SCYLLADB WHITE PAPER

SCYLLA

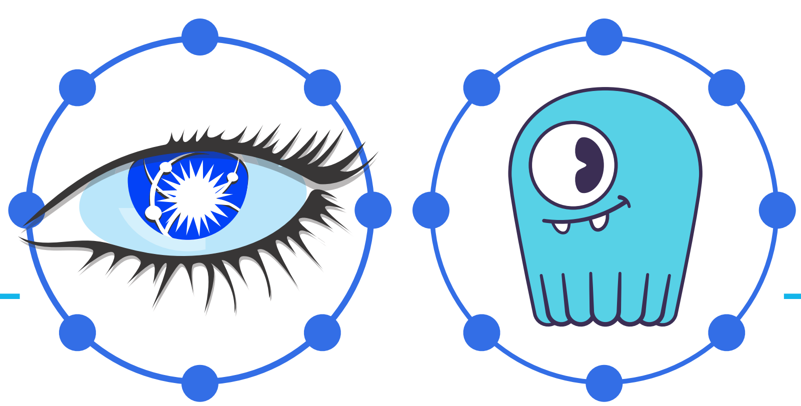
Apache Cassandra 4.0 Performance Benchmark

Comparing Cassandra 4.0, Cassandra 3.11 and Scylla Open Source 4.4

Piotr Grabowski
Software Engineer, ScyllaDB

Juliusz Stasiewicz
Software Engineer, ScyllaDB

Karol Baryla
Junior Software Engineer, ScyllaDB



CONTENTS

In July 2021, after nearly six years of work, the engineers behind Apache Cassandra incremented the database's major version from 3 to 4. In the rapidly evolving realm of big data, six years encompasses almost an entire technology cycle, with new Java virtual machines, new system kernels, new hardware, new libraries and even new algorithms. Progress in these areas presented the engineers behind Cassandra with unprecedented opportunities to achieve new levels of performance. Throughout this period, ScyllaDB also progressed significantly in the areas of high performance, resilience, and operational integrity, continuously improving the Scylla database engine with new features and optimizations.

In this paper, we compare the performance of the latest release of Scylla Open Source against Cassandra 3 and the newly released Cassandra 4. We present the latencies and throughputs measured for various workloads, as well as the speed of common administrative operations such expanding clusters and running major compactions.

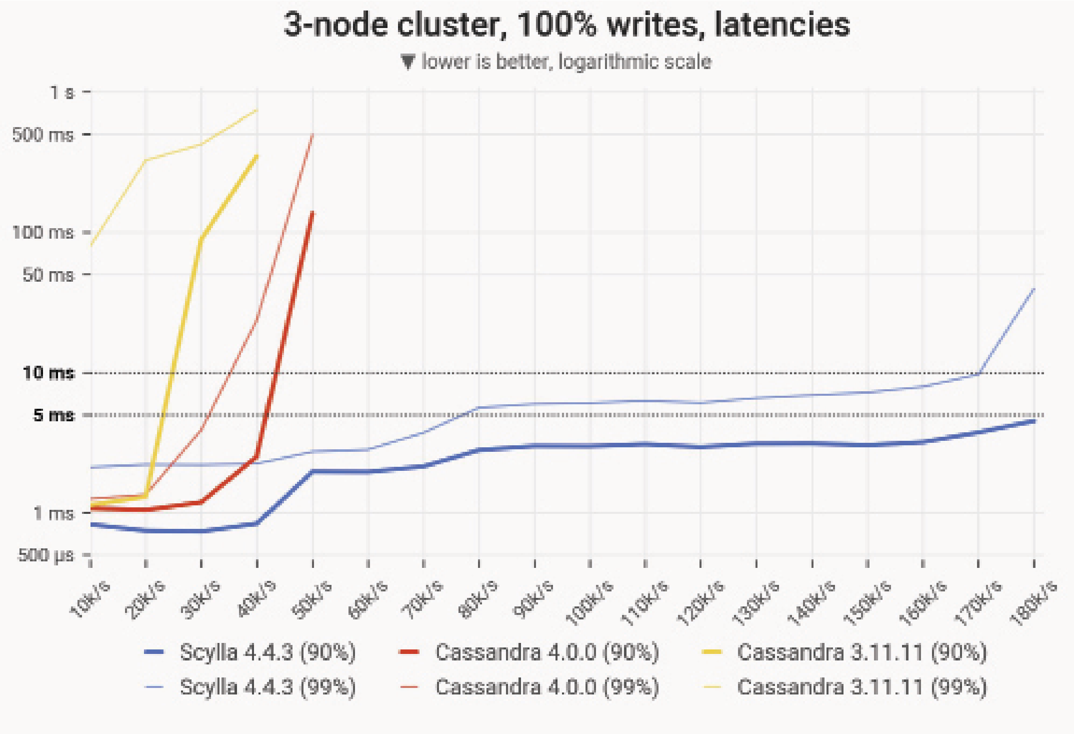
SUMMARY OF RESULTS

The detailed results and the fully optimized setup instructions are shared later in this report. We compared two deployment options in the AWS EC2 environment:

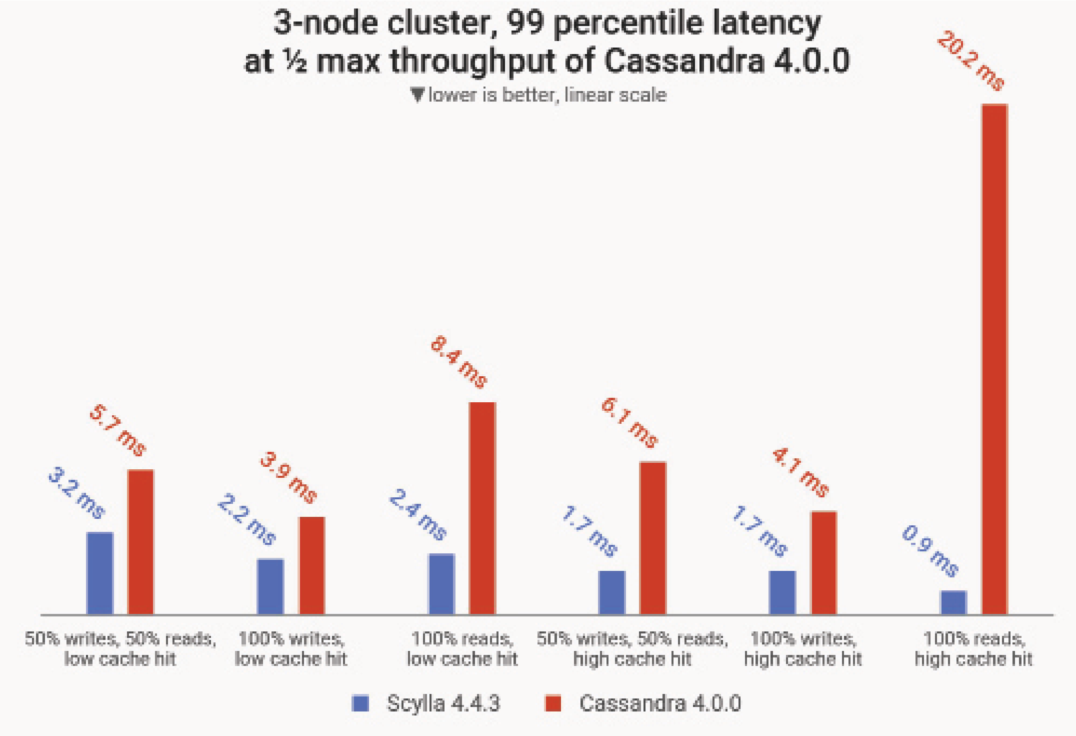
1. The first is an apples-to-apples comparison of 3-node clusters.
2. The second is a larger-scale setup where we used node sizes optimal for each database. Scylla can utilize very large nodes so we compared a setup of 4 i3.metal machines (288 vCPUs in total) vs. 40 (!) i3.4xlarge Cassandra machines (640 vCPUs in total - almost 2.5x the Scylla's resources).

Key findings:

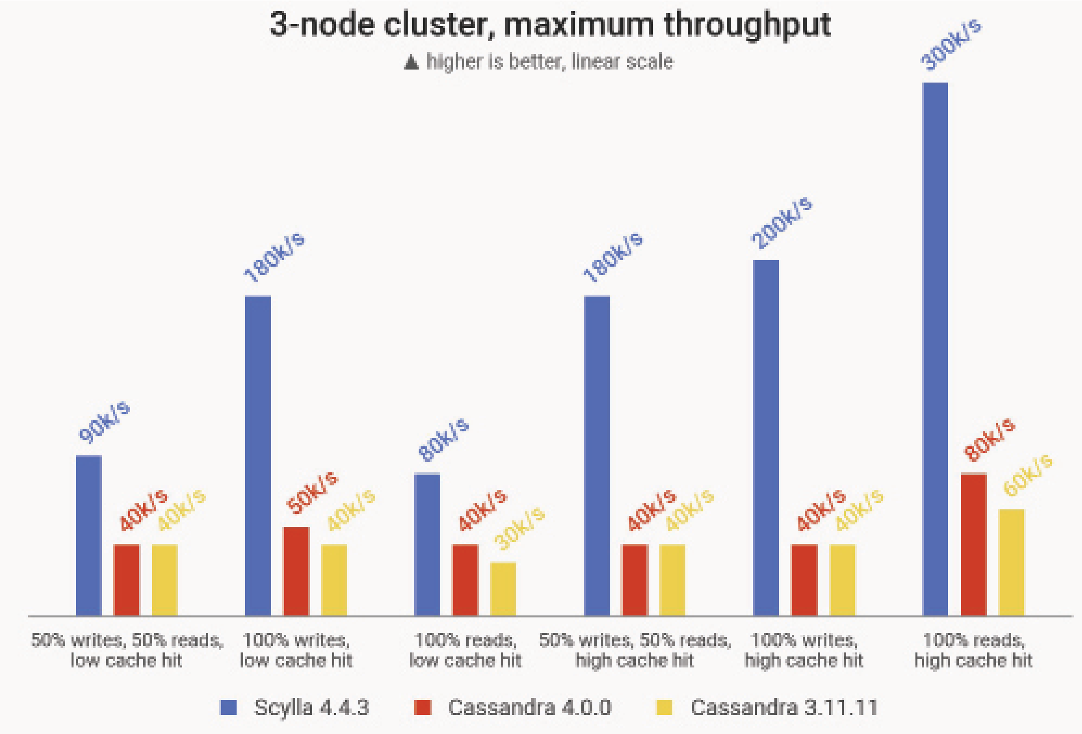
- Cassandra 4.0 has better P99 latency than Cassandra 3.11 by 100x
- Cassandra 4.0 speeds up admin operations by up to 34% compared to Cassandra 3.11
- Scylla has 2x-5x better throughput than Cassandra 4.0 on the same 3-node cluster
- Scylla has 3x-8x better throughput than Cassandra 4.0 on the same 3-node cluster while P99 < 10 ms
- Scylla adds a node 3x faster than Cassandra 4.0
- Scylla replaces a node 4x faster than Cassandra 4.0
- Scylla doubles a 3-node cluster capacity 2.5x faster than Cassandra 4.0
- A 40 TB cluster is 2.5x cheaper with Scylla while providing 42% more throughput under P99 latency of 10 ms
- Scylla adds 25% capacity to a 40 TB optimized cluster 11x faster than Cassandra 4.0.
- Scylla finishes compaction 32x faster than Cassandra 4.0
- Cassandra 4.0 can achieve a better latency with 40 i3.4xlarge nodes than 4 i3.metal Scylla nodes when the throughput is low and the cluster is being underutilized. Explanation follows.



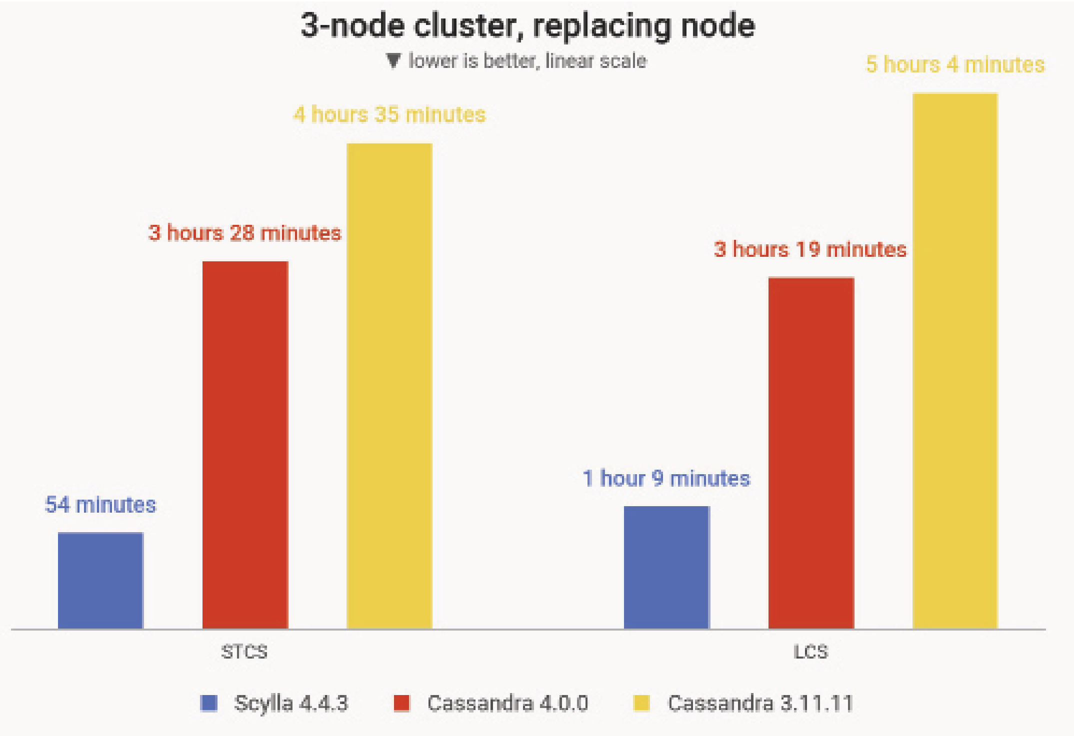
The 90- and 99-percentile latencies of UPDATE queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. ***Cassandra quickly became functionally nonoperational, serving requests with tail latencies that exceeded 1 second.***



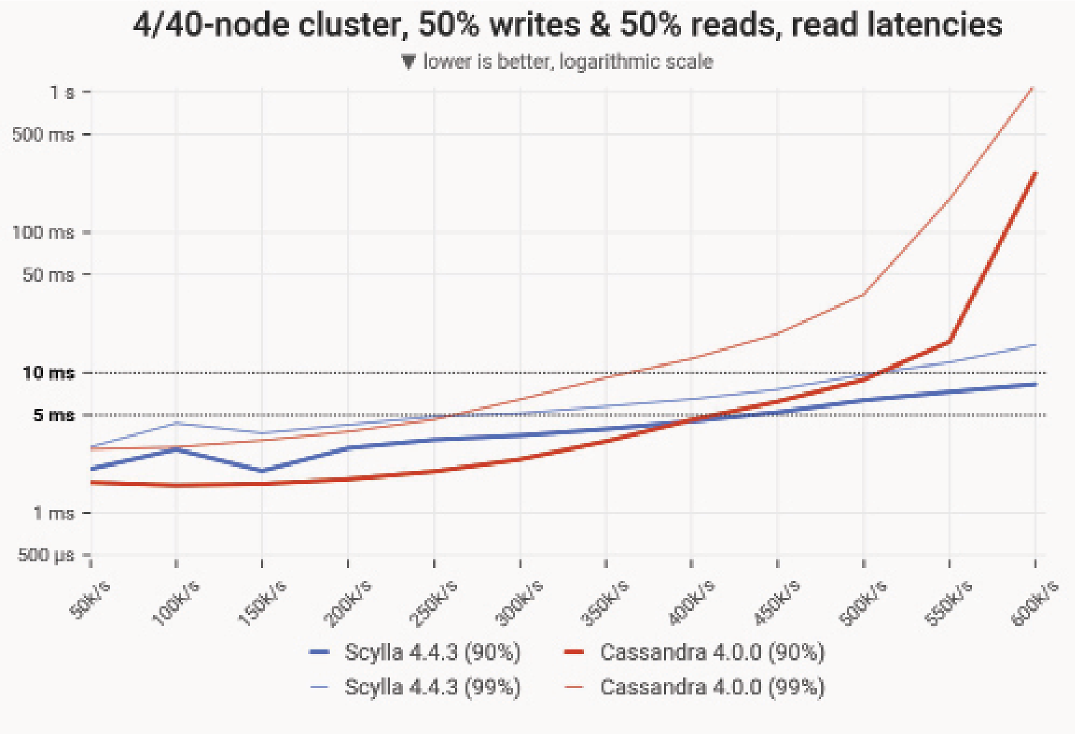
The 99-percentile (P99) latencies in different scenarios, as measured on 3 x i3.4xlarge machines (48 vCPUs in total) under load that puts Cassandra 4.0 halfway to saturation. ***Scylla excels at response times****: Cassandra 4.0 P99 latencies are anywhere between 80% to 2,200% greater than Scylla 4.4.*



The maximum throughput (measured in operations per second) achieved on 3 x i3.4xlarge machines (48 vCPUs). ***Scylla processed 2x - 5x more requests*** *than either of the Cassandra releases.*



The time taken by replacing a 1 TB node, measured under Size-Tiered Compaction Strategy (STCS) and Leveled Compaction Strategy (LCS). ***By default (STCS) Scylla is almost 4x faster than Cassandra 4.0.***



Latencies of `SELECT` *query, as measured on 40 TB cluster on uneven hardware — 4 nodes (288 vCPUs) for Scylla and 40 nodes (640 vCPUs) for Cassandra.*

LIMITATIONS OF OUR TESTING

It's important to note that this basic performance analysis does not cover all factors in deciding whether to stay put on Cassandra 3.x, upgrade to Cassandra 4.0, or to migrate to Scylla Open Source 4.4. Users may be wondering if the new features of Cassandra 4.0 are compelling enough, or how changes between implemented features compare between Cassandra and Scylla. To give a couple examples, you can read more about the difference in CDC implementations here, and how Scylla's Lightweight Transactions (LWT) differ from Cassandra's here. Apart from comparison of basic administrative tasks like adding one or more nodes, which is covered below, benchmarking implementation of specific features is beyond the scope of this paper.

Additionally, there are issues of risk aversion based on stability and maturity for any new software release — for example, the ZGC garbage collector we used currently employs Java 16, which is supported by Cassandra but not considered production-ready; newer JVMs are not yet officially supported by Cassandra.

CLUSTER OF THREE I3.4XLARGE NODES

3-NODE TEST SETUP

The purpose of this test was to **compare the performance of Scylla vs. both versions of Cassandra on the exact same hardware**. We wanted to use relatively typical, current generation servers running on AWS so that others could replicate these tests, and in order to reflect a real-world setup.

AWS Instance Type	Cassandra/ Scylla	Loaders
EC2 Instance type	i3.4xlarge	c5n.9xlarge
Cluster size	3	3
vCPUs (total)	16 (48)	36 (108)
RAM (total)	122 (366) GiB	96 (288) GiB
Storage (total)	2x 1.9TB NVMe in RAID0 (3.8 TB)	Not important for a loader (EBS-only)
Network	Up to 10 Gbps	50 Gbps

We set up our cluster on Amazon EC2, in a single Availability Zone within us-east-2. Database cluster servers were initialized with clean machine images (AMIs), running **CentOS 7.9 with Scylla Open Source 4.4** and **Ubuntu 20.04 with Cassandra 4.0** or **Cassandra 3.11** (which we'll refer to as "C*4" and "C*3", respectively).

Apart from the cluster, three loader machines were employed to run `cassandra-stress` in order to insert data and, later, to provide background load to mess with the administrative operations.

Once up and running, the databases were loaded by cassandra-stress with random data organized into the default schema at RF=3. The loading continued until the cluster's total disk usage reached approx. 3 TB (or 1 TB per node). The exact disk occupancy would, of course, depend on running compactions and the size of other related files (commitlogs, etc.). Based on the size of the payload, this translated to ~3.43 billion partitions. We then flushed the data and waited until the compactions finished, so we could start the actual benchmarking.

THROUGHPUT AND LATENCIES

The actual benchmarking is a series of simple invocations of `cassandra-stress` with CL=QUORUM. For 30 minutes we kept firing 10,000 requests per second and monitored the latencies. Then we increased the request rate by another 10,000 for another 30 min, and so on. (20,000 in case of larger throughputs). The procedure repeated until the DB was no longer capable of withstanding the traffic, i.e. until `cassandra-stress` could not achieve the desired throughput or until the 90-percentile latencies exceeded 1 second.

Note: This approach meant that throughput numbers are presented with 10k/s granularity (in some cases 20k/s).

We tested the databases with the following distributions of data:

1. "Real-life" (Gaussian) distribution, with sensible cache-hit ratios of 30-60%
2. Uniform distribution, with a close-to-zero cache hit ratio
3. "In-memory" distribution, expected to yield almost 100% cache hits

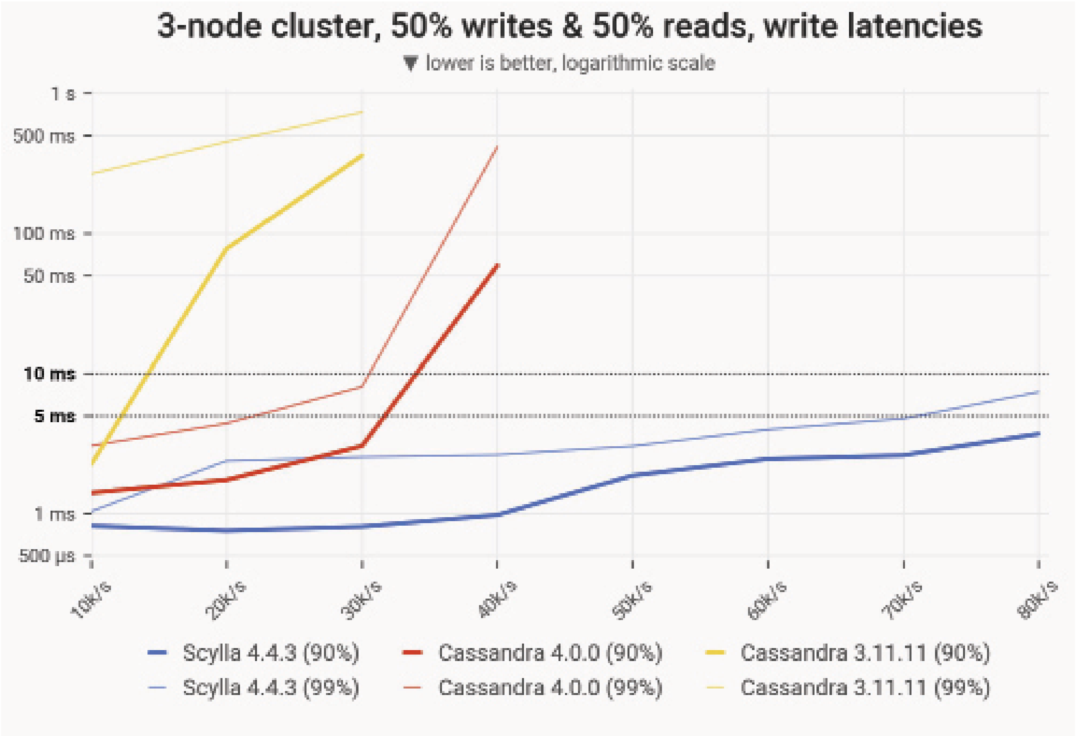
Within these scenarios we ran the following workloads:

1. 100% writes
2. 100% reads
3. 50% writes and 50% reads

"REAL-LIFE" (GAUSSIAN) DISTRIBUTION

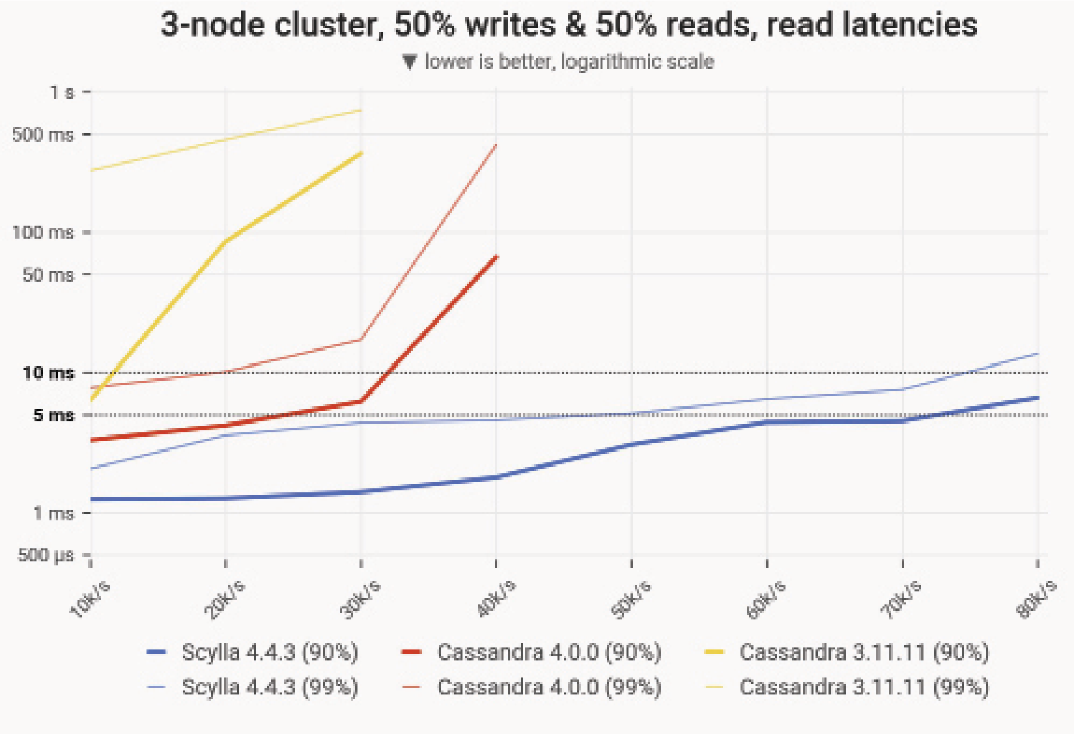
In this scenario we issued queries that touched partitions **randomly drawn from a narrow Gaussian distribution**. We made an Ansatz about the bell curve: We assumed that its six-sigma spans the RAM of the cluster (corrected for the replication factor). The purpose of this experiment was to model a realistic workload, with a substantial cache hit ratio but less than 100%, because most of our users observe the figures of 60-90%. We expected Cassandra to perform well in this scenario because its key cache is denser than Scylla's, i.e. it efficiently stores data in RAM, though it relies on SSTables stored in the OS page cache, which can be heavyweight to look up. By comparison, Scylla uses a row-based cache mechanism. This Gaussian distribution test should indicate which database uses the more efficient caching mechanism for reads.

Mixed Workload – 50% reads and 50% writes



The 90- and 99-percentile latencies of UPDATE queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. Workload consisted of 50% reads and 50% writes, randomly targeting a "realistic" Gaussian distribution. Cassandra 3 quickly became nonoperational, Cassandra 4 was a little better. Meanwhile Scylla maintained low and consistent write latencies across the entire range.

Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	80k/s	40k/s	30k/s	1.33x	2x
Maximum throughput with 90% latency < 10ms	80k/s	30k/s	10k/s	3x	2.66x
Maximum throughput with 99% latency < 10ms	80k/s	30k/s	-	-	2.66x



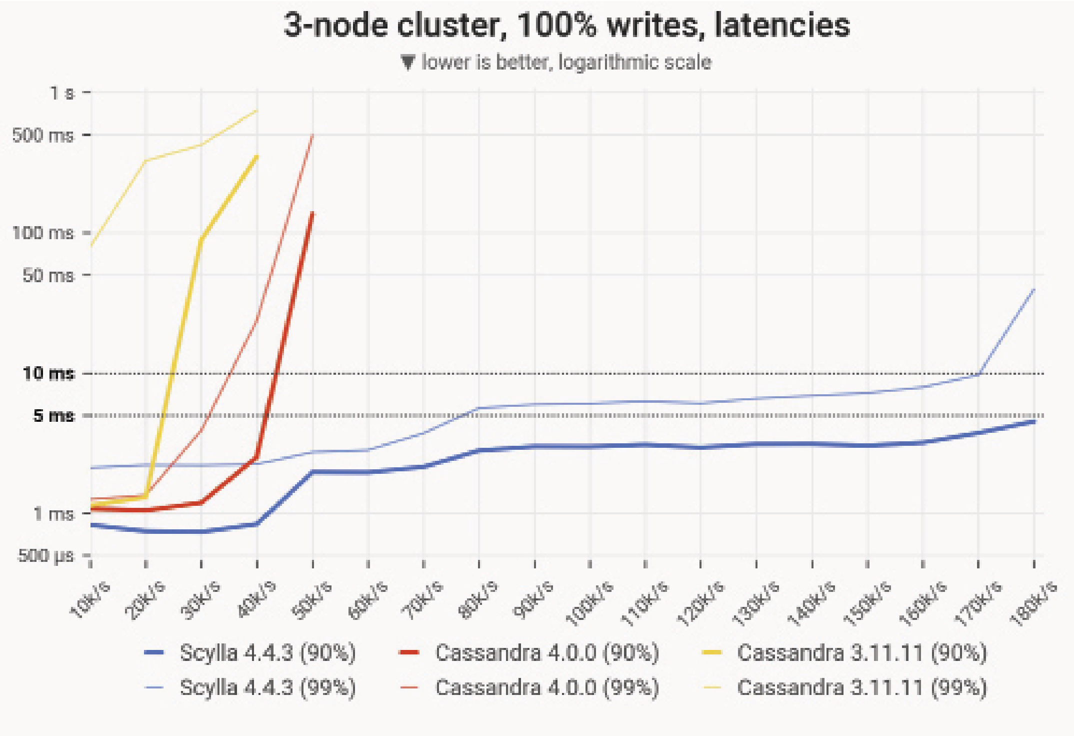
The 90- and 99-percentile latencies of SELECT queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload consisted of 50% reads and 50% writes, randomly targeting a "realistic" Gaussian distribution. Cassandra 3 quickly became nonoperational, Cassandra 4 performed a little better. Meanwhile Scylla maintained low and consistent response times across the entire range.

Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	90k/s	40k/s	40k/s	1x	2.25x
Maximum throughput with 90% latency < 10ms	80k/s	30k/s	10k/s	3x	2.66x
Maximum throughput with 99% latency < 10ms	70k/s	10k/s	-	-	7x

UNIFORM DISTRIBUTION (DISK-INTENSIVE, LOW CACHE HIT RATIO)

In this scenario we issued queries that touched random partitions of the **entire dataset**. In our setup this should result in high disk traffic and/or negligible cache hit rates, i.e. that of a few %.

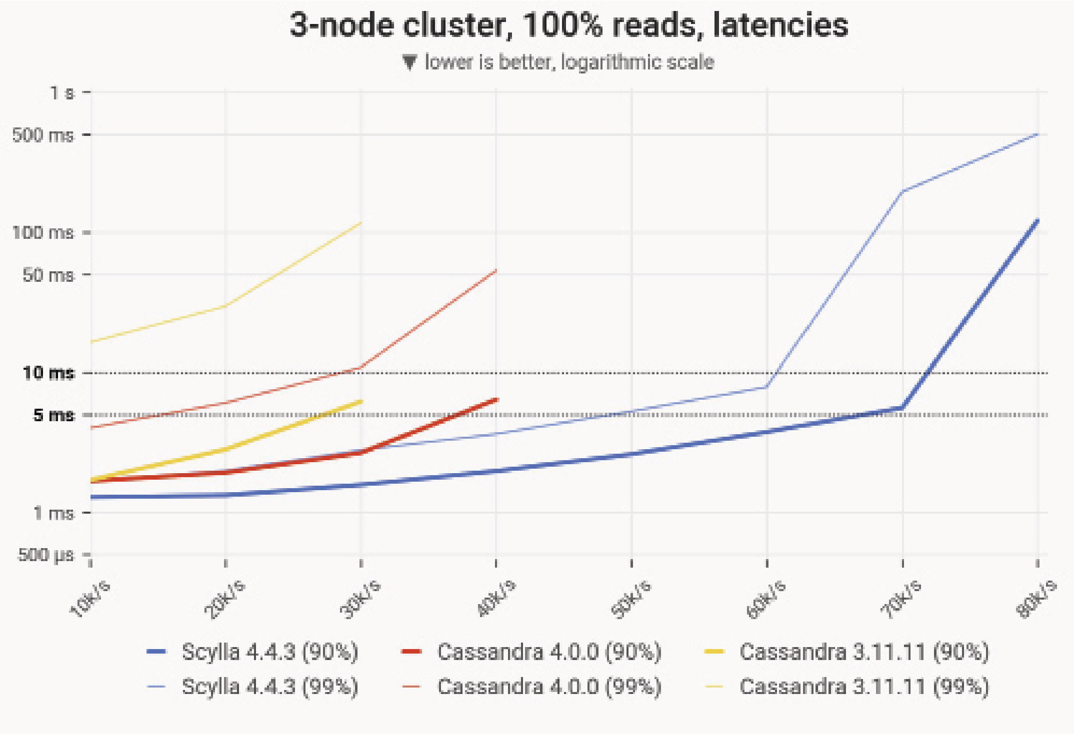
Writes Workload – Only Writes



The 90- and 99-percentile latencies of UPDATE queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. Workload was uniformly distributed, i.e. every partition in the 1 TB dataset had an equal chance of being updated. Cassandra 3 quickly became nonoperational. Cassandra 4 performed a little better. Meanwhile Scylla maintained low and consistent write latencies up until 170,000-180,000 ops/s.

Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	180k/s	50k/s	40k/s	1.25x	3.6x
Maximum throughput with 90% latency < 10ms	180k/s	40k/s	20k/s	2x	4.5x
Maximum throughput with 99% latency < 10ms	170k/s	30k/s			5.66x

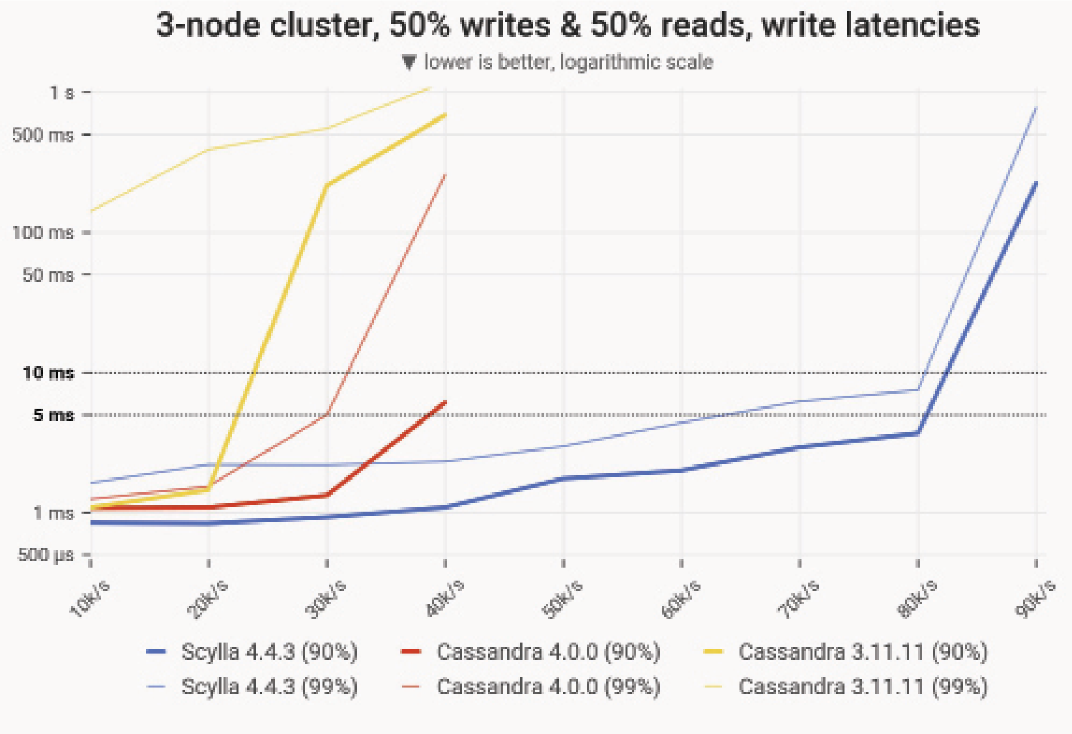
Reads Workload – Only Reads



The 90- and 99-percentile latencies of SELECT queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed, i.e. every partition in the 1 TB dataset had an equal chance of being selected. Scylla served 90% of queries in <5 ms until the load reached 70,000 ops/s. Please note that almost all reads were served from disk.

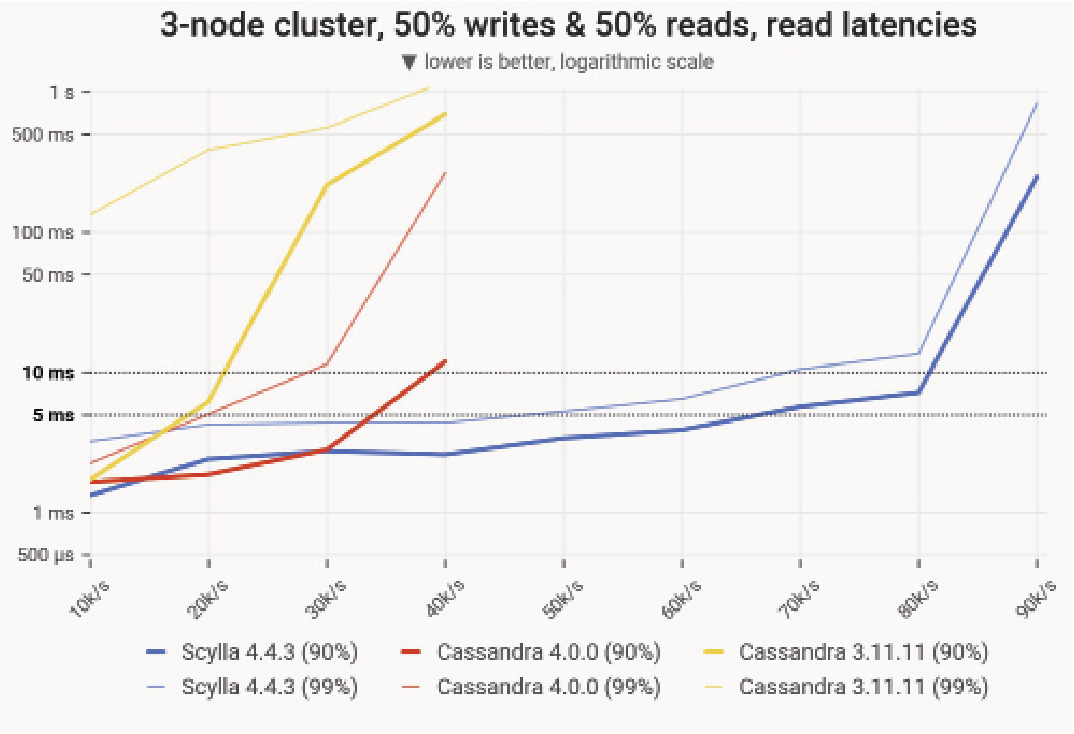
Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	80k/s	40k/s	30k/s	1.25x	2x
Maximum throughput with 90% latency < 10ms	70k/s	40k/s	30k/s	1.25x	1.75x
Maximum throughput with 99% latency < 10ms	60k/s	20k/s			3x

Mixed Workload – 50% reads and 50% writes



The 90- and 99-percentile latencies of UPDATE queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed, i.e. every partition in the 1 TB dataset had an equal chance of being selected/updated. At 80,000 ops/s Scylla maintained the latencies of 99% of queries in a single-figure regime (in milliseconds).

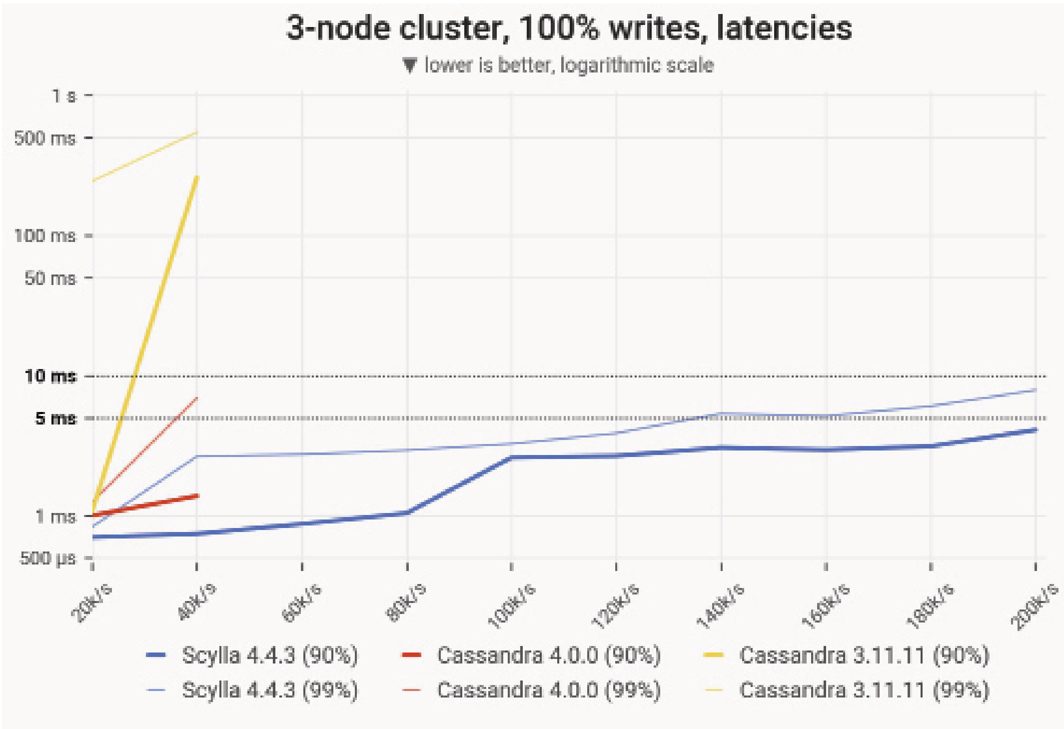
Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	90k/s	40k/s	40k/s	1x	2.25x
Maximum throughput with 90% latency < 10ms	80k/s	40k/s	20k/s	2x	2x
Maximum throughput with 99% latency < 10ms	80k/s	30k/s			2.66x



The 90- and 99-percentile latencies of SELECT queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed, i.e. every partition in the 1 TB dataset had an equal chance of being selected/ updated. Under such conditions Scylla handled 2x more traffic than the Cassandra releases and offered highly predictable response times.

Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	90k/s	40k/s	40k/s	1x	2.25x
Maximum throughput with 90% latency < 10ms	80k/s	30k/s	20k/s	1.5x	2.66x
Maximum throughput with 99% latency < 10ms	60k/s	20k/s			3x

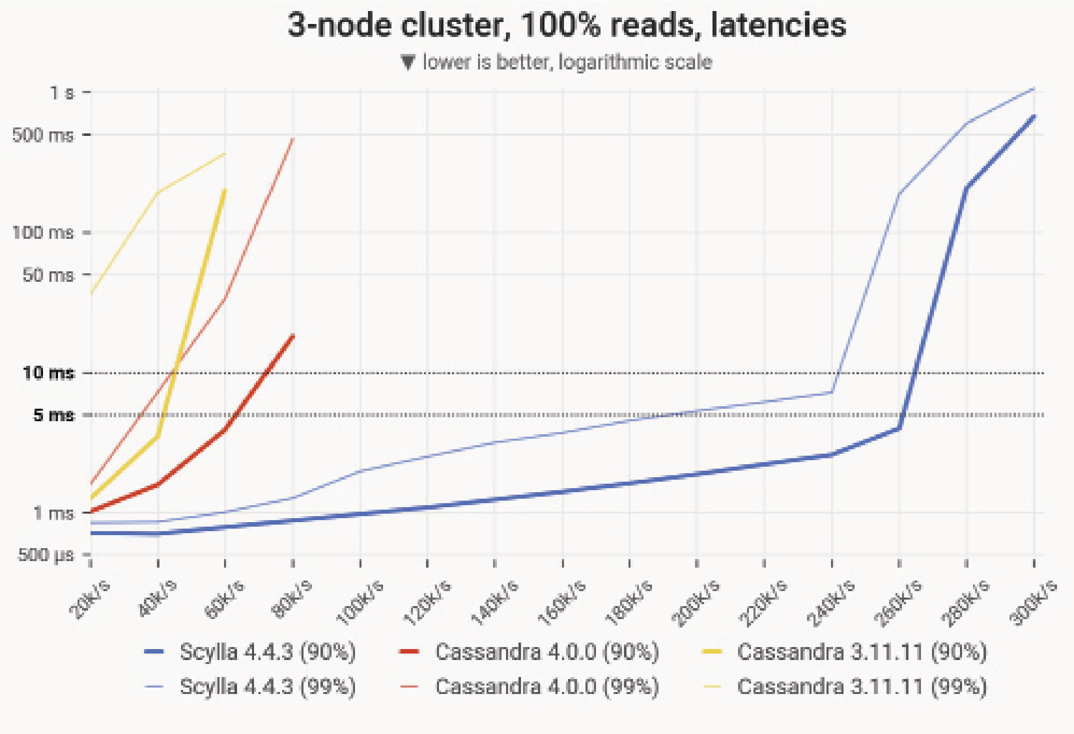
Writes Workload – Only Writes



The 90- and 99-percentile latencies of UPDATE queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed over 60 GB of data, so that every partition resided in cache and had an equal chance of being updated. The Cassandra databases instantly became nonoperational; Scylla withstood load more than 5x that of the Cassandra releases and maintained low and consistent write latencies over the entire range.

Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	200k/s	40k/s	40k/s	1x	5x
Maximum throughput with 90% latency < 10ms	200k/s	40k/s	20k/s	2x	5x
Maximum throughput with 99% latency < 10ms	200k/s	40k/s			5x

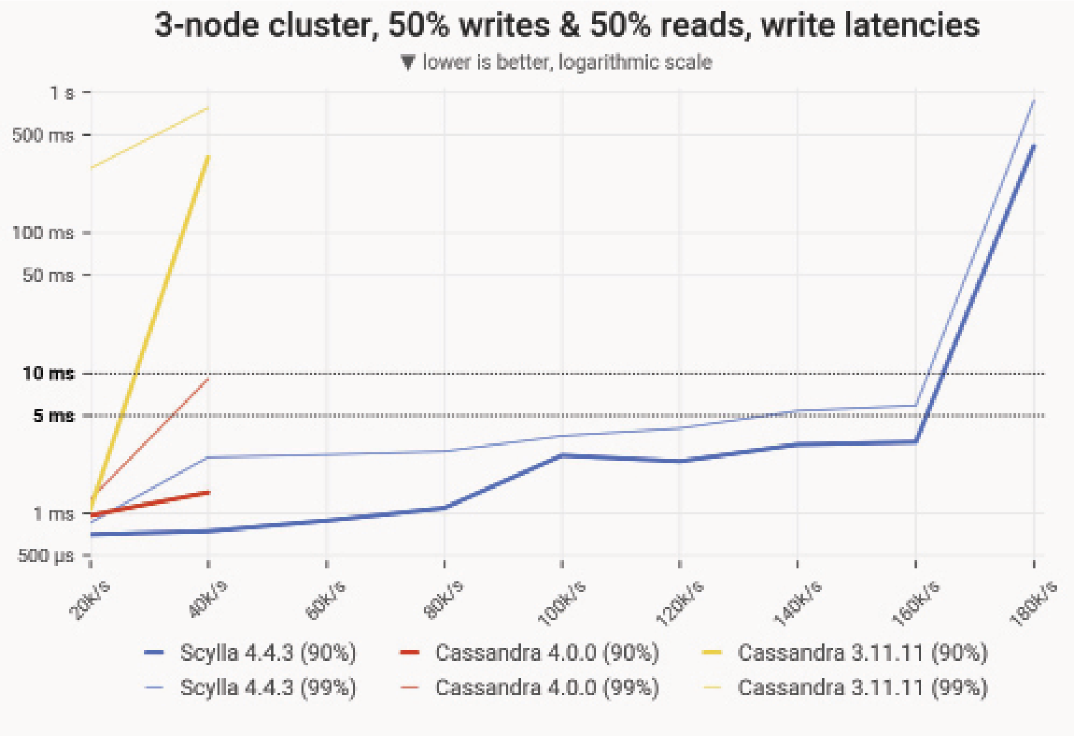
Reads Workload – Only Reads



The 90- and 99-percentile latencies of SELECT queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed over 60 GB of data, so that every partition resided in cache and had an equal chance of being selected. Scylla withstood load more than 3x higher than Cassandra 4 and 4x greater than Cassandra 3.

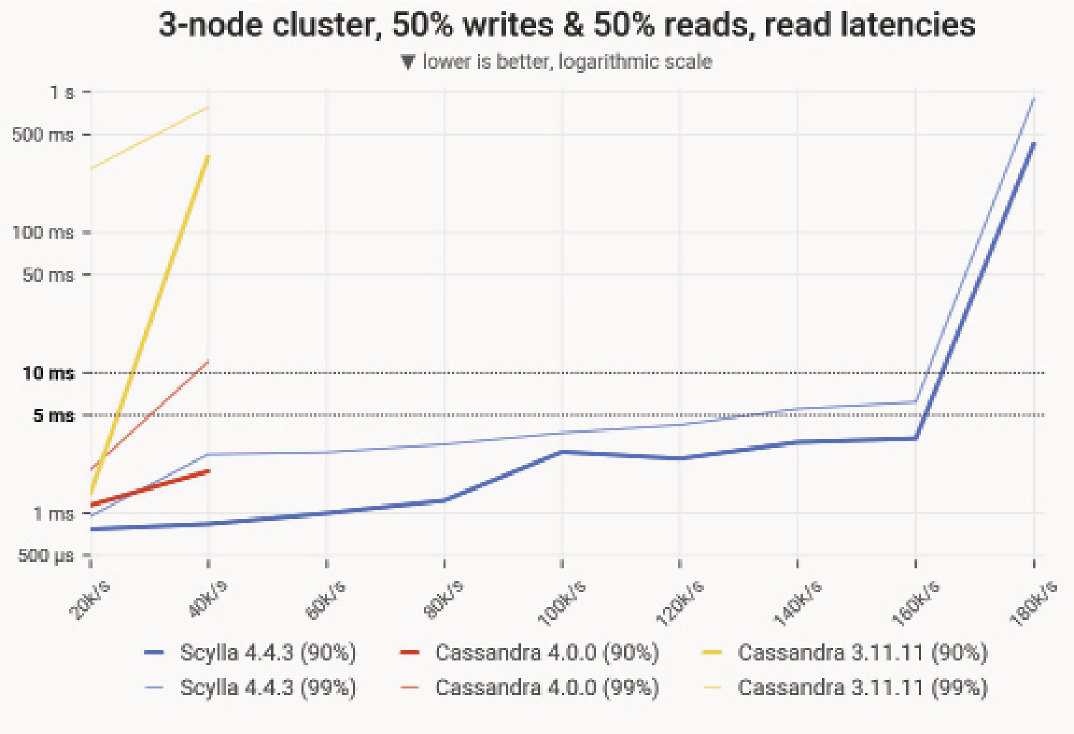
Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	300k/s	80k/s	60k/s	1.33x	3.75x
Maximum throughput with 90% latency < 10ms	260k/s	60k/s	40k/s	1.5x	4.33x
Maximum throughput with 99% latency < 10ms	240k/s	40k/s	-	-	6x

Mixed Workload – 50% reads and 50% writes



The 90- and 99-percentile latencies of UPDATE queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed over 60 GB of data, so that every partition resided in cache and had an equal chance of being selected/updated. Scylla withstood load over 3x higher than either of the Cassandra releases.

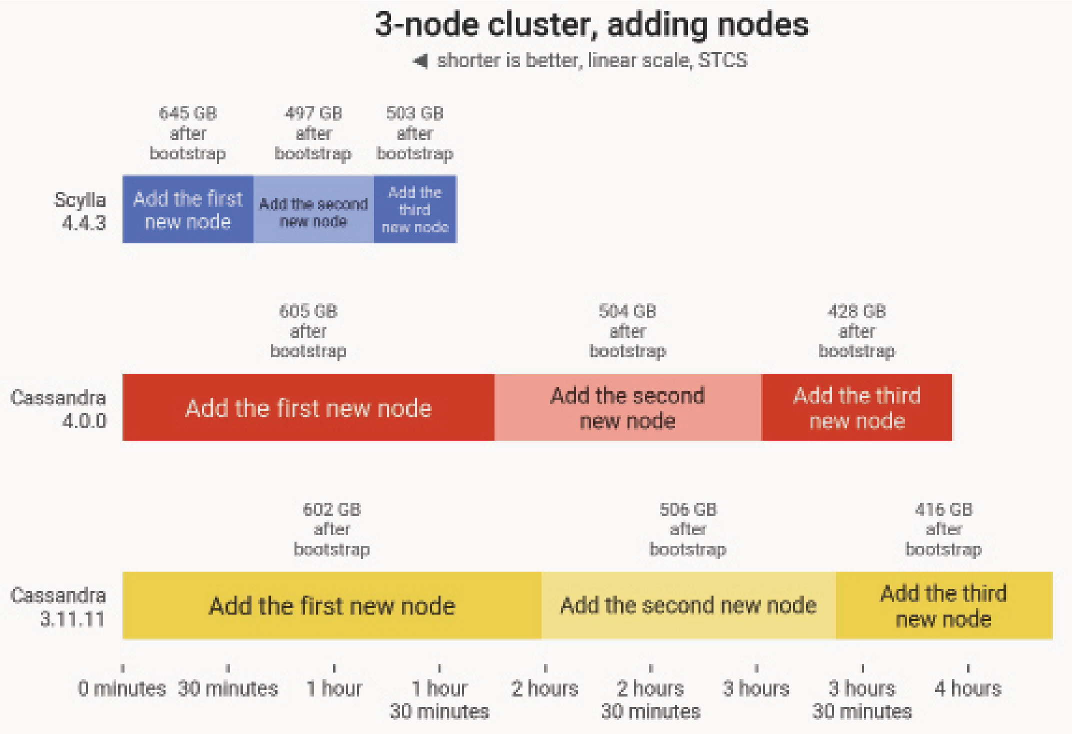
Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	180k/s	40k/s	40k/s	1x	4.5x
Maximum throughput with 90% latency < 10ms	160k/s	40k/s	20k/s	2x	4x
Maximum throughput with 99% latency < 10ms	160k/s	40k/s	-	-	4x



The 90- and 99-percentile latencies of SELECT queries, as measured on three i3.4xlarge machines (48 vCPUs in total) in a range of load rates. The workload was uniformly distributed over 60 GB of data, so that every partition resided in cache and had an equal chance of being selected/updated. Scylla withstood load more than 3x higher than either of the Cassandra releases.

Metric	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	180k/s	40k/s	40k/s	1x	4.5x
Maximum throughput with 90% latency < 10ms	160k/s	40k/s	20k/s	2x	4x
Maximum throughput with 99% latency < 10ms	160k/s	20k/s	-	-	8x

ADDING NODES

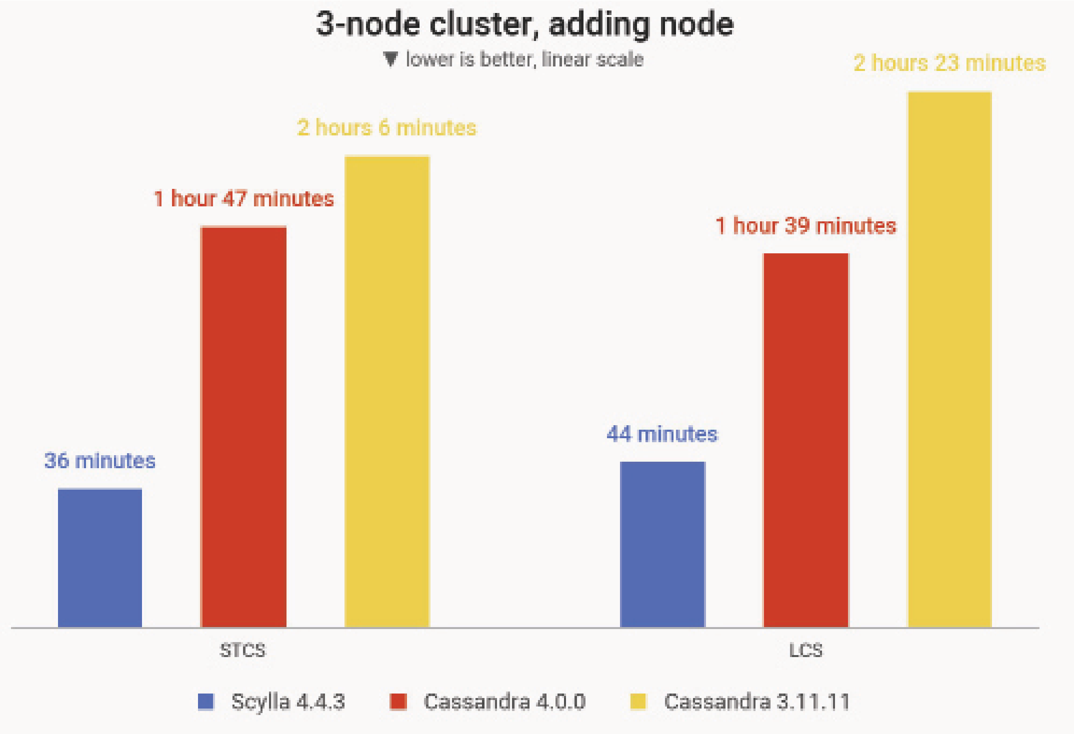


The timeline of adding 3 nodes to an existing 3-node cluster (resulting in six i3.4xlarge machines). Total time for Scylla 4.4 to double the cluster size was 94 minutes 57 seconds. For Cassandra 4.0, it took 238 minutes 21 seconds (just shy of 4 hours); Cassandra 3.11 took 270 minutes (4.5 hours). While Cassandra 4.0 noted a 12% improvement over Cassandra 3.11, Scylla completed the entire operation before either version of Cassandra bootstraped its first new node.

ONE NEW NODE

In this benchmark, we measured how long it takes to add a new node to the cluster. The reported times are the intervals between starting a Scylla/Cassandra node and having it fully finished bootstrapping (CQL port open).

Cassandra 4.0 is equipped with a new feature — Zero Copy Streaming — which allows for efficient streaming of entire SSTables. An SSTable is eligible for ZCS if all of its partitions need to be transferred, which can be the case when `LeveledCompactionStrategy` (LCS) is enabled. Willing to demonstrate this feature, we run the next benchmarks with the usual `SizeTieredCompactionStrategy` (STCS) compared to LCS.



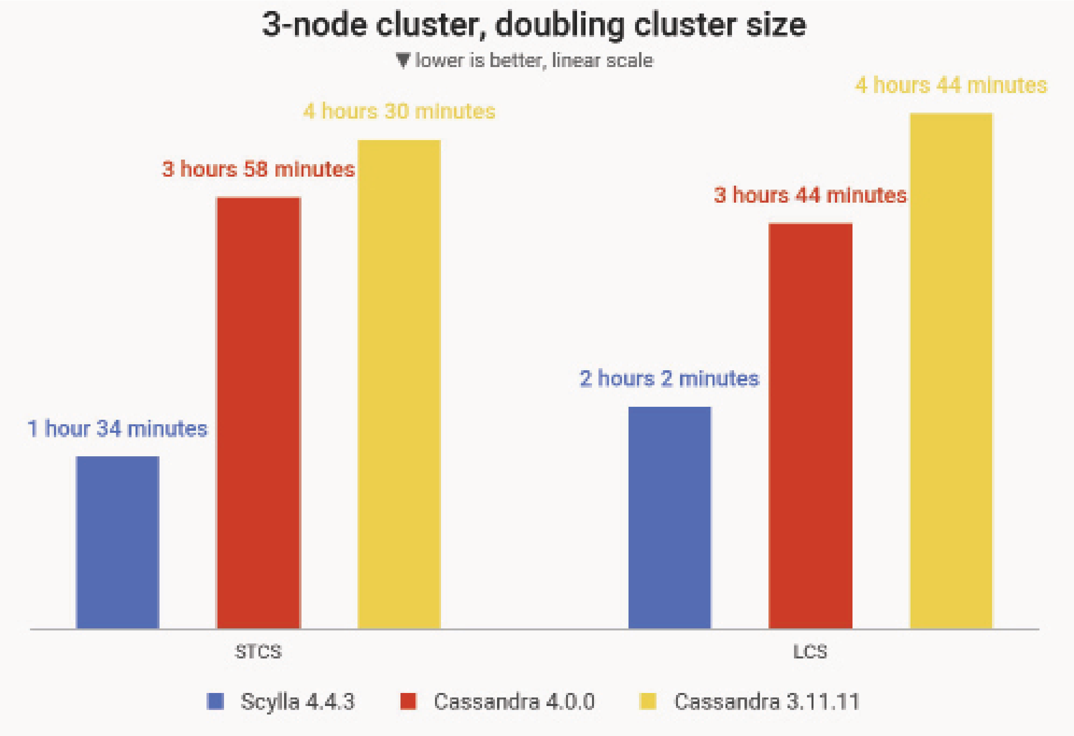
The time needed to add a node to an already existing 3-node cluster (resulting in 4 i3.4xlarge machines). The cluster was initially loaded with 1 TB of data at RF=3. Cassandra 4.0 showed an improvement over Cassandra 3.11, but Scylla still out-performed the Cassandra releases by a large margin.

Strategy	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11
STCS	36 minutes 56 seconds	1 hour 47 minutes 1 second	2 hours 6 minutes
LCS	44 minutes 11 seconds	1 hour 39 minutes 45 seconds	2 hours 23 minutes 10 seconds

Strategy	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
STCS	-15%	-65%
LCS	-30%	-55%

DOUBLING CLUSTER SIZE

In this benchmark, we measured how long it takes to double the cluster node count, going from 3 nodes to 6 nodes. Three new nodes were added sequentially, i.e. waiting for the previous one to fully bootstrap before starting the next one. The reported time spans from the instant the startup of the first new node was initiated, all the way until the bootstrap of the third new node finished.



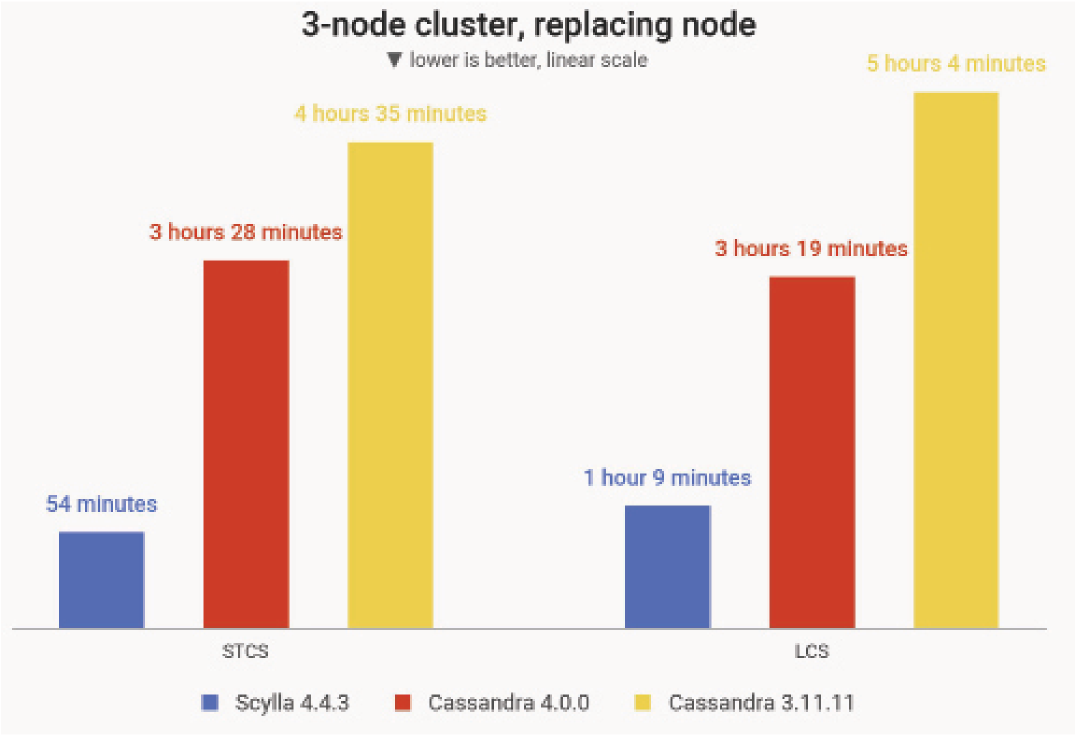
The time needed to add 3 nodes to an existing 3-node cluster of i3.4xlarge machines, preloaded with 1 TB of data at RF=3. Cassandra 4.0 performed moderately better than Cassandra 3.11. but Scylla outperformed the other databases.

Strategy	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11
STCS	1 hour 34 minutes 57 seconds	3 hours 58 minutes 21 seconds	4 hours 30 minutes 7 seconds
LCS	2 hours 2 minutes 37 seconds	3 hours 44 minutes 6 seconds	4 hours 44 minutes 46 seconds

Strategy	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
STCS	-11%	-60%
LCS	-21%	-45%

REPLACE NODE

In this benchmark, we measured how long it took to replace a single node. One of the nodes was brought down and another one was started in its place. Throughout this process the cluster was being agitated by a mixed R/W background load of 25,000 ops at CL=QUORUM.



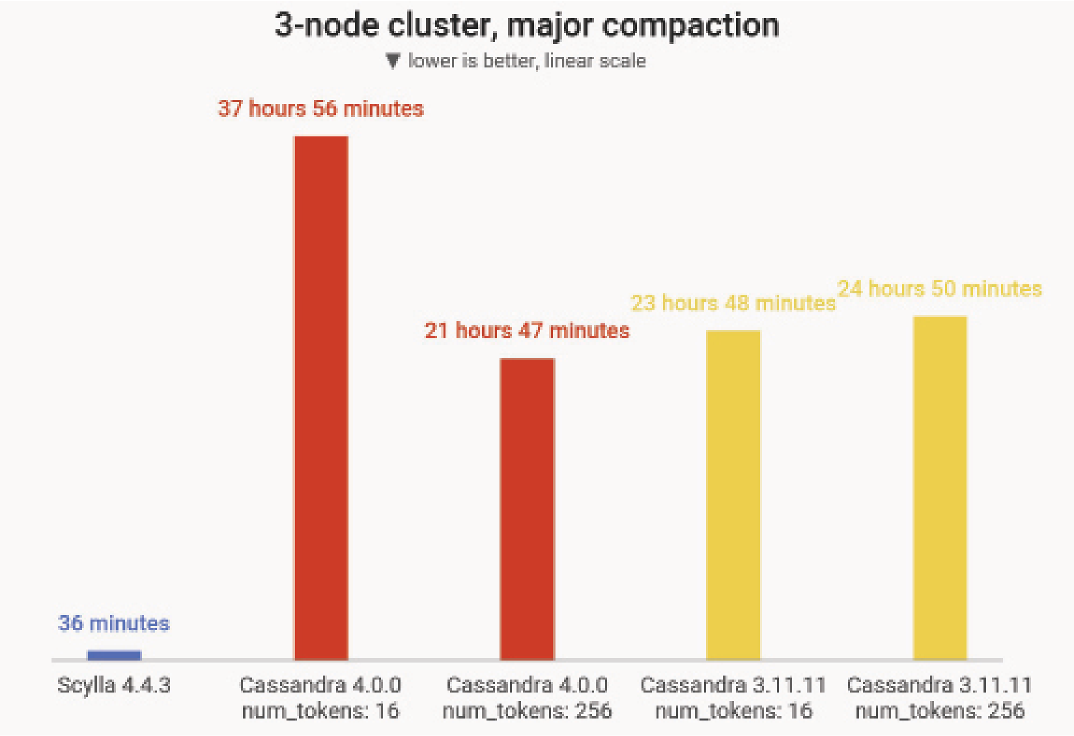
The time needed to replace a node in a 3-node cluster of i3.4xlarge machines, preloaded with 1 TB of data at RF=3. Cassandra 4.0 noted an improvement over Cassandra 3.11. but Scylla was still the clear winner, taking about an hour to do what Cassandra 4.0 took more than 3 hours to accomplish.

Strategy	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11
STCS	54 minutes, 19 seconds	3 hours, 28 minutes, 46 seconds	4 hours, 35 minutes, 56 seconds
LCS	1 hour, 9 minutes, 18 seconds	3 hours, 19 minutes, 17 seconds	5 hours, 4 minutes, 9 seconds

Strategy	Cassandra 4.0 vs Cassandra 3.11	Scylla 4.4.3 vs Cassandra 4.0
STCS	-24%	-73%
LCS	-34%	-65%

MAJOR COMPACTION

In this benchmark, we measured how long it took to replace a single node. One of the nodes was In this benchmark, we measured how long it takes to perform a major compaction on a single node loaded with roughly 1TB of data. Thanks to Scylla's sharded architecture, it can perform the major compactions on each shard concurrently, while Cassandra is single-thread bound. The result of major compaction is the same in both Scylla and Cassandra: A read is served by a single SSTable. In the later section of this paper we also measure the speed of a major compaction in a case where there are many small Cassandra nodes (which get higher parallelism). We observed worse major compaction performance in Cassandra 4.0.0 with the default `num_tokens: 16` parameter.



Major compaction of 1 TB of data at RF=1 on i3.4xlarge machine. Scylla demonstrated the power of a sharded architecture by compacting on all cores concurrently. In this case Scylla was up to 60x faster. This figure should continue to scale linearly with the number of cores.

	Scylla 4.4.3	Cassandra 4.0	Cassandra 3.11
Major Compaction (num_ tokens: 16)	num_tokens: 16 not recommended	21 hours, 47 minutes, 34 seconds (78,454 seconds)	24 hours, 50 minutes, 42 seconds (89,442 seconds)
Major Compaction (num_ tokens: 256)	36 minutes, 8 seconds (2,168 seconds)	37 hours, 56 minutes, 32 seconds (136,592 seconds)	23 hours, 48 minutes, 56 seconds (85,736 seconds)

“4 VS. 40” BENCHMARK

Now let us compare the databases installed on **different hardware**. In this scenario, Scylla gets four powerful 72-core servers, while Cassandra gets 40 of the same i3.4xlarge servers as before. Why would anyone ever consider such a test? After all, we’re comparing some 4 machines to 40 **very different** machines. (In terms of CPU count, RAM volume or cluster topology this would appear to be an apples-to-oranges comparison.)

However, due to its sharded architecture and custom memory management Scylla can utilize very large hunks of hardware. Meanwhile, Cassandra and its JVM’s garbage collectors excel when they are heavily distributed, with many smaller nodes on the team. So, the true purpose of this test is to show that both CQL solutions can perform similarly in a pretty fair duel, yet Cassandra requires about **2.5x more hardware, for 2.5x the cost**. What’s really at stake now is a **reduction in the administrative burden**, where a DBA would have either 40 servers to maintain or just 4.

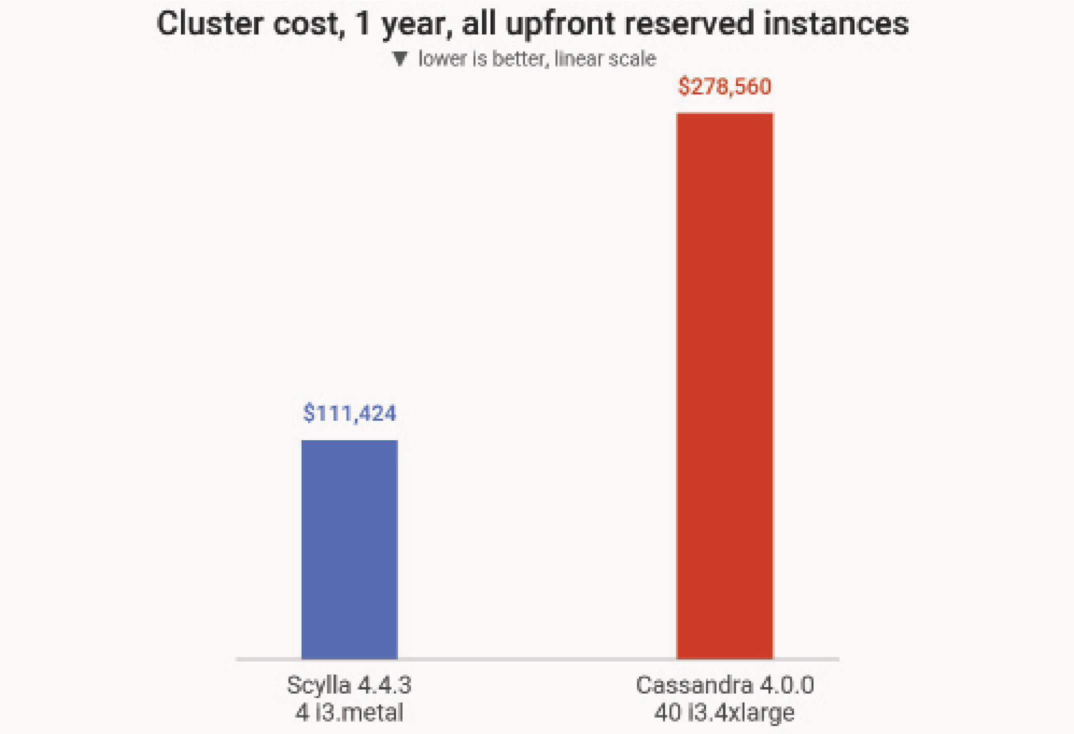
4 VS. 40 NODE SETUP

We set up clusters on Amazon EC2 in a single Availability Zone within us-east-2 datacenter, **but this time the Scylla cluster consisted of 4 i3.metal VMs**. The competing Cassandra cluster consisted of 40 i3.4xlarge VMs. Servers were initialized with clean machine images (AMIs) of Ubuntu 20.04 (Cassandra 4.0) or CentOS 7.9 (Scylla 4.4).

Apart from the cluster, fifteen loader machines were used to run cassandra-stress to insert data, and – later – to provide background load at CL=QUORUM to mess with the administrative operations.

	Scylla	Cassandra	Loaders
EC2 Instance type	i3.metal	i3.4xlarge	c5n.9xlarge
Cluster size	4	40	15
Storage (total)	8x 1.9 TB NVMe in RAID0 (60.8 TB)	2x 1.9 TB NVMe in RAID0 (152 TB)	Not important for a loader (EBS-only)
Network	25 Gbps	Up to 10 Gbps	50 Gbps
vCPUs (total)	72 (288)	16 (640)	36 (540)
RAM (total)	512 (2048) GiB	122 (4880) GiB	96 (1440) GiB

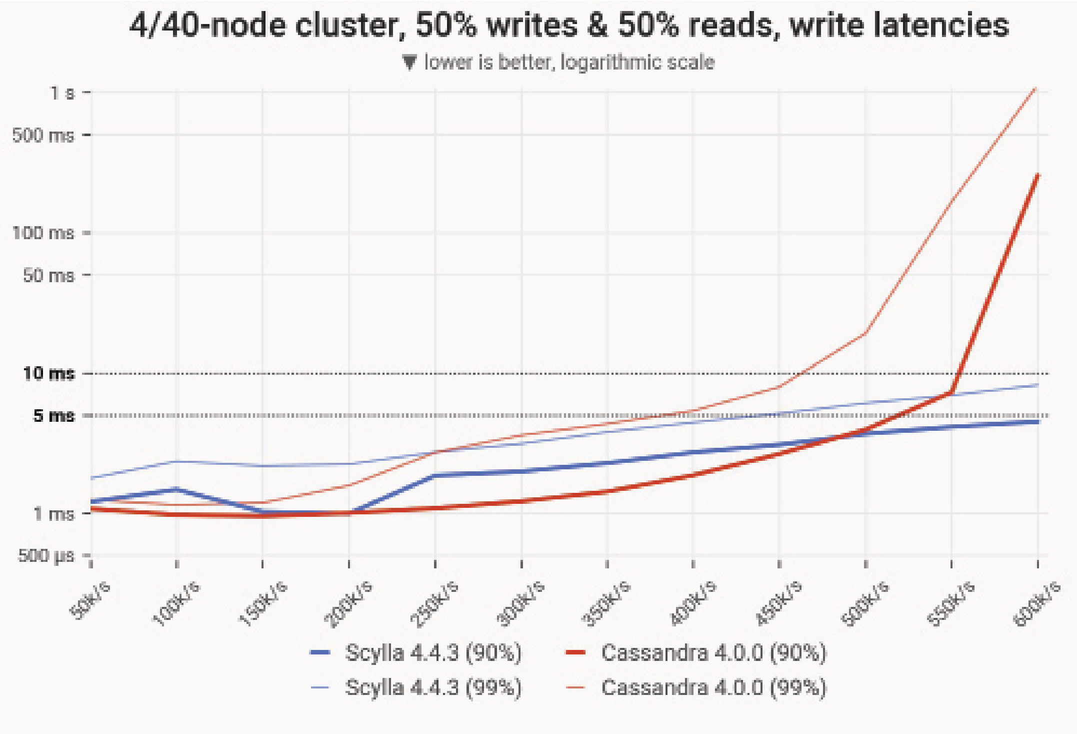
Once up and running, both databases were loaded with random data at RF=3 until the cluster’s total disk usage reached approximately 40 TB. This translated to 1 TB of data per Cassandra node and 10 TB of data per Scylla node. After loading was done, we flushed the data and waited until the compactions finished, so we could start the actual benchmarking.



A Scylla cluster can be 10x smaller in node count and run on a cluster 2.5x less expensive, yet maintain the equivalent performance of Cassandra 4.

THROUGHPUT AND LATENCIES

MIXED WORKLOAD – 50% READS AND 50% WRITES

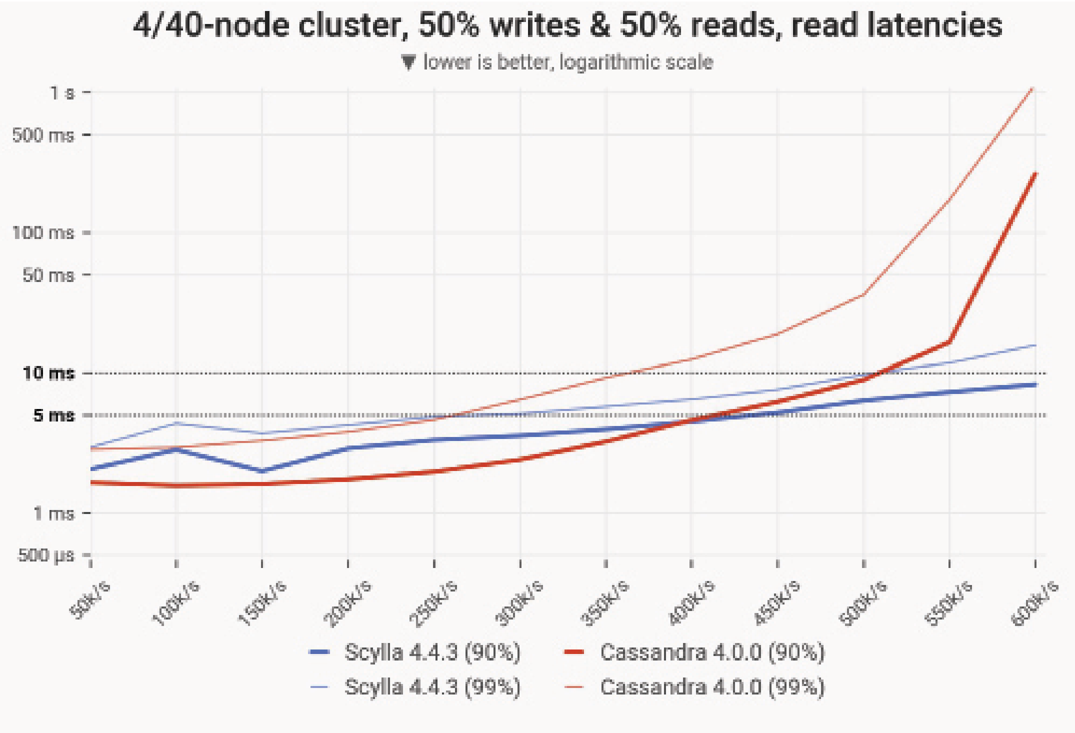


The 90- and 99-percentile latencies of UPDATE queries, as measured on:

- *4-node Scylla cluster (4 x i3.metal, 288 vCPUs in total)*
- *40-node Cassandra cluster (40 x i3.4xlarge, 640 vCPUs in total).*

The workload was uniformly distributed, i.e. every partition in the multi-TB dataset had an equal chance of being selected/updated. Under low load Cassandra slightly outperformed Scylla.The reason for this is that Scylla runs more compaction automatically when it is idle and the default scheduler tick of 0.5 ms hurts the P99 latency. (Note, there is a parameter that controls this but we wanted to provide out-of-the-box results with zero custom tuning or configuration.)

Metric	Scylla 4.4.3	Cassandra 4.0	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	600k/s	600k/s	1x
Maximum throughput with 99% latency < 10ms	600k/s	450k/s	1.33x



The 90- and 99-percentile latencies of SELECT queries, as measured on:

- *4-node Scylla cluster (4 x i3.metal, 288 vCPUs in total)*
- *40-node Cassandra cluster (40 x i3.4xlarge, 640 vCPUs in total).*

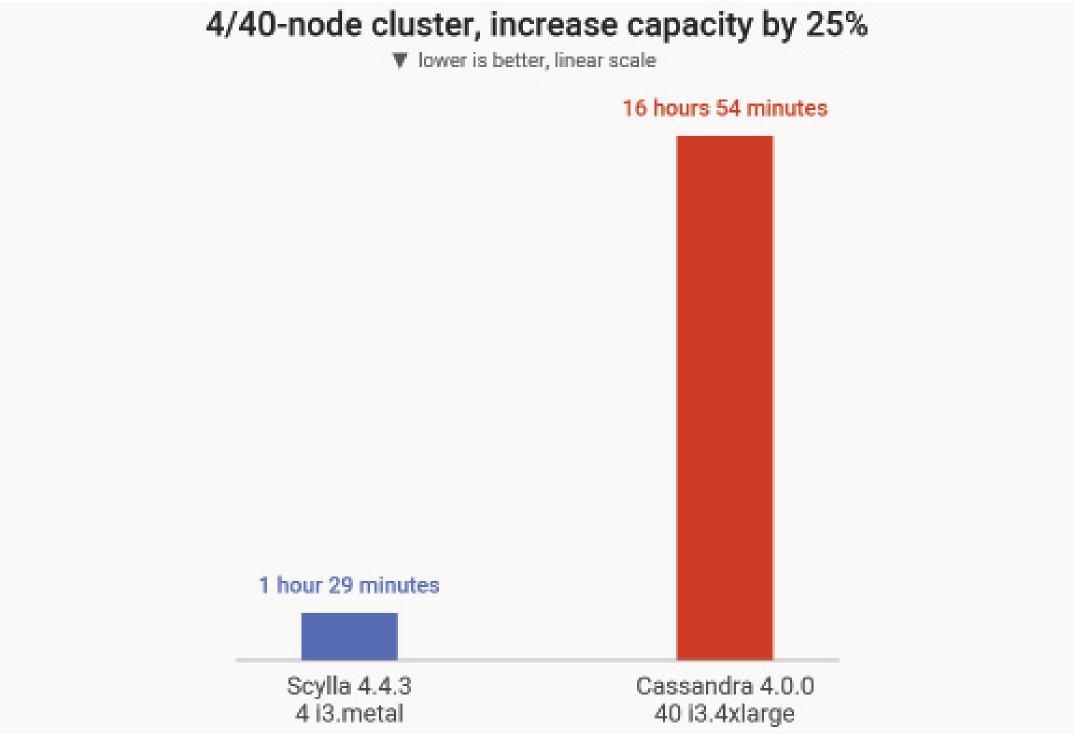
The workload was uniformly distributed, i.e. every partition in the multi-TB dataset had an equal chance of being selected/updated. Under low load Cassandra slightly outperformed Scylla.

Metric	Scylla 4.4.3	Cassandra 4.0	Scylla 4.4.3 vs Cassandra 4.0
Maximum throughput	600k/s	600k/s	1x
Maximum throughput with 99% latency < 10ms	500k/s	350k/s	1.42x

SCALING THE CLUSTER UