Cockroach LABS

2018 Cloud Report

Written by **Masha Schneider** & **Andy Woods**

Table of Contents

Introduction

Our customers rely on us to help them navigate the complexities of the increasingly competitive cloud wars. Should they use Amazon Web Services (AWS)? Google Cloud Platform (GCP)? Microsoft Azure? How should they tune their workload for different offerings? Which is more reliable?

We are committed to building a cloud neutral product, and we run test clusters on all three leading US cloud providers. As we were testing features for our 2.1 release, we noticed something interesting: AWS offered 40% greater throughput than GCP.

We were curious as to why AWS offered such a stark difference in throughput, and set out to test the performance of GCP and AWS in more detail. Ultimately, we compared the two platforms on TPC-C performance (e.g., throughput and latency), CPU, network, I/O, and cost.

This inspired what has become the 2018 Cloud Computing Report.

Our conclusion? AWS outperforms Google on nearly every criteria we tested — including cost.

NOTE: We didn't test Microsoft Azure due to bandwidth constraints but plan to do so in the near future.

Machine Type

GCP has a variety of instance types (including standard and high CPU) but we focused on the `n1-standard-16` machine with Intel Xeon Scalable Processor (Skylake) in the us-east region (Skylake offers a marginal 4% improvement over standard hardware on `n1-standard-16`). We were familiar with this instance type as we used it to conduct our previous performance benchmarking.

A similar configuration isn't quite as trivial as it sounds for AWS. AWS has more flavors of instances than GCP. It has the standard high cpu and general instances. We chose the latest compute-optimized AWS instance type, c5d.4xlarge instances, to match n1-standard-16, because they both have 16 CPUs and SSDs (although AWS only offers 32 gb of RAM as compared to 60 gb of RAM on GCP) within the us-east-2 region.

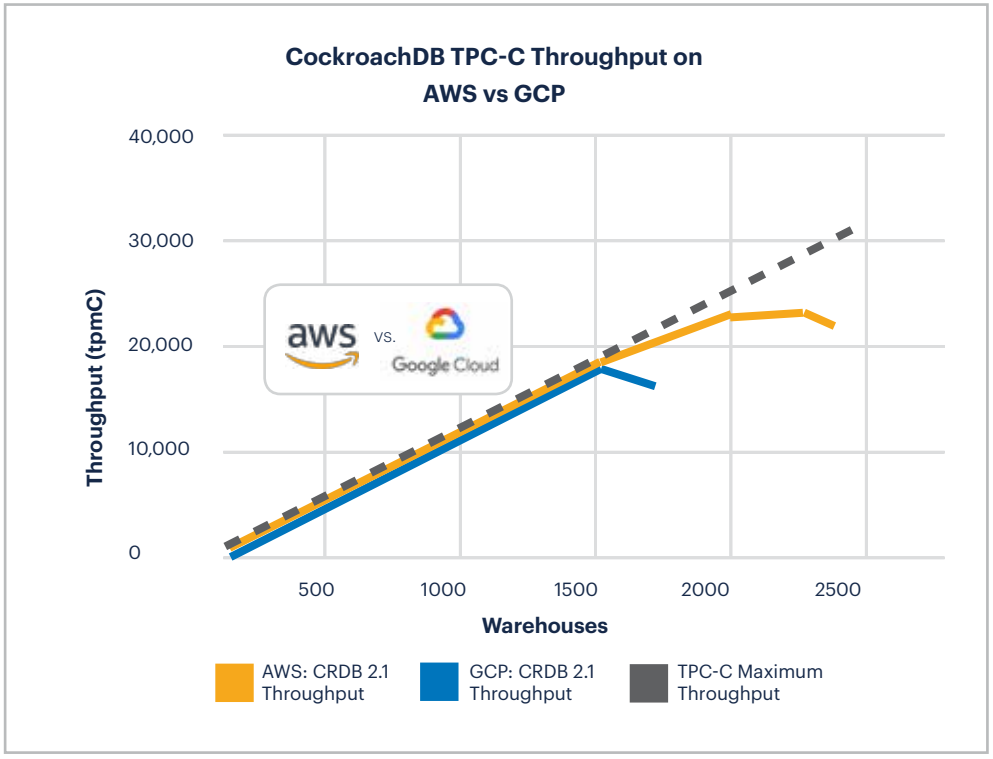
Experiments

We designed our first experiment to tease out whether or not AWS and GCP performance differed on a simulated customer workload. We started with a customer workload (and not micro-benchmarks) because it most directly simulates real-world customer behavior.

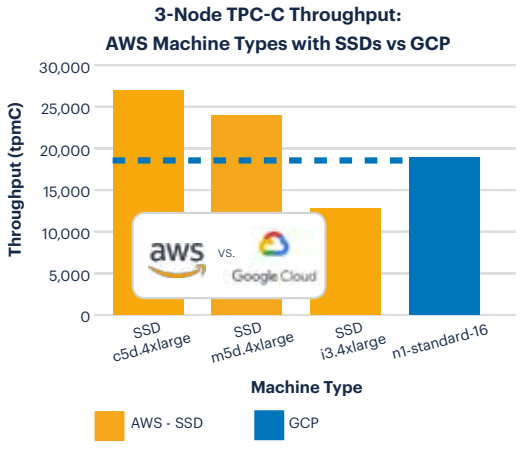
It was only after observing differences in applied workloads that we moved onto micro-benchmarks like CPU, network, and I/O performance. Differences in micro-benchmarks matter more when informed by the knowledge that the overall customer workload performance of the platforms differ. CPU, network, and I/O all represent separate hypothesis for why performance might vary between GCP and AWS.

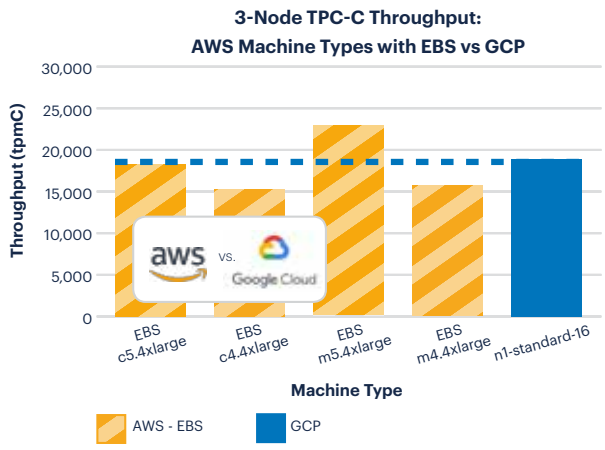
TPC-C Performance

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.



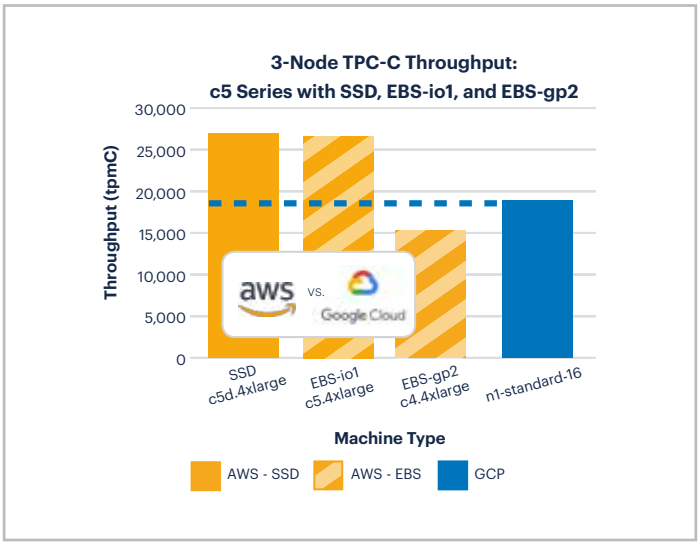
CockroachDB 2.1 achieves 40% more throughput (tpmC) on TPC-C when tested on AWS using `c5d.4xlarge` than on GCP via `n1-standard-16`. We were shocked that AWS offered such superior performance. Previously, our internal testing suggested more equitable outcomes between AWS and GCP. We decided to expand beyond the `c5` series to test TPC-C against some of the most popular AWS instance types.



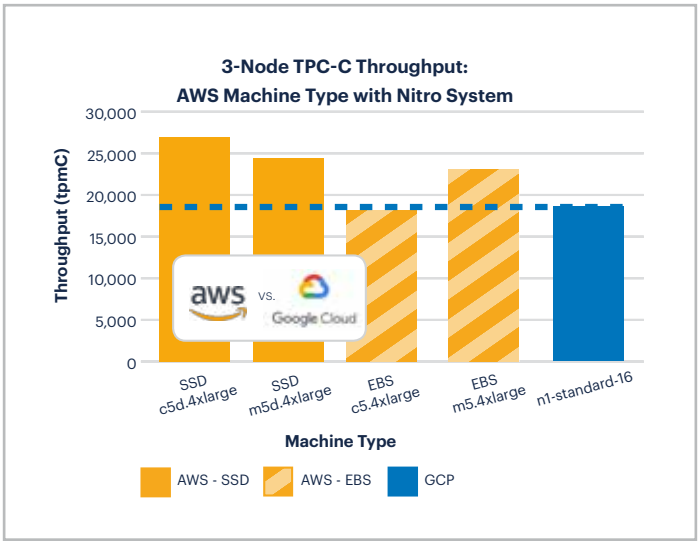


At first blush, it appears that SSDs offered by `c5d` and `m5d` outperform EBS. Unfortunately, it's a bit more complicated than that as AWS offers EBS out of the box with gp2 volume types rather than the higher performing io1 volume type.

To isolate this change, we focused on the higher performing `c5` series with SSDs, EBS-gp2, and EBS-io1 volume types:



Clearly, EBS volumes offer effective performance if tuned to the io1 volume type and provided with sufficient iOPS. So if the difference isn't explained by SSD vs. EBS, what else might explain it? AWS recently introduced their new Nitro System present in `c5` and `m5` series. The AWS Nitro System offers approximately the same or superior performance when compared to a similar GCP instance.



The results were clear: AWS wins on TPC-C benchmark performance. But what causes such large performance differentials? We set out to learn more by testing a series of microbenchmarks on CPU, network, and I/O.

