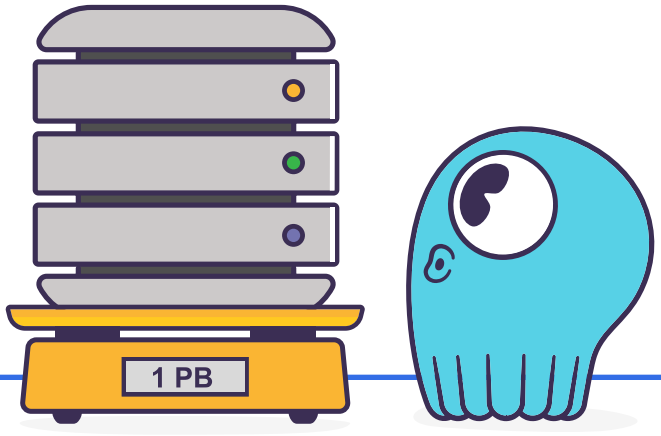
SCYLLA

SCYLLADB WHITE PAPER

What We Learned Benchmarking Petabyte-Scale Workloads with ScyllaDB



With the rise of real-time applications reading and writing petabytes of data daily, it's not surprising that database speed at scale is facing increased scrutiny. Even if you're not planning for growth, a surge could occur when you're least expecting it. Yet, scaling latency-sensitive data-intensive applications is not trivial. Teams often learn all too late that the database they originally selected is not up to the task.

Benchmarks performed at petabyte scale can help you understand how a particular database handles the extremely large workloads that your company expects – or at least hopes – to encounter. However, such benchmarks can be challenging to design and execute.

At ScyllaDB, we decided to perform a foundational petabyte benchmark of our high-performance, low-latency database in early 2022 for a number of reasons:

- To help the increasing number of ScyllaDB users and prospects with petabyte-scale use cases understand if our database is aligned with their requirements for speed at scale.
- To establish a baseline against which to measure the performance improvements achieved with our upcoming ScyllaDB 5.x releases and new AWS EC2 instances.
- To quantify the latency impact of ScyllaDB's unique workload prioritization capability, which allows admins to allocate "shares" of the available hardware infrastructure to different workloads.

This paper outlines the configuration process, results (with and without workload prioritization), and lessons learned that might benefit others planning their own petabyte-scale benchmark.

EXECUTIVE SUMMARY OF THE RESULTS

ScyllaDB stored a 1 PB data set using only 20 large machines running two parallel mixed workloads at 7.5 million operations per second and single-digit millisecond latency.

The results reflect a storage density of 50 TB/server, which is unparalleled in the industry. The amount of servers contributes to a low total cost of ownership (TCO) as well as operational simplicity.

ScyllaDB demonstrated workload prioritization, one of its unique features that allows users to assign priorities per workload. Workload prioritization enables cluster consolidation and offers an additional level of savings. In this setup, a smaller – but highly important – workload of 1TB was hosted on the same mega 1 PB deployment. Traditionally, such a workload can get starved, since it is 1000x smaller than the large workload. Moreover, the large workload was dominating the cluster with 7M TPS while the smaller one had "only" 280K TPS. Nevertheless, when assigned a priority, the smaller workload improved its latency by 2x to 1-2 ms in P99.

To summarize, ScyllaDB allows you to scale to any workload and consolidate multiple workloads into a single moderately-sized cluster.

CONFIGURATION

TWO CONCURRENT WORKLOADS: OLTP AND OLAP

We designed a model to represent a petabyte-scale application where event data needs to be stored rapidly, then accessed by multiple locations – often simultaneously – for both online transaction processing and analytics. We chose two concurrent workloads:

- **Application data (OLTP):** Data for transaction processing – for example, real-time ad bidding or real-time location data in delivery/Uber/IoT applications. This workload requires low latency to process the frequent, incoming requests. We approximated this would involve several terabytes.

- **User data (OLAP):** A larger data set of user-specific data used primarily for analytics. It is read-heavy, and updated regularly. This was estimated to require a petabyte.

To apply this model, we constructed a 20 node ScyllaDB cluster (details below) and loaded it with 1 PB (uncompressed) of User data and 1 TB of Application data. The User workload was ~5 million TPS, and we measured two variants of it: one read-only, and another with 80% reads and 20% writes. Since this workload simulated online analytics, high throughput was critical. At the same time, we ran a smaller 200,000 TPS Application workload with 50% reads and 50% writes. Since this workload represented online transaction processing, low latency was prioritized over high throughput.

Both workloads were constructed as a key-value data set with keys randomly selected in gaussian distribution[1]. The User keyspace had 500 billion keys, while the Application keyspace had 6 billion[2]. We used LZ4 compression on the server. However, cassandra-stress, our client tool, only generated fully randomized strings which didn't compress at all. So, we decided to reduce the payload in a ratio of 3:1 to reflect a real life case. In addition, we used replication of two (RF=2).

1. Note that ScyllaDB supports wide column data sets as well as key-value ones.

2. For additional details on the workload configuration, see Operating at Monstrous New Scales: Benchmarking Petabyte Workloads on ScyllaDB.

CLUSTER SPECS

To build the **ScyllaDB cluster**, we provisioned 20 x i3en.metal AWS instances. Each instance had:

- 96 vCPUs
- 768 GiB RAM
- 60 TB NVMe disk space
- 100 Gbps network bandwidth

For the **load generators,** we used 50 x c5n.9xlarge AWS instances. Each instance had:

- 36 vCPUs
- 96 GiB RAM
- 50 Gbps network bandwidth

SOFTWARE

The ScyllaDB nodes were running ScyllaDB Enterprise version 2021.1.6 (based on ScyllaDB Open Source 4.3). To generate the various load scenarios, we used the industry standard cassandra-stress tool over the ScyllaDB shard-aware Java driver. Using a shard-aware driver is essential for getting the best performance from the ScyllaDB cluster. We also used the ScyllaDB monitoring stack for gathering and visualizing the metrics.

RESULTS

DATA INGESTION: 7.5 M INSERTS PER SECOND

We could insert data at a rate of 7.5M inserts per second using the 50 concurrent load generators. With this high throughput, we saw a 4 ms P99 write latency. That allowed us to load the 1 PB cluster in roughly 20 hours. The CPU load utilization during ingestion was approximately 90%, on average. The total throughput during ingestion was approximately 120 gigabytes per second.

LATENCY: SINGLE-DIGIT MILLISECOND P99 LATENCY AT 7M TPS

The primary goal of this benchmark was to see how much we could load the ScyllaDB petabyte cluster and still provide single-digit millisecond 99 percentile latency. **The ScyllaDB cluster achieved single-digit millisecond P99 latency at 7M TPS throughput: under 7 ms for writes and just over 2 ms for reads.**

As you can see from the lower two lines, the P50 read and write latency is well under 1 millisecond. The upper two lines depict P99 latencies. As expected, the P99 latency increased as throughput increased. We were pleased to see that it remained safely in the single-digit millisecond range up to 7M TPS.

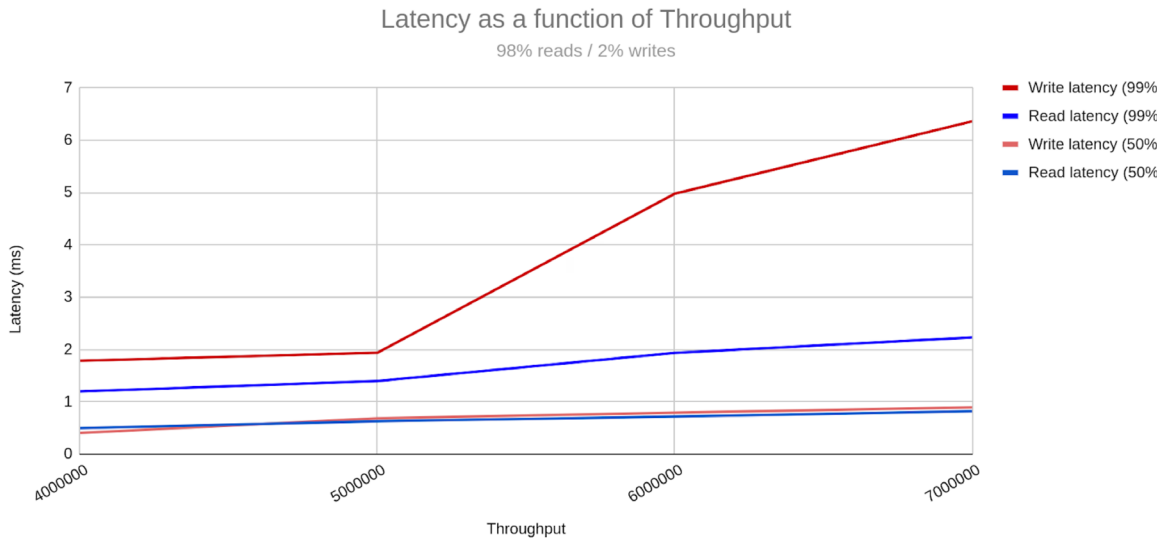


Figure 1: The ScyllaDB cluster achieved single-digit millisecond P99 latency with 7M TPS

Let's drill down into the results for different workloads.[3]

Concurrent Workloads: R/W + Read-Only

Workload (in TPS)	Application: 280K R/W	User: 7M read-only
Write latency (in ms)	0.821 P50 2.232 P99	
Read latency (in ms)	1.433 P50 6.832 P99	0.885 P50 6.350 P99

Concurrent Workloads: R/W + Read-Only

Workload (in TPS)	Application: 200K R/W	User: 5M read-only
Write latency (in ms)	0.632 P50 1.398 P99	
Read latency (in ms)	1.046 P50 2.279 P99	0.680 P50 1.932 P99

Concurrent Workloads: R/W + 80/20

Workload (in TPS)	Application: 200K R/W	User: 5M 80/20 R/W
Write latency (in ms)	0.682 P50 2.454 P99	0.326 P50 1.252 P99
Read latency (in ms)	1.195 P50 4.555 P99	0.744 P50 3.709 P99

This last scenario used a more realistic scenario (80% reads, 20% writes) for the Application workload. Interestingly, this caused a significant increase in P99 latency for both writes and reads. That might cause an SLA violation for an application that requires latencies of 2 or 3 milliseconds, for example.

Luckily, ScyllaDB is equipped with a built-in mechanism that allows users to specify a priority per workload. The capability, called workload prioritization, is trivial to configure. We wanted to see how the application workload would change once we applied a higher priority to it.

IMPACT OF WORKLOAD PRIORITIZATION

We then applied ScyllaDB's workload prioritization capability and measured its impact on the latency-sensitive Application workload. Workload prioritization can dynamically balance and equalize the system resources and divide them between different workloads that share the same hardware infrastructure.

We did not change the 1000 shares allocated to the Application workload, but we did reduce the number of shares granted to the User workload from 1,000 to 500. This cut the Application workload's P99 latencies significantly.

Impact of Workload Prioritization

Workload (in TPS)	Application: 200K R/W		User: 5M 80/20 R/W	
	before: 1000 shares	after: 1000 shares	before: 1000 shares	after: 500 shares
Read latency (in ms)	1.195 P50 4.555 P99	0.354 P50 1.184 P99	0.326 P50 1.252 P99	0.440 P50 3.244 P99
Read latency (in ms)	1.195 P50 4.555 P99	0.855 P50 3.731 P99	0.744 P50 3.709 P99	1.043 P50 6.455 P99

3. For more details on why we selected these workloads and how we configured them, see Operating at Monstrous New Scales: Benchmarking Petabyte Workloads on ScyllaDB.

LESSONS LEARNED

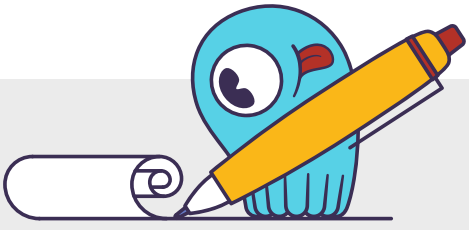
If you're thinking about performing your own petabyte-scale benchmark, here are some lessons learned that you might want to consider:

- **Provisioning:** It took a few days to find an availability zone in AWS that had sufficient instance types for a petabyte-scale benchmark. If you plan to deploy such a large cluster, make sure to provision your resources well ahead.
- **Hardware tuning – interrupt handling:** Our default assignment of cores to I/O queue handling wasn't optimized for this scenario. Interrupt handling CPUs had to be manually assigned to maximize throughput. A fix for this is being merged into out-of-the-box machine images.
- **Hardware tuning – CPU power governor:** We needed to set the CPU power governor on each node to "performance" in order to maximize the performance of the system.
- **cassandra-stress:** cassandra-stress was not designed for this scale (e.g., the default population distribution is too small). Be prepared to use non-default settings to create a petabyte of data.
- **Potential optimizations:** We suspect that using BYPASS CACHE (a ScyllaDB CQL extension that asks the server to just disregard the cache and execute the query directly from disk) could improve performance by 70%. If the complete data set fits in a cache, we can see an improvement of up to a factor of four. In future benchmarks, we will measure the impact of using BYPASS_CACHE.
- **ScyllaDB configurations:** We recommend using the following non-default ScyllaDB configurations if running a petabyte-scale benchmark:
  - Node level
    - 4 irq-serving CPUs (rather than 2 by default) – for handling high network throughput
    - `mount -o discard` (now default in OSS head-of-line)
  - scylla.yaml:
    - `compaction_static_shares: 100` – optimized for append-mostly workload (*)
    - Head-of-line has improvements for compaction backlog controller
    - `compaction_enforce_min_threshold: true`
  - Schema:

```
compaction = {
  'class':
'IncrementalCompactionStrategy',
  'sstable_size_in_mb': 10000,
  'space_amplification_goal': 1.25
} AND compression = {'sstable_
compression': 'LZ4Compressor'};
```

ABOUT SCYLLADB

ScyllaDB is the database for data-intensive apps that require high performance and low latency. It enables teams to harness the ever-increasing computing power of modern infrastructures – eliminating barriers to scale as data grows. Unlike any other database, ScyllaDB is built with deep architectural advancements that enable exceptional end-user experiences at radically lower costs. Over 400 game-changing companies like Disney+ Hotstar, Expedia, FireEye, Discord, Crypto.com, Zillow, Starbucks, Comcast, and Samsung use ScyllaDB for their toughest database challenges. ScyllaDB is available as free open source software, a fully-supported enterprise product, and a fully managed service on multiple cloud providers. For more information: ScyllaDB.com



SCYLLA

United States Headquarters
2445 Faber Place, Suite 200
Palo Alto, CA 94303 U.S.A.
Email: info@scylladb.com

Israel Headquarters
11 Galgalei Haplada
Herzelia, Israel

Copyright © 2022 ScyllaDB Inc. All rights reserved. All trademarks or registered trademarks used herein are property of their respective owners.