) cannot perform at the same levels of AWS and GCP. We chose to hold the scale constant to the other clouds to better be able to make cross-cloud comparisons. Similarly to write latency, Azure consistently underperforms AWS and GCP on storage read latency.

Like with Azure throughput, the Standard_GS4 offers the best Read iOPS on Azure.

[FIG 33]

Storage: Azure Read iOPS



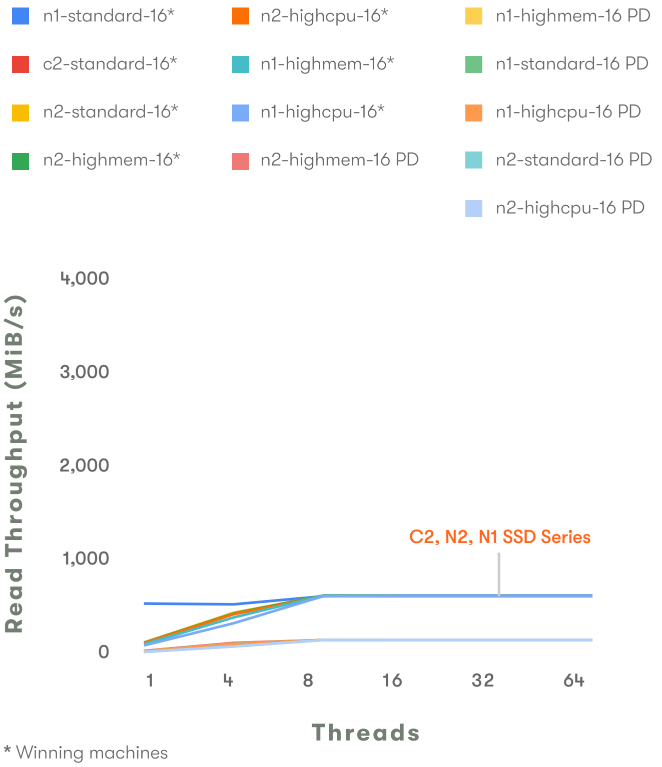
* Winning machine

GCP Storage I/O Read Performance

Just like in the write throughput benchmarks, all of GCPs SSDs outperform their PDs. While the spread is tightly grouped across most threads, it does appear as if the n1-standard-16 offers slightly better initial read throughput.

[FIG 34]

Storage: GCP Read Throughput (MiB/s)

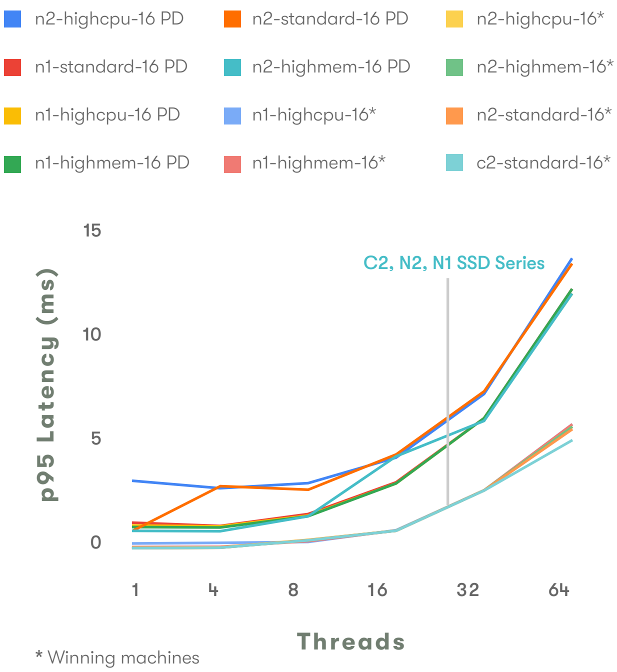


* Winning machines

Interestingly, unlike in the write latency charts, the SSDs also outperform all PDs in read latency.

[FIG 35]

Storage: GCP p95 Read Latency

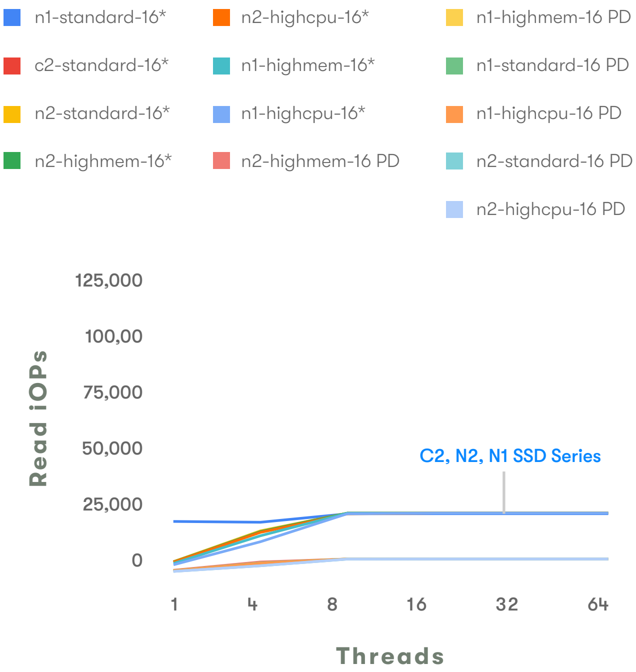


* Winning machines

Finally, Read IOPS match Read throughput.

[FIG 36]

Storage: GCP Read iOPs



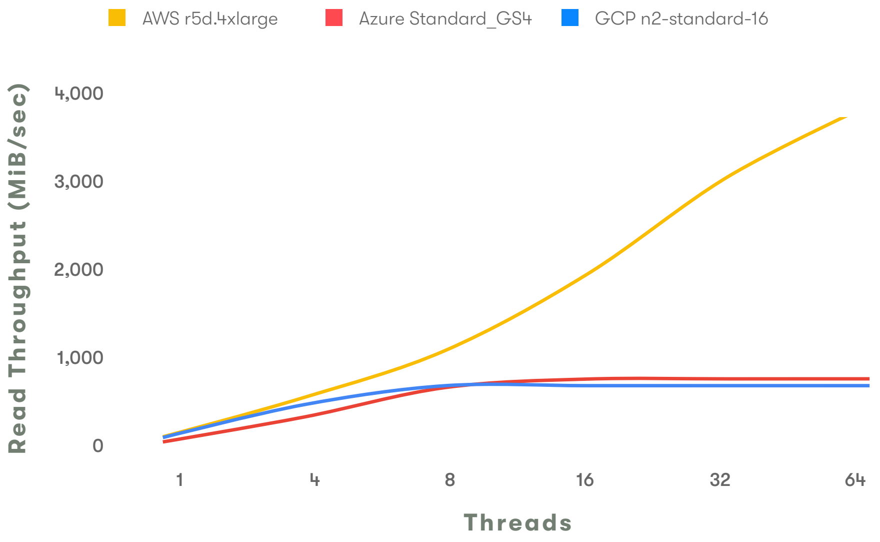
* Winning machines

Combined Storage Read Performance

Similarly to the combined storage write performance, AWS wins across all categories with its i3 machine type. We chose to test workload performance by using TPC-C, a popular OLTP benchmark tool that simulates an e-commerce business, given our familiarity with this workload. TPC-C is a popular OLTP benchmark tool that simulates an e-commerce business with a number of different warehouses processing multiple transactions at once. It can be explained through the above microbenchmarks, including CPU, network, and storage I/O.

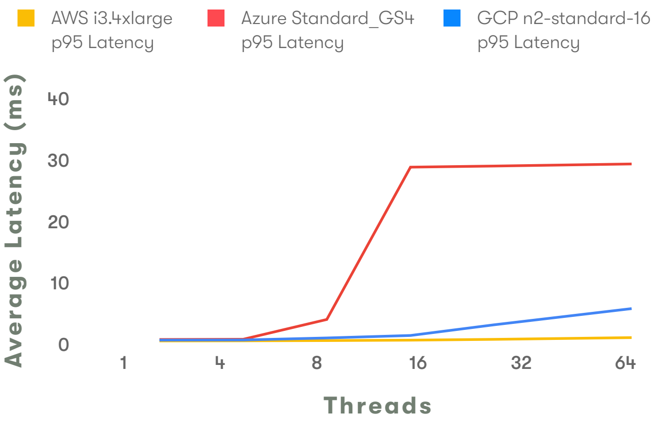
[FIG 37]

Storage: Read Throughput



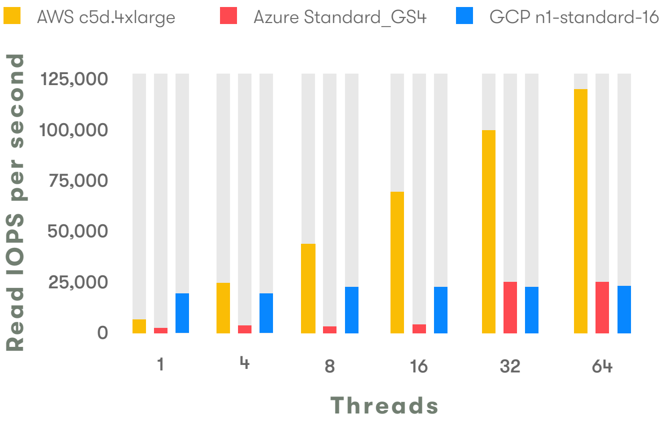
[FIG 38]

Storage: Minimum Read Latency



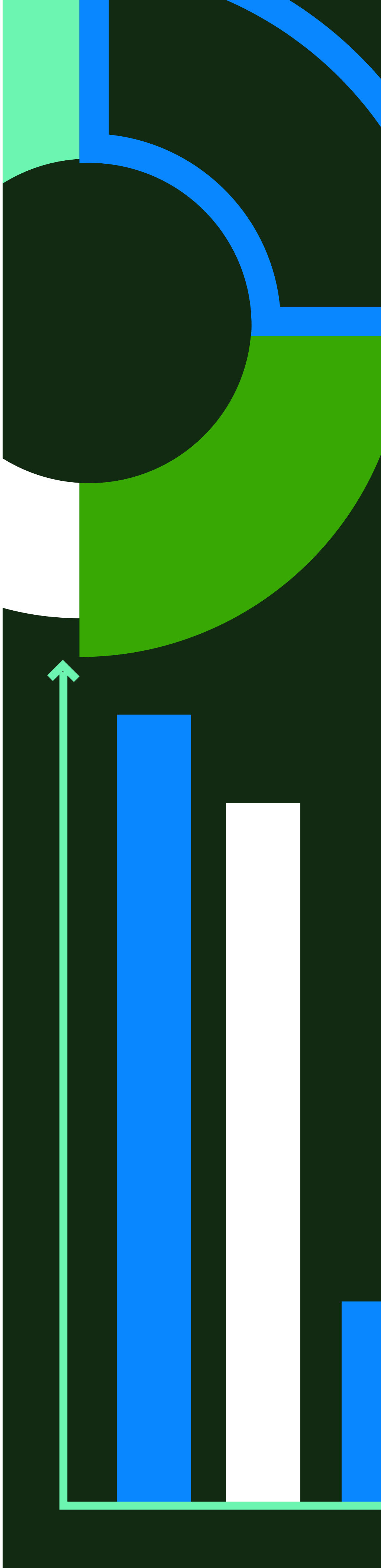
[FIG 39]

Storage: Read iOPs



Google's SSDs consistently outperform PDs in all storage metrics.

TPC-C Performance



We chose to test overall workload performance by using TPC-C, given our affinity for and familiarity with this workload. TPC-C is a popular OLTP benchmark tool that simulates an e-commerce business with a number of different warehouses processing multiple transactions at once. It can be explained through the above microbenchmarks, including CPU, network, and storage I/O.

TPC-C is measured in two different ways. One is a throughput metric, throughput-per-minute type C (tpmC) (also known as the number of orders processed per minute). The other metric is the total number of warehouses supported. Each warehouse is a fixed data size and has a max amount of tpmC it's allowed to support, so the total data size of the benchmark is scaled proportionally to throughput. For each metric, TPC-C places latency bounds that must be adhered to in order to consider a run "passing". Among others, a limiting passing criteria is that the p90 latency on transactions must remain below 5 seconds. This allows an operator to take throughput and latency into account in one metric. Here, we consider the maximum tpmC supported by CockroachDB running on each system before the latency bounds are exceeded.

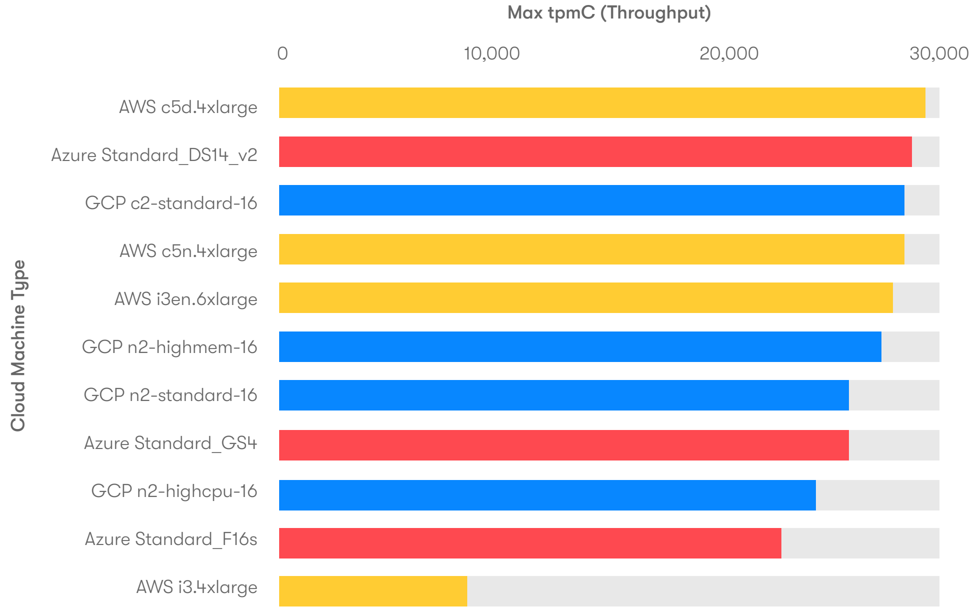
In 2017, our internal testing suggested more equitable outcomes between AWS and GCP. In 2018, AWS outperformed GCP by 40%. We attributed this to AWS's Nitro System present in `c5` and `m5` series. Did this hold true in the 2020 report?

In 2019, we saw that AWS came across on top on this benchmark once again, but that GCP made tremendous strides to close the gap between itself and AWS. Azure performed similarly to the top two with its best machines. All clouds are within 5% of one another.

Interestingly, the highest performing machine types from each cloud are also the same machine types which performed the best on the CPU and Network Throughput tests. Both AWS's c5n.4xlarge and GCP's c2-standard-16 won the CPU, Network Throughput, and Network Latency tests while Azure's Standard_DS14_v2 won the CPU and Network Throughput throughput tests. However, the machine types which performed best on the read and write storage tests (e.g., AWS i3.4xlarge and i3en.6xlarge, GCPs n2-standard-16, and Azure's Standard_GS4) varied in their TPC-C performance. This suggests that these tests are less influential in determining OLTP performance. These results match our expectation that OLTP workloads like TPC-C are often limited by compute resources due to their relatively high ratio of transactions to data size.

[FIG 40]

2020 TPC-C Results by Cloud



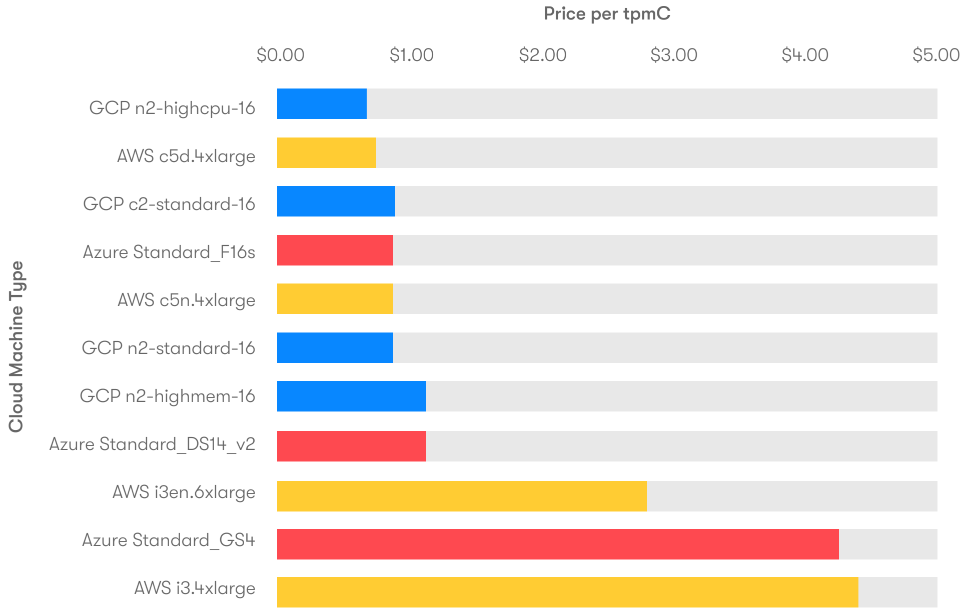
TPC-C Performance per Dollar

Efficiency matters as much as performance. If you can achieve top performance but have to pay 2x or 3x, it may not be worth it. For this reason, TPC-C is typically measured in terms of price per tpmC. This allows for fair comparisons across clouds as well as within clouds. In this analysis, we use the default on-demand pricing available for each cloud because pricing is an extremely complex topic. GCP, in particular, was keen to note that a true pricing comparison model would need to take into account on-demand pricing, sustained use discounts, and committed use discounts. While is true that there is a high cost associated with paying up-front costs, we applied this evenly across all three clouds.

We recommend exploring various permutations of these pricing options depending upon your workload's requirements. Producing a complex price comparison across each cloud would be a gigantic undertaking, in and of itself, and we believe that Cockroach Labs is not best positioned to offer this kind of analysis. Finally, we are reporting the raw TPC-C performance numbers above because we are also aware that, depending upon the size of your organization, you may be able to negotiate discounts not available from the list prices on each vendor's website.

[FIG 41]

Price per tpmC by Cloud Machine Type



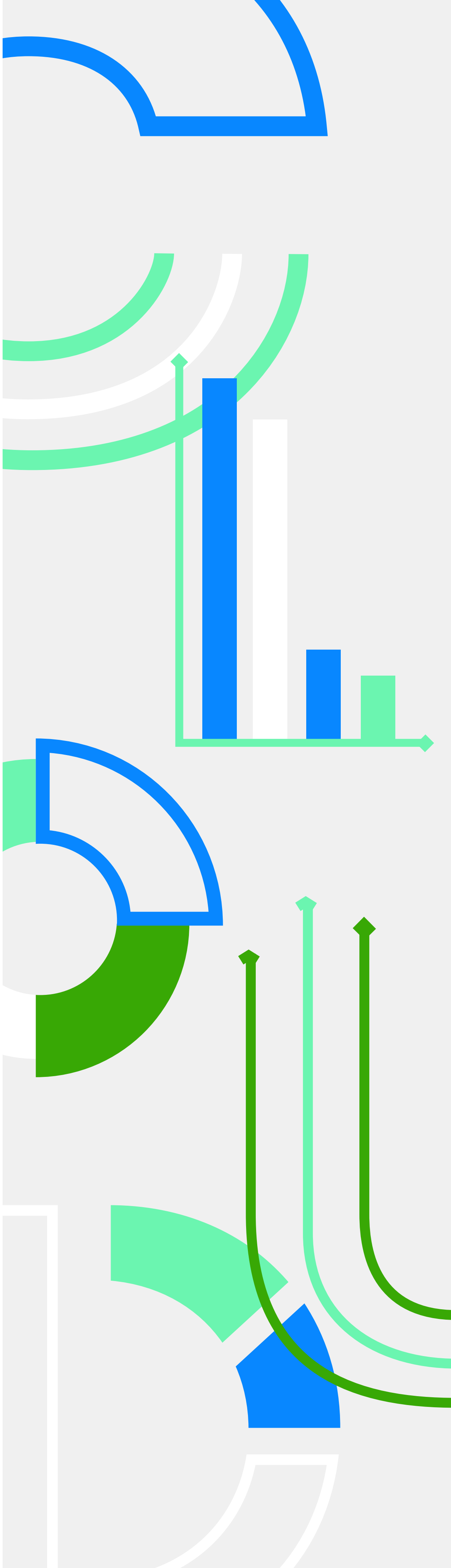
Again, all three clouds come close on the cheapest price per tpmC. However, this year we see that the GCP n2-highcpu-16 offers the best performance per dollar in the tested machine types. If price is less of a concern, AWS is the best performer on throughput alone.

Conclusion

GCP shows dramatic improvement in the 2020 Cloud Report edging out AWS and Azure on price per performance of TPC-C but slightly underperforming AWS and Azure on max tpmC available on a three node cluster.

Setting up a highly performant configuration isn't always intuitive. It's also important to note that over the past couple years of testing, we've seen different cloud providers performance change drastically. Since these results fluctuate as the clouds adopt new hardware, it's important to regularly re-evaluate your configuration (and cloud vendor).

CockroachDB remains committed to our stance as a cloud-agnostic database. We will continue to use AWS, Azure, GCP, and others for internal stability and performance testing. We also expect that these results will change over time as all three companies continue to invest in a modern infrastructure ecosystem.



CockroachLabs is the company behind CockroachDB, the ultra-resilient SQL database.

With a mission to Make Data Easy, Cockroach Labs is led by a team of former Google engineers who have had front row seats to nearly two decades of database evolution. The company is headquartered in New York City and is backed by an outstanding group of investors including Benchmark, G/V, Index Ventures, Redpoint, and Sequoia.

cockroachlabs.com