CPU Experiment

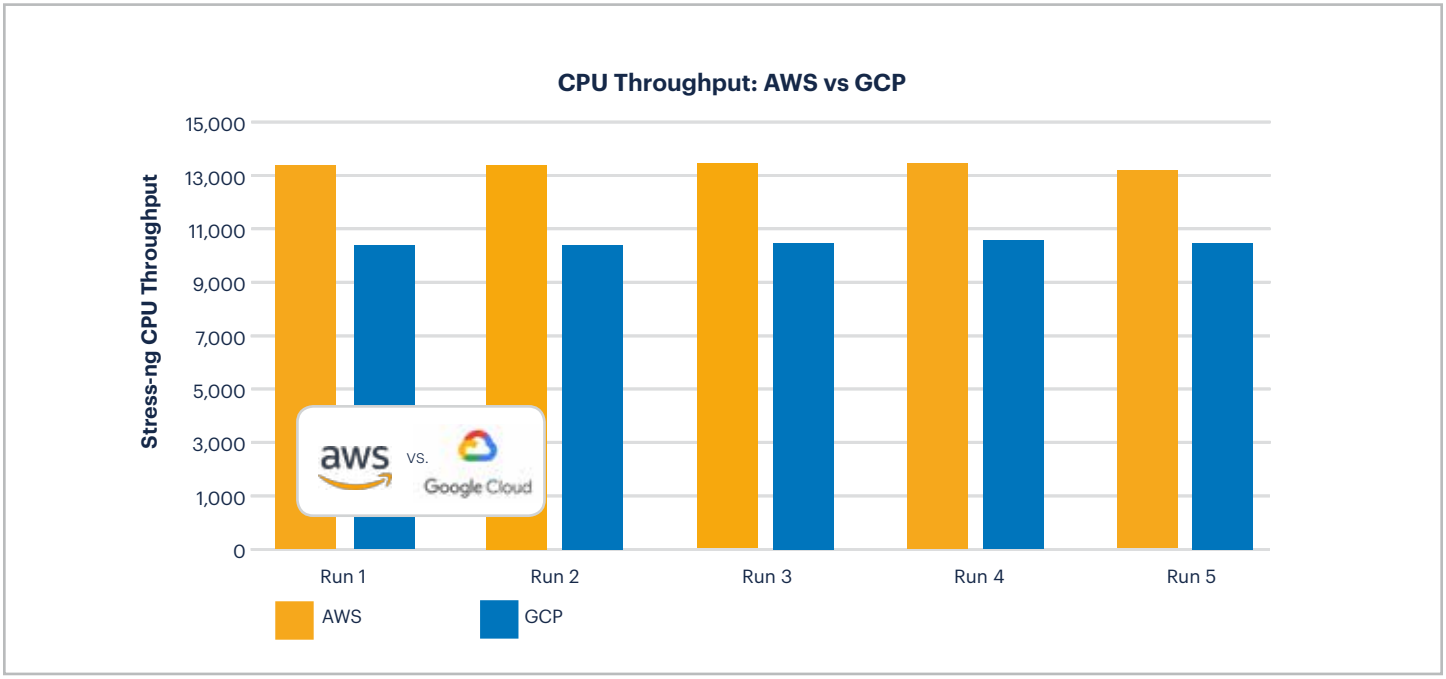
We began our micro-benchmark testing like any aspirational scientists by seeking to disprove our main hypothesis: that differences in AWS and GCP provisions might affect CPU performance.

We focused on a CPU performance microbenchmark first as CPU can have a large impact on performance.

To test CPU performance, we evaluated third party benchmarks based on popularity and ease of use. The two most frequently used benchmark test suites in the market today are sysbench and stress-ng. We chose stress-ng because it offered more benchmarks and provided more flexible configurations than sysbench.

We ran the following Stress-ng command five times on both AWS and GCP:

```
stress-ng --metrics-brief --cpu 16 -t 1m
```



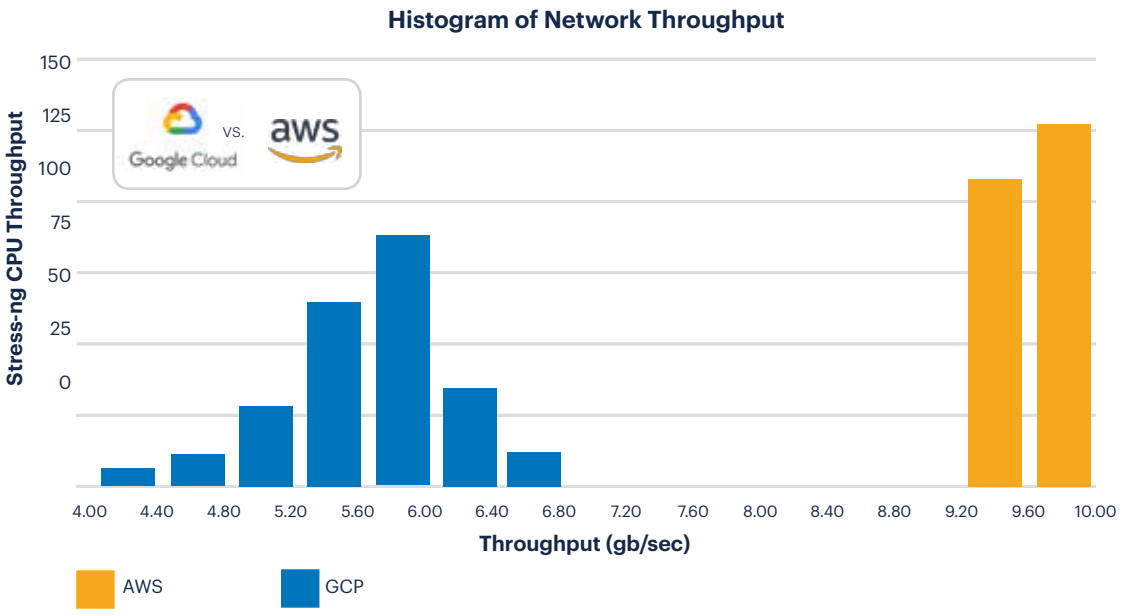
AWS offered 28% more throughput (~2,900) on stress-ng than GCP. Both AWS and GCP offered generally consistent CPU performance across runs. This is a credit to the investments made by both platforms as unpredictability can have a material cost for business paid in the over-provisioning of virtual machines.

Now that we observed an initial difference in both CPU performance on GCP and AWS, we couldn't help ourselves from continuing to investigate other potential differences. Was the entirety of the TPC-C difference generated from the advantage in CPU performance?

Network Experiment

Next, we tested network throughput and latency. To test network, we measured throughput using a popular tool called iPerf and latency via another popular tool PING.

iPerf's configurations include a buffer data size (128kb), a protocol, a server and and a client. iPerf attempts to connect the client and the server with the data from buffer size via the protocol. We setup iPerf similarly to this blog post. This test provides a throughout for the network which allows for us to compare the performance of the network on AWS and GCP. We ran the test four times each for AWS and GCP and aggregated the results of all four tests in histograms (each 1 sec run is stacked to form this chart):



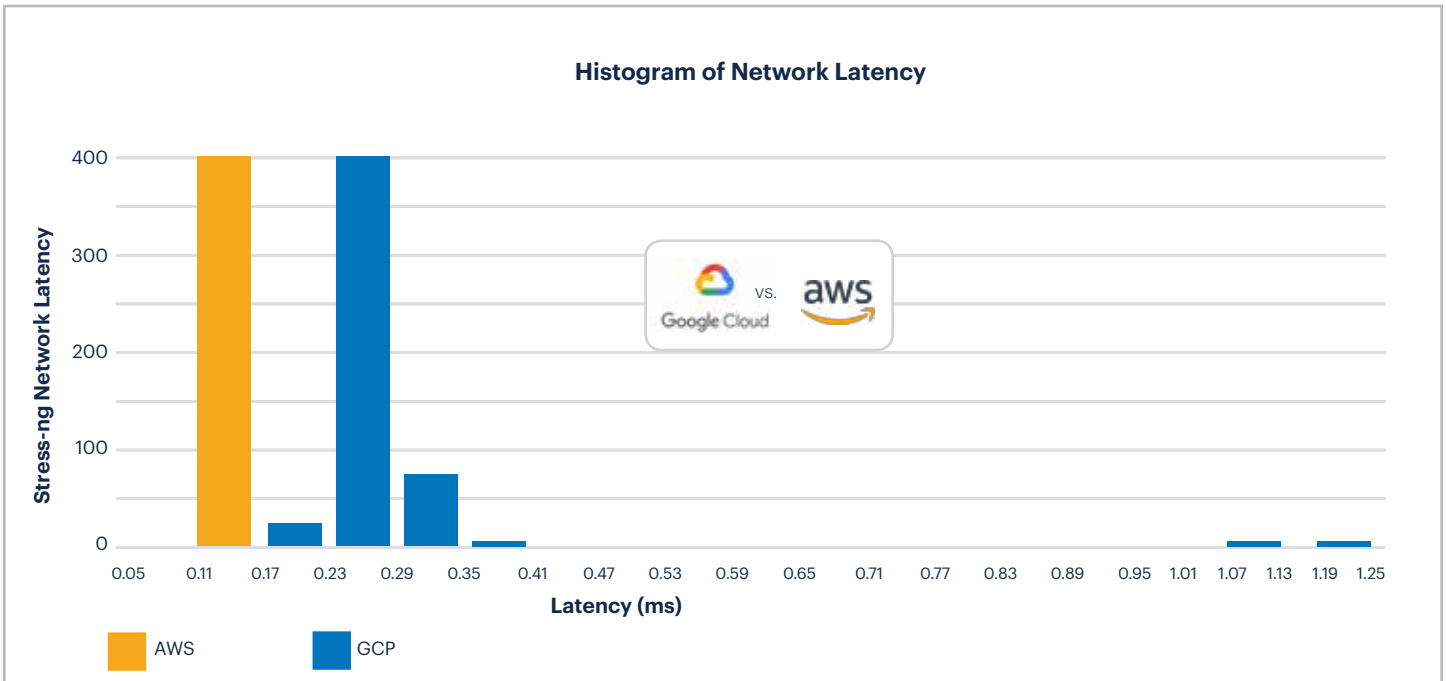
	SUM of Throughput		AVERAGE of Throughput		STDEV of Throughput		MIN of Throughput		MAX of Throughput	
RUN	GCP	AWS	GCP	AWS	GCP	AWS	GCP	AWS	GCP	AWS
1	333.21	576.43	5.55	9.61	0.459	0.007	4.01	9.60	6.31	9.63
2	334.00	576.48	5.57	9.61	0.312	0.007	4.72	9.60	6.11	9.63
3	329.18	576.25	5.49	9.60	0.593	0.005	4.03	9.60	6.60	9.62
4	349.45	566.67	5.82	9.60	0.482	0.005	4.64	9.60	6.67	9.62
Grand Total	**1345.84**	**2295.83**	**5.61**	**9.61**	**0.487**	**0.006**	**4.01**	**9.60**	**6.67**	**9.63**

GCP shows a fairly normal distribution of network throughput centered at ~5.6 gb/sec. In addition to the raw network throughput, we also care about the variance of network throughput as unpredictable performance can cost businesses money and time. Throughput ranges from 4.01 gb/sec to 6.67 gb/sec — a somewhat unpredictable spread of network performance, reinforced by the observed average variance for GCP of 0.487 gb/sec.

AWS, on the other hand, stands out as it offers significantly higher throughput, centered on 9.6 gb/sec, while providing a much tighter spread between 9.6 gb/sec and 9.63 gb/sec when compared to GCP. On AWS, iPerf transferred a total network throughput of 2,296 gb. This is an increase of 70% over GCP which is, on average, more than 4 gb/sec increase in throughput.

What about network throughput variance? On AWS, the variance is only 0.006 gb/sec. **This means that the GCP network throughput is 81x more variable when compared to AWS.**

We tested network latency, in addition to the throughput and variance. Without testing for latency, we can miss significant delays in service that may be masked by overall performance. We used the industry standard tool PING to measure latency.



	AVERAGE of Latency		STDEV of Latency		MIN of Latency		MAX of Latency	
Run	AWS	GCP	AWS	GCP	AWS	GCP	AWS	GCP
1	0.056	0.227	0.003	0.106	0.052	0.164	0.072	1.250
2	0.057	0.217	0.003	0.090	0.054	0.153	0.077	1.080
3	0.057	0.221	0.003	0.094	0.051	0.167	0.075	1.120
4	0.057	0.227	0.003	0.106	0.053	0.162	0.074	1.240
Grand Total	**0.057**	**0.223**	**0.003**	**0.099**	**0.051**	**0.153**	**0.077**	**1.250**

At first glance it appears that GCP has a tighter latency spread (when compared to the network throughput) centered on 0.2 ms. After looking a bit more closely we can see that there are several outliers such that the max latency, 1.25 ms, is more than 5 times the average!

Similarly, to network throughput, AWS offers a stark difference to GCP.

AWS's values are centered on an average latency, 0.057 ms. **In fact the spread is so tight it can't be visualized on the same scale as GCP.** The max latency is only 0.077 ms — a difference of only .02 ms (or 35%) from the average!

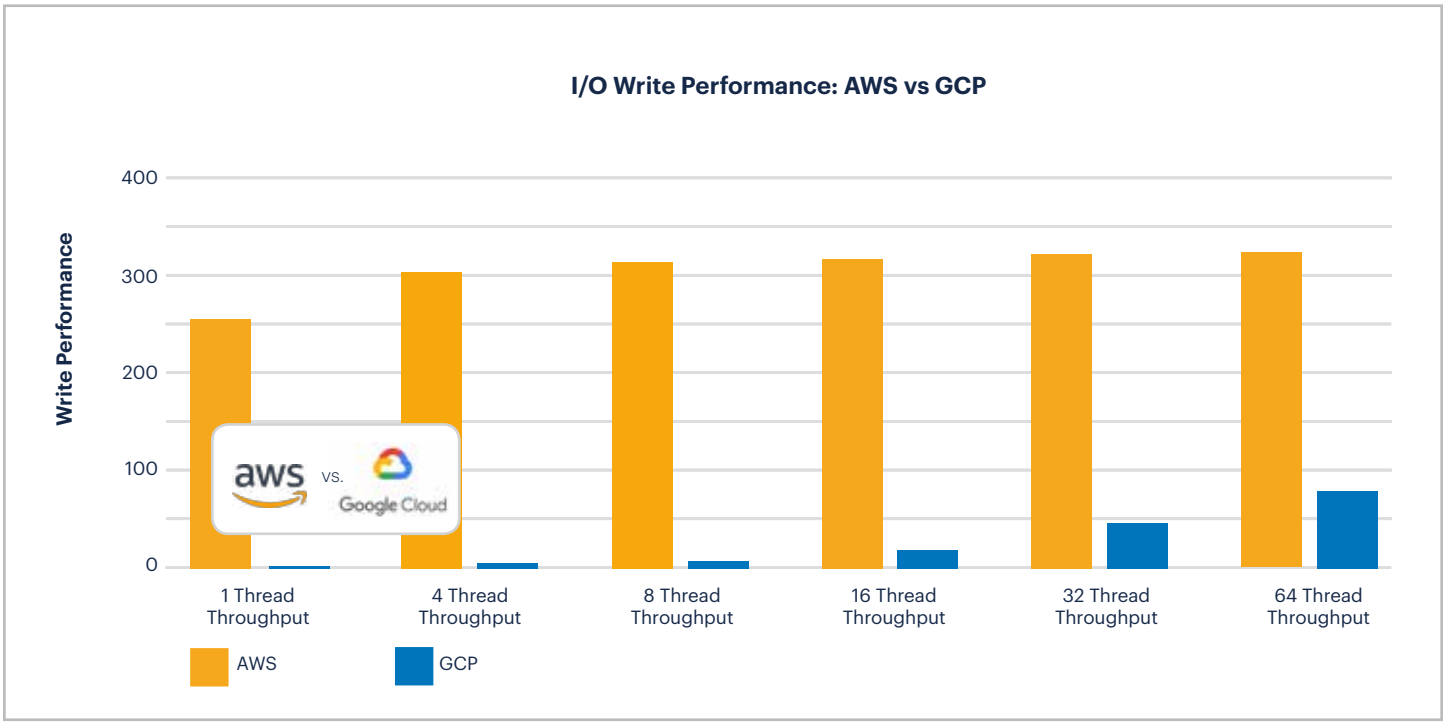
AWS offers significantly better network throughput and latency with none of the variability present in GCP. Further, it looks like Amazon may be racing further ahead in network performance with the introduction of the `c5n` machine type that offers significantly higher network performance across all instance sizes as compared to the rest of the c series.

I/O Experiment

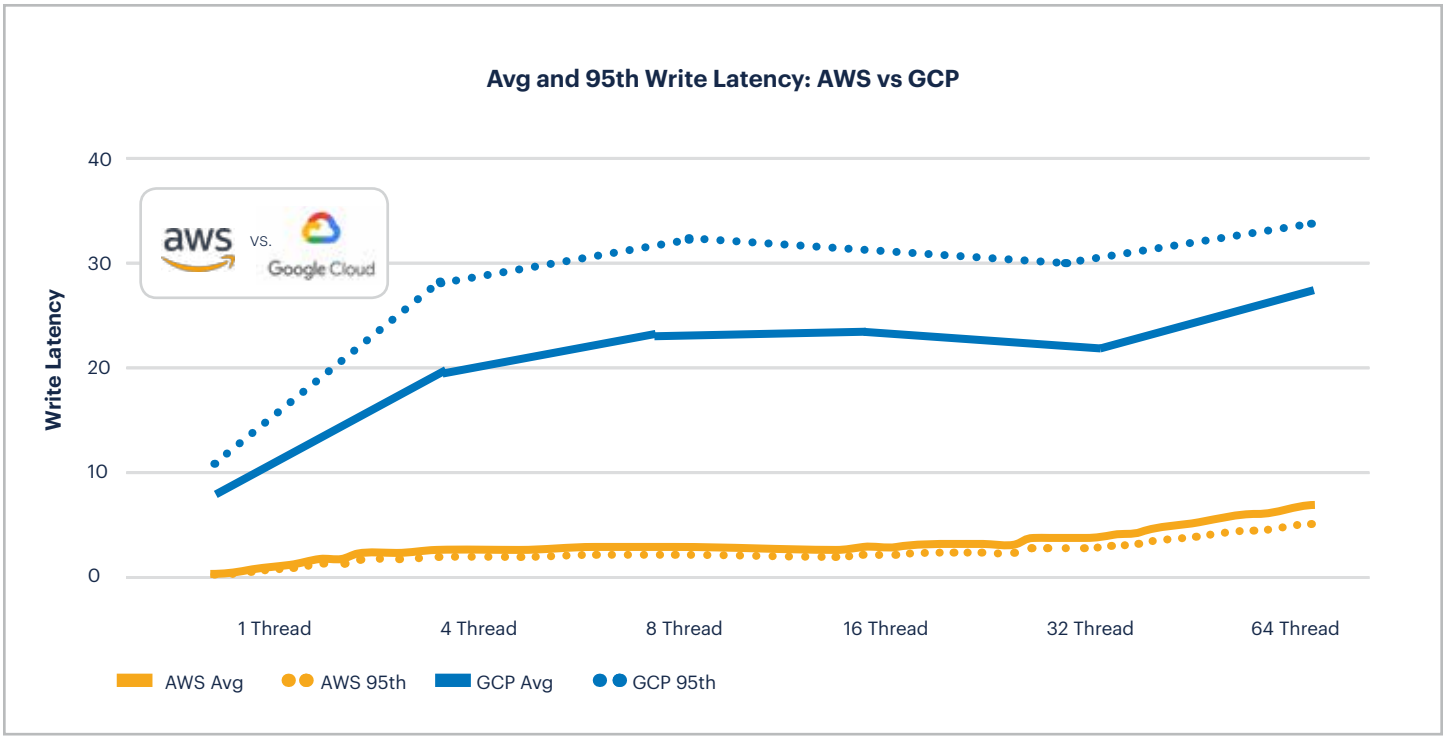
We tested I/O using a configuration of Sysbench that simulates small writes with frequent syncs for both write and read performance. We ran the sysbench test writing to an SSD to achieve similar results to running a database in production.

This test measures throughput based on a fixed set of threads, or the number of items concurrently writing to disk.

First, we tested write performance:

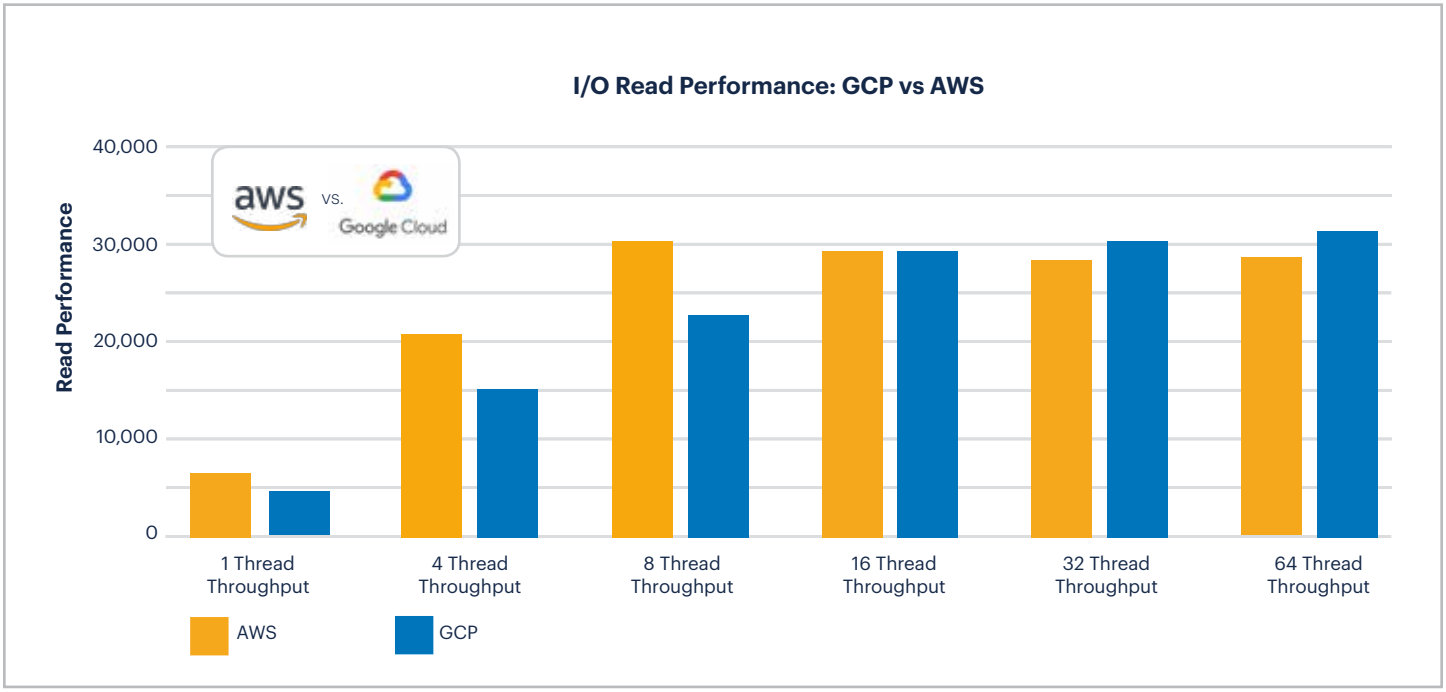


AWS consistently offers more write throughput across all thread variance from 1 thread up to 64. In fact, it can be as high as 67x difference in throughput.

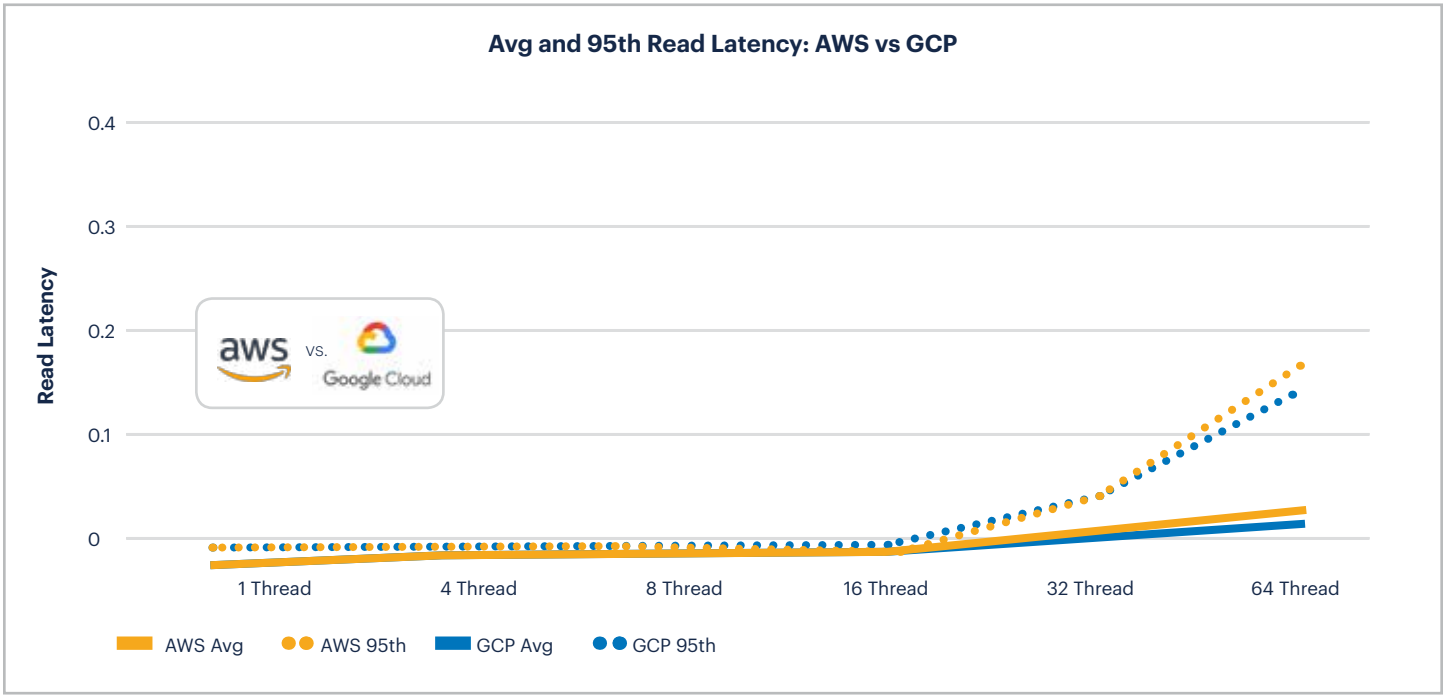


AWS also offers better average and 95th percentile write latency across all thread tests.

When we evaluated for write latency and throughput, AWS continued to hold clear advantages.



AWS provides more read throughput from 1 to 16 threads. At 32 and 64 threads, GCP provides marginally more throughput.



AWS provides more read throughput from 1 to 16 threads. At 32 and 64 threads GCP provides marginally more throughput. Similarly to write latency, AWS wins the read latency battle up to 32 threads. At 32 and 64 threads GCP and AWS split the results.

Overall, AWS wins for write and read performance up to 16 threads. GCP offers a marginally better performance with similar latency to AWS for read performance at 32 threads and up.

No Barrier

What about performing this experiment with no barrier? As a refresher, no barrier is a method of writing directly to disk without waiting for the write cache to be flushed. This is faster, but in the event of a crash, data can be corrupted. Note that no barrier can be safely used with battery-backed write caches. We were curious how big a performance advantage this offered so we tested no barrier on GCP and AWS and saw large changes in performance.

Compared to not setting no barrier on GCP, GCP with no barrier speeds things up by 6x! On AWS, no barrier (vs. not setting no barrier) is only a 25% speed up. As such, we decided to re-run the entirety of the above experiment with no barrier. Even with the benefit of no barrier, the head-to-head results remain largely unchanged. And, since running with no barrier offers some additional risk, we think it reinforces our original conclusion that AWS beats GCP on I/O performance.

Cost

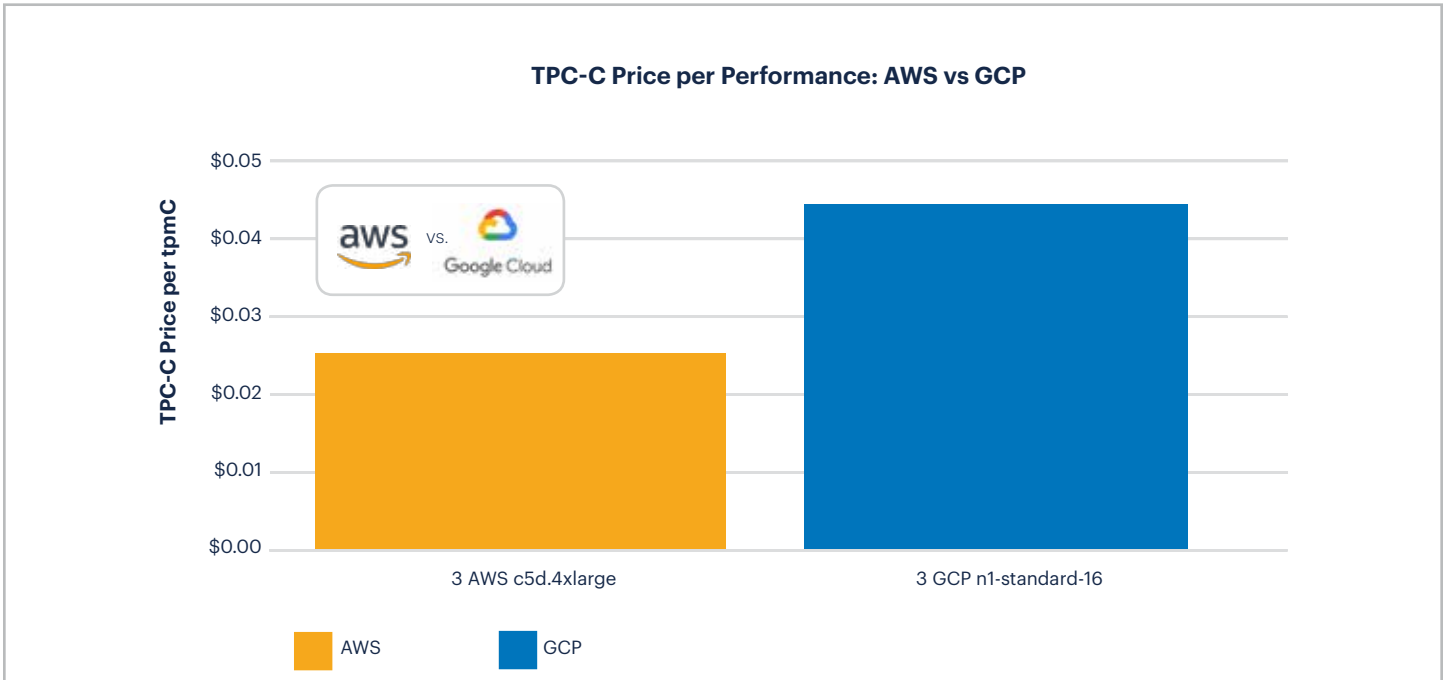
On applied benchmarks (e.g., TPC-C) and the more descriptive micro-benchmarks — CPU, network, and I/O — AWS outperformed GCP. But at what cost? Do you pay for this increased performance on AWS?

Let's circle back to the TPC-C setup discussed at the beginning. For TPC-C, we used `n1-standard-16` on GCP with local SSD and `c5d.4xlarge` on AWS. For both clouds we assumed the most generous discounts available:

On GCP we assumed a three-year committed use price discount with local SSD in the central region.

On AWS we assumed a three-year standard contract paid up front.

Not only is GCP more expensive than AWS, but it also achieves worse performance. This is doubly reflected in the price per performance (below), which shows **GCP costing 2.5 times more than AWS per tpmC** (the primary metric of throughput in TPC-C)!



Machine Name	CPUs	Ram (gb)	TPC-C Throughput (tpmC)	Price per Month	Price per tpmC
3 GCP n1-standard-16	16	60	18,568.90	$839.02	$0.05
3 AWS c5d.4xlarge	16	32	26,079.00	$606.63	$0.02

Conclusion

AWS outperformed GCP on applied performance (e.g., TPC-C) and a variety of micro-benchmarks (e.g, CPU, network, and I/O) as well as cost. We recommend using AWS for your most important workloads.

CockroachDB remains committed to our stance as a cloud-agnostic database. We will continue to use GCP, AWS, Microsoft Azure, 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 the modern infrastructure ecosystem.

Note, the 2018 Cloud Computing Report focused on evaluating AWS and GCP because they are the most popular cloud platforms among our customers. In future editions, we plan to expand upon our testing with Microsoft Azure, Digital Ocean, and other cloud platforms.

CockroachLabs is the company behind CockroachDB, the SQL database for building global cloud services.

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.

Run your business on a database built right.
Learn more at:
www.cockroachlabs.com