MariaDB

EVALUATING MARIADB XPAND AND COCKROACH WITH SYSBENCH

MARCH 2022

WHITEPAPER

TABLE OF CONTENTS

INTRODUCTION

This paper expands on the performance comparison between MariaDB Xpand and CockroachDB previously provided in the "MariaDB Xpand Crunches CockroachDB with Sysbench" blog post. This paper details the performance results of each database's respective cloud offering using Amazon Web Services (AWS), and takes a deeper dive into the data, looks at latency as a factor, analyzes price/performance metrics and explains how to reproduce the results.

Distributed SQL databases offer the promise of consistency, a standard query language, cloud scale, high availability and disaster recovery. These databases evolved from standard relational databases and NoSQL databases combined with new algorithmic technology. They are relatively new on the market and published performance data is sparse. MariaDB hopes to change this by providing fair and transparent results.

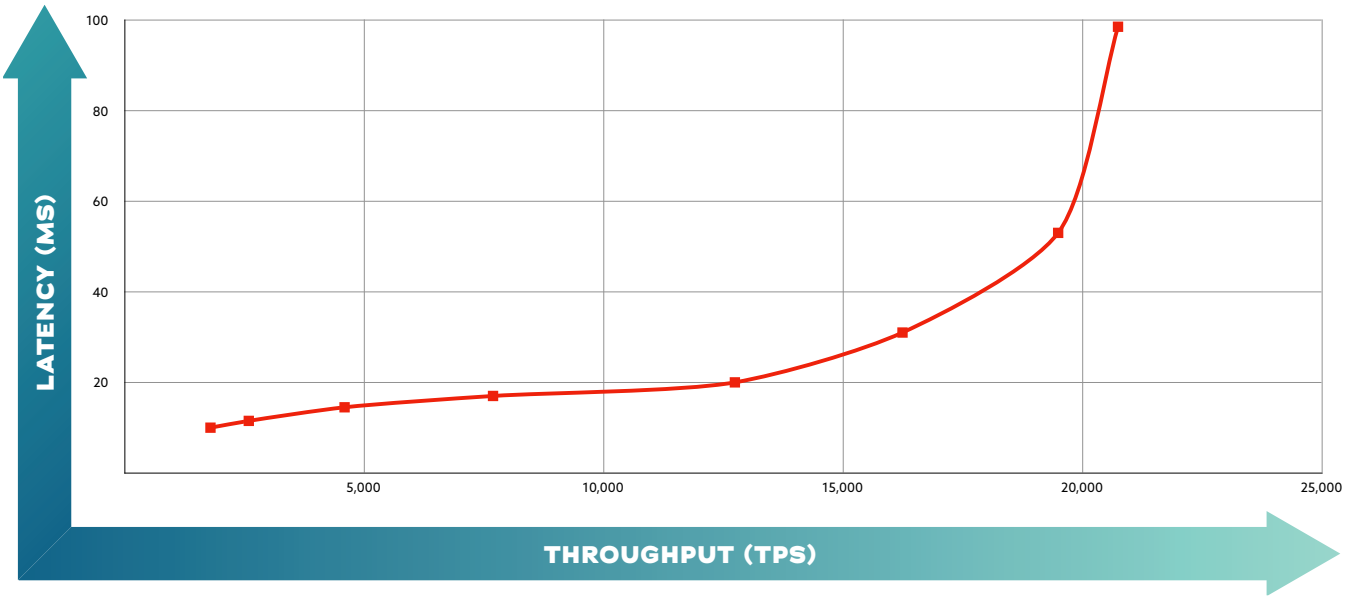
MariaDB Xpand is used by large-scale applications with millions of users by global companies such as Samsung and Massive Media. Xpand excels at read and write scaling data applications with absolute availability in the case of multiple zone failures. Additionally, Xpand offers columnar indexes to enable ad hoc, operational and personal analytics on transactional data. This capability removes the need to copy data to an analytical database for intraday and real-time analytics. Xpand is compatible with MariaDB and MySQL and is available on premises or via SkySQL, MariaDB's database-as-a-service offering.

CockroachDB claims scalability, availability, and disaster recovery. It is available as an on-premises installation or via CockroachDB Cloud, a fully hosted and managed service offering. CockroachDB includes partial compatibility with PostgreSQL. It does not currently support columnar indexes or state support for operational analytics.

METHODOLOGY

Vendors often publish benchmark results that claim bragging rights for the highest transaction value or the lowest latency. These results are often sparse and lack necessary details to give a sense of where those numbers relate to overall system performance.

Our benchmarking methods, on the other hand, are designed to convey the full picture of system (hardware and software) capacity under test. We achieve this by running the test workload under increasing concurrency while capturing throughput and response time (latency) and plotting them on a chart. The result is a chart like the one below.



We typically start with 8 concurrent sessions and each new run doubles the concurrency. In this chart, we use up to 1024 concurrent sessions, but depending on the workload type, we can go higher or lower.

In this particular chart, we see that as we increase the concurrency, the latency also slightly increases. This is normal behavior since we expect as more and more resources are used, there will be contention in the system which will translate into increased wait time. The latency gradually increases up to 128 concurrent sessions, however, above 128 and 256 we see an inflection point of a significant increase in wait time. This tells us that we are reaching the useful capacity of this particular system and any further increase in concurrency will result in higher and higher latency. Although at 1024 sessions we still get marginally more throughput, the very high latency makes this impractical for a real workload.

By looking at this graph, we can conclude that the useful capacity of this particular system will be somewhere between 64 and 128 concurrent sessions because latency is still low and there seems to be enough system capacity to take on occasional workload spikes.

The other conclusion we can draw from this graph is that if we do have the need to support 256, 1024 or higher concurrency rates, then we have to upgrade the system to a more powerful one. In the case of distributed SQL systems, such as Xpand, you can just add additional nodes to gain more capacity and support for higher throughput at lower latency levels.

RESULTS

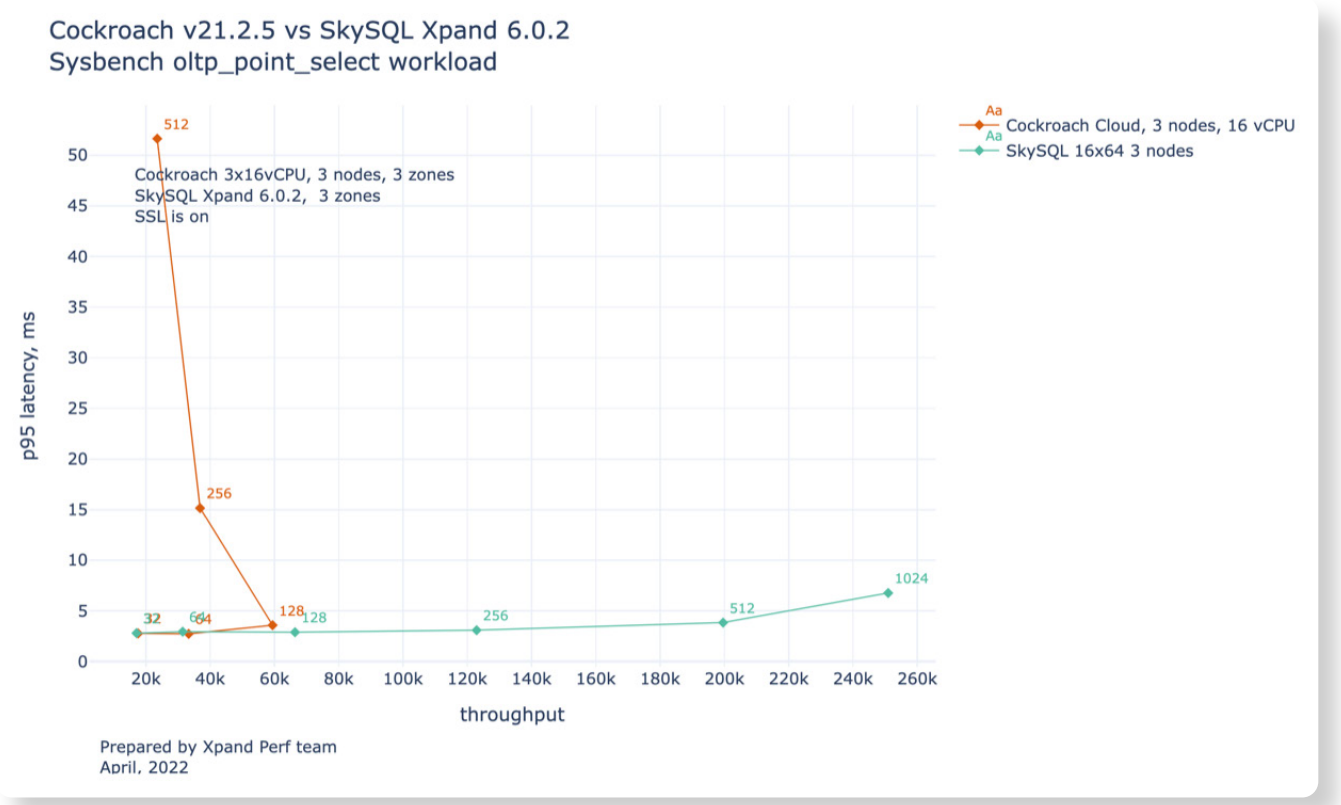
In each test, Xpand on SkySQL outperformed and outscaled CockroachDB on Cockroach Cloud with dedicated instances. This performance advantage includes both latency and throughput measures. The Sysbench system performance benchmark was used for testing. These are the findings:

OLTP Point Select

The Sysbench OLTP point-select test is a read scale test. It executes queries similar in form to "SELECT c FROM x WHERE id=?"

In this test both Xpand and Cockroach did well at the low end of 32 threads. Both databases had roughly equivalent latency and throughput though CockroachDB had a minor advantage. However, as the number of threads and operations increased, Cockroach quickly started to experience heavy latency. Even at 1024 threads, Xpand maintained p95 latency of 6.77ms and an average latency of 4.08ms.

P95 LATENCY, THROUGHPUT AND THREADS



Raw Data

Cockroach

concurrency	throughput	avg_latency	95th_latency	errors
32	17577.13	1.820	2.782	0
64	33315.22	1.920	2.739	0
128	59427.49	2.153	3.593	0
256	36863.53	6.943	15.168	0
512	23531.92	21.753	51.624	0

Xpand

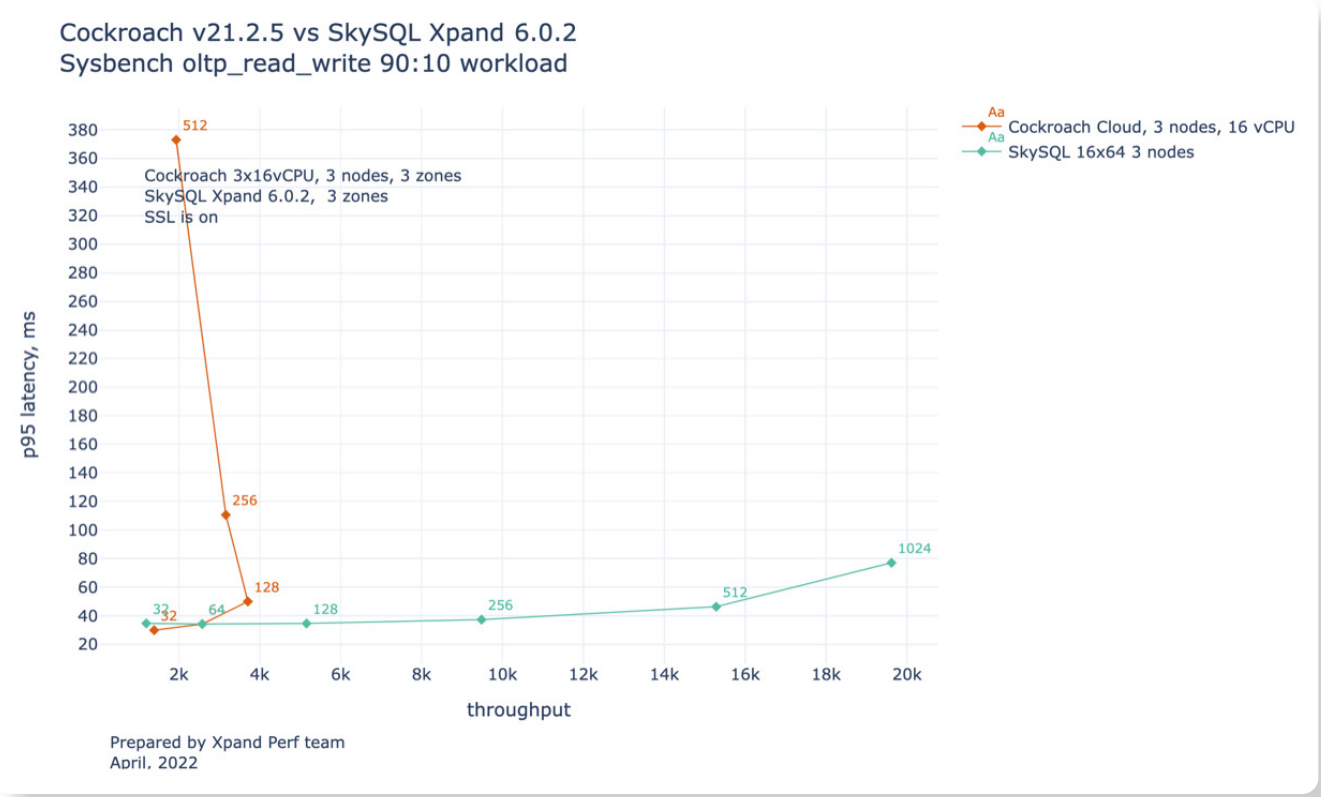
concurrency	throughput	avg_latency	95th_latency	errors
32	17083.65	1.873	2.812	0
64	31453.66	2.034	2.953	0
128	66406.55	1.927	2.900	0
256	122908.35	2.082	3.099	0
512	199609.09	2.564	3.850	0
1024	250977.28	4.077	6.767	0

OLTP Read/Write 90:10

The Sysbench OLTP read/write test is a mix of 90% reads and 10% writes. The test simulates what will happen when both read and write scale are required, but with a "read-mostly" scenario.

As with the point-select test, both Xpand and CockroachDB were close in terms of performance at the low end. Cockroach had a minor advantage in terms of both throughput and latency. The errors are a side effect of Sysbench and should be ignored. As the number of threads and thus volume of data increased, Cockroach dramatically increased in latency and throughput decreased. Cockroach reached p95 latency of 373.04ms at 512 threads and throughput dramatically reduced. Xpand reached a 77.03ms p95 latency at 1024 threads with almost 10x the throughput of Cockroach at 512 threads.

P95 LATENCY, THROUGHPUT AND THREADS



Raw Data

Cockroach

concurrency	throughput	avg_latency	95th_latency	errors
32	1391.07	23.001	29.964	7
64	2578.39	24.818	34.230	2
128	3706.76	34.527	50.036	11
256	3162.32	80.942	110.582	0
512	1933.97	264.677	373.040	0

Xpand

concurrency	throughput	avg_latency	95th_latency	errors
32	1194.98	26.776	34.732	0
64	2577.94	24.823	34.257	0
128	5156.31	24.820	34.684	0
256	9483.89	26.983	37.340	0
512	15283.68	33.470	46.380	0
1024	19616.68	52.161	77.033	0

INTERPRETATIONS

It appears that in a standard Xpand on SkySQL configuration and with a default CockroachDB on Cockroach Cloud configuration, Xpand greatly outscales and outperforms CockroachDB. It seems that the primary reason is that CockroachDB is CPU bound because the number of threads is limited to three times the number of vCPUs. We were unable to achieve results similar to what is claimed on their performance page. It should be noted that those results were not stated to be achieved with Cockroach's database-as-a-service. With the exception of changes required to ensure the same level of consistency for an equal comparison, we left both products in default configurations.

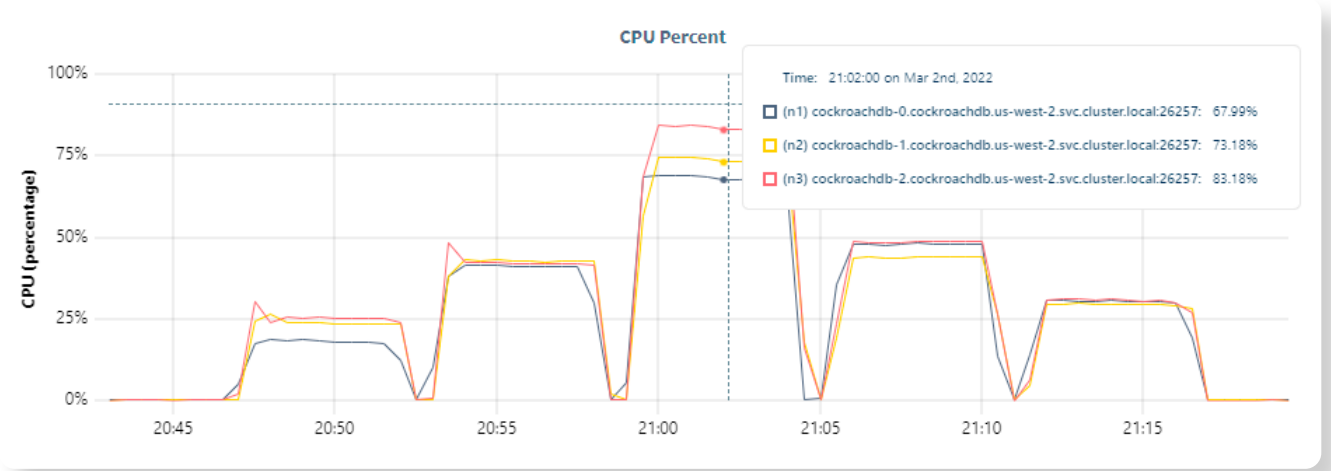
CockroachDB CPU Statistics

We investigated CockroachDB's performance bottleneck with the following script:

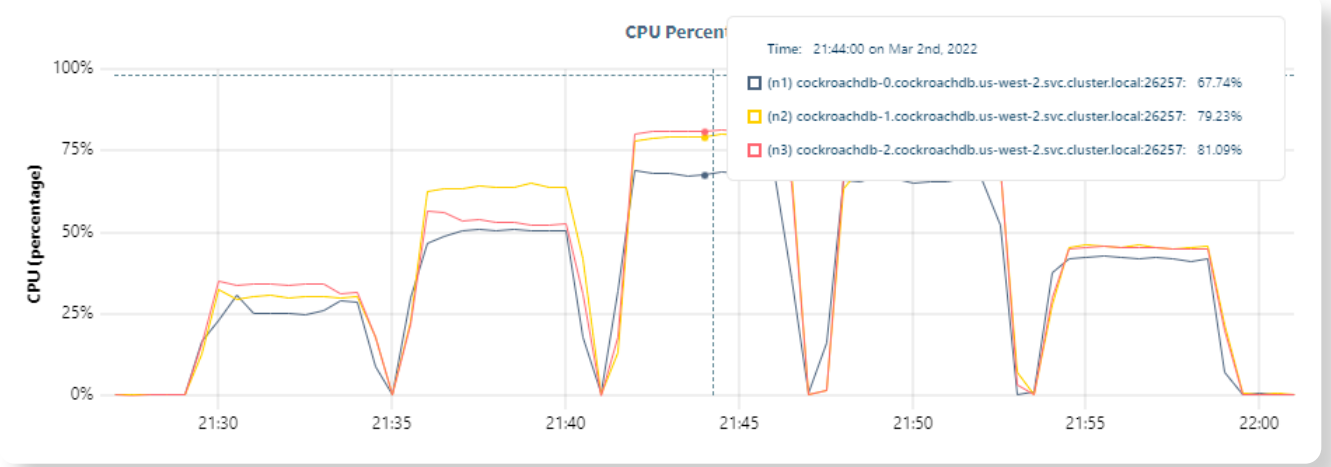
```
threads=( 32 64 128 256 512 )
for t in "${threads[@]}"
do
sysbench oltp_read_write --db-driver=pgsql --point-selects=9 --range-selects=false
--index-updates=0 --non-index-updates=1 --delete-inserts=0 --rand-type=uniform --report-
interval=10 --tables=10 --table-size=1000000 --time=300  --histogram --pgsql-host=${HOST}
--pgsql-user=sysbench --pgsql-password=sysbench2022 --pgsql-port=26257  --pgsql-db=sysbench
--threads=${t} --rand-seed=1234567 --pgsql-sslmode=verify-ca run > sysbench.${t}.out

    sleep 60
done
```

OLTP POINT SELECT CPU UTILIZATION FOR COCKROACHDB



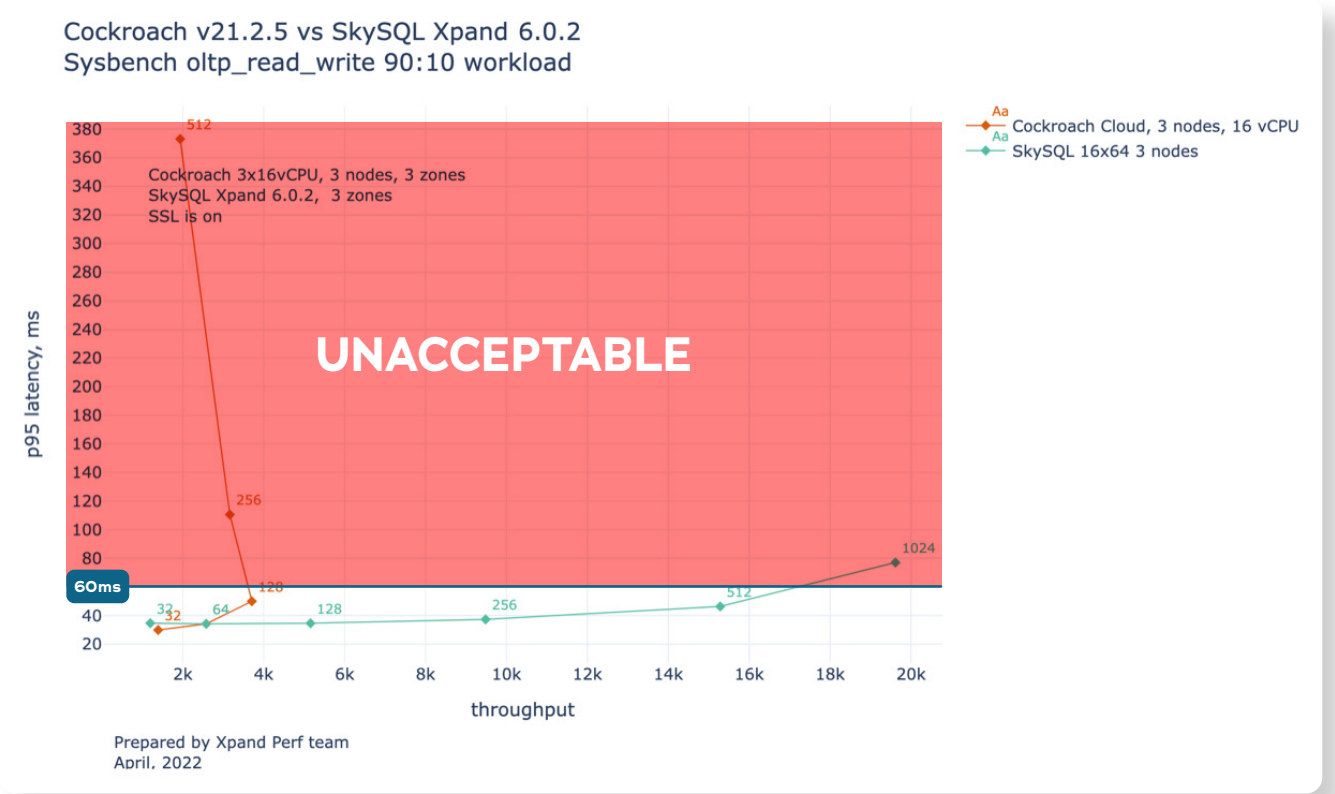
OLTP READ/WRITE CPU UTILIZATION FOR COCKROACHDB



PRICE/PERFORMANCE ANALYSIS

Xpand is significantly lower cost to operate than CockroachDB when load is considered. The monthly cost of operating a three-node cluster of CockroachDB is about $8,343.90. A similar cluster of Xpand on SkySQL with MaxScale (database proxy) is $9,673.16. However, this raw price does not tell the whole story. When performance differences are considered Xpand is substantially less expensive than CockroachDB.

While Sysbench is not fully representative of a typical real-world application, the cost per operation can be used as a proxy to understand the cost difference between two databases. In order to perform such a comparison, a performance goal must be established. For our purpose, let's assume that p95 latency above 60ms is unacceptable to our users.



Given the raw number of operations that can be executed with this performance, the throughput and monthly cost, we can derive a cost per operation and the difference between the two services.

Configuration	CockroachDB	Xpand
TOTAL COST	8343.9	9673.157
MAX OPERATIONS <60MS	3706.76	15283.68
OPERATIONS PER DAY	320264064	1320509952
OPERATIONS PER 30 DAYS	9607921920	39615298560
COST PER OPERATION	0.0000008684396136	0.0000002441773091
COST PER OPERATION DIFFERENCE	**355.66%**	

Given this analysis, an operation costs roughly 355.66% more on CockroachDB running on their database-as-a-service compared to Xpand on MariaDB's database-as-a-service platform, SkySQL.

Throughput and operations per month are only one aspect of price/performance. Generally, peak load processing governs cluster sizing. Meaning if an application required handling 15000 operations per second with a latency under 60ms, then a three-node Xpand cluster is sufficient. Based on these findings it can be assumed that CockroachDB would require a much larger cluster and raw cost would be higher.

CAPABILITIES CONSIDERATION

While this paper is not about different capabilities of the systems under test, there are capabilities inherent in the default SkySQL Xpand HA cluster configuration which may not be available in CockroachDB. Specifically, the capabilities provided by MaxScale. When a node fails during in-flight transactions, MaxScale automatically retries the transactions on the replacement node. If this capability is available in CockroachDB it is not documented (https://www.cockroachlabs.com/docs/stable/multi-active-availability.html). Secondly, MaxScale is configured in an HA configuration that allows the proxy to fail without affecting overall operations. While this paper is focused on price and performance considerations based on Sysbench results, these capabilities may be important when evaluating the overall cost versus capabilities of the system.

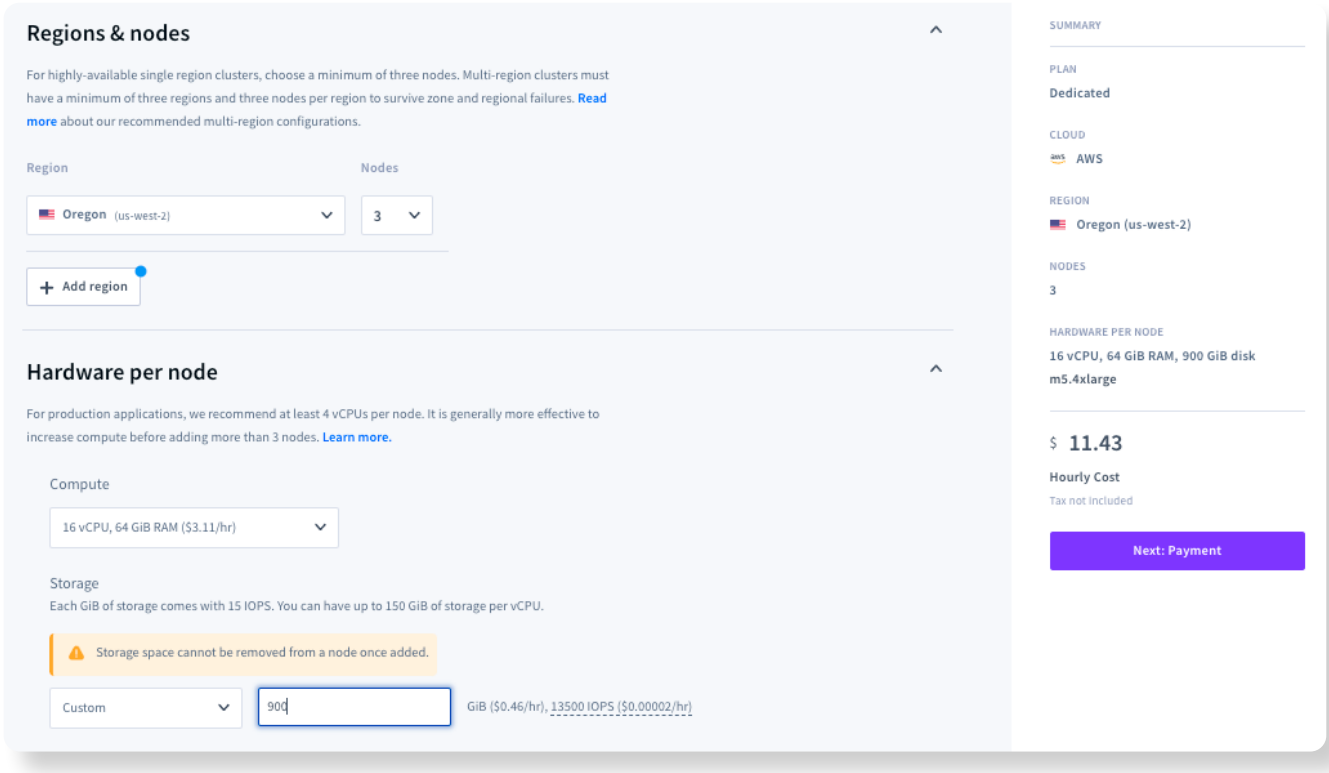
INFRASTRUCTURE CONFIGURATION

For this benchmark we used as close to the default configuration for each cloud as possible. Modifications were required in order to ensure an apples-to-apples configuration.

- **Driver**
  - 32vcpu
  - 128GB
  - sysbench 1.1.0-b73ae9a
  - CentOS Linux 7 (Core)
- **For both Xpand and CockroachDB**
  - AWS us-west-2
  - Three 16x64 instances
  - 13500 IOPS
  - RAM 64GiB
  - EBS storage 900GiB
- **For CockroachDB**
  - Version 21.2.5
  - Configured to three nodes instead of the default four
- **For Xpand**
  - Version 6.0.2
  - Configured to three replicas instead of the default two
  - Two instances of MaxScale database proxy

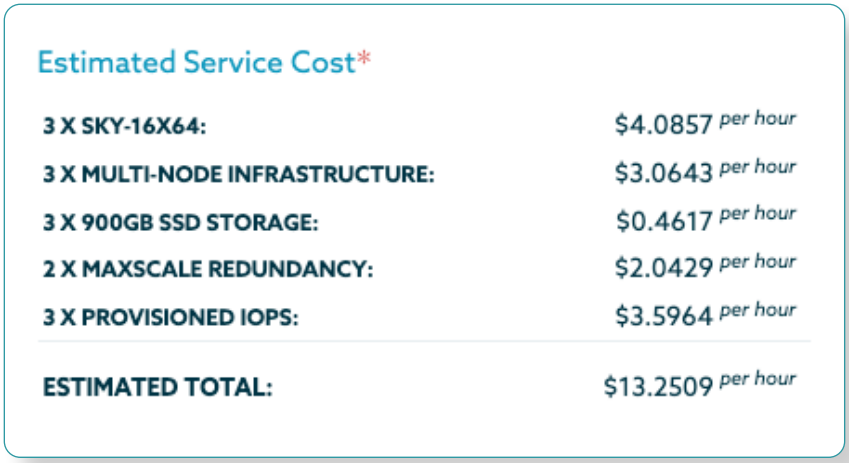
CockroachDB Setup and Price Information

For CockroachDB, we set up 3 nodes in us-west-2 (Oregon). By default CockroachDB uses 3 replicas, meaning all data exists on every node.



SkySQL/Xpand Setup and Price Information

For SkySQL we used a 3 zone configuration (default in the UI) and two MaxScale instances (due to limited CPU capacity with 1 MaxScale). We also used us-west-2. By default, SkySQL would have two replicas. In the interest of fairness we have configured it to the CockroachDB default (see later section).



CODE TO REPRODUCE THE RESULTS

Sysbench OLTP Point Select

```
threads=( 8 16 32 64 128 256 512 1024  )
for t in "${threads[@]}"
do
sysbench ${WORKLOAD} --create_secondary=false --auto_inc=false --db-driver=mysql --mysql-
host=$HOST --mysql-user=sysbench --mysql-password=sysbench2022 --mysql-port=5001 --mysql-
db=sysbench --report-interval=10 --table-size=1000000 --tables=10 --threads=${t} --auto-inc=off
--mysql-ssl-ca=aws_skysql_chain.pem run > sysbench.${t}.out

    sleep 60
done
```

Sysbench OLTP Read/Write

```
threads=( 8 16 32 64 128 256 512 1024  )
for t in "${threads[@]}"
do
sysbench oltp_read_write --point-selects=9 --range-selects=false --index-updates=0 --non-index-
updates=1 --delete-inserts=0 --rand-type=uniform --report-interval=10 --tables=10 --table-
size=1000000 --time=300 --histogram  --mysql-db=sysbench --db-driver=mysql --mysql-host=${HOST}
--mysql-user=sysbench --mysql-password=sysbench2022 --mysql-port=5001 --threads=${t} --rand-
seed=1234567 --mysql-ssl-ca=aws_skysql_chain.pem run > sysbench.${t}.out

    sleep 60
done
```

CHANGES TO SKYSQL XPAND CONFIGURATION

By default Xpand uses two replicas instead of the three that CockroachDB uses. In the interest of providing a direct comparison we altered this configuration to match CockroachDB's default configuration. The script is as follows:

```
use sysbench;
 ALTER TABLE sbtest1 slices=45;
 ALTER TABLE sbtest2 slices=45;
 ALTER TABLE sbtest3 slices=45;
 ALTER TABLE sbtest4 slices=45;
 ALTER TABLE sbtest5 slices=45;
 ALTER TABLE sbtest6 slices=45;
 ALTER TABLE sbtest7 slices=45;
 ALTER TABLE sbtest8 slices=45;
 ALTER TABLE sbtest9 slices=45;
 ALTER TABLE sbtest10 slices=45;

ALTER TABLE sbtest1 replicas=3;
ALTER TABLE sbtest2 replicas=3;
ALTER TABLE sbtest3 replicas=3;
ALTER TABLE sbtest4 replicas=3;
ALTER TABLE sbtest5 replicas=3;
ALTER TABLE sbtest6 replicas=3;
ALTER TABLE sbtest7 replicas=3;
ALTER TABLE sbtest8 replicas=3;
ALTER TABLE sbtest9 replicas=3;
ALTER TABLE sbtest10 replicas=3;
```

Note that this modification likely improves the read performance of Xpand over default, but may reduce the write performance. It also makes the cluster more highly available since there is an extra redundant node in the event of an outage.

HOW TO TUNE COCKROACHDB FOR BETTER RESULTS

The methodology used for our benchmark relied on as close to default configuration as possible in order to provide an apples-to-apples comparison for development teams and the vendors that provide adequate and reasonable defaults for their services. With that said, applying the following tuning parameters to CockroachDB will improve performance. While we did not perform a detailed test, they appear to improve scale and performance by a factor of 2. Please consider setting these when reproducing the results.

```
ALTER RANGE default CONFIGURE ZONE USING range_min_bytes = 0,range_max_bytes = 67108864,gc.
ttlseconds = 90000;

SET CLUSTER SETTING kv.range_merge.queue_enabled = false;    -- default is true
SET CLUSTER SETTING kv.range_split.by_load_enabled = true;   -- default is true
SET CLUSTER SETTING kv.range_split.load_qps_threshold = 200; -- default 2500
```

LEARN MORE

Please watch this on-demand webcast on MariaDB Xpand vs. CockroachDB for a closer look at the architecture of both databases, their respective feature sets and benchmark performance details.

Try Xpand

We invite you to start your trial of Xpand today, or for the easiest and fastest way to get started use Xpand in the cloud. If you need help running or interpreting the benchmark join us on the MariaDB Community Slack in the #benchmark channel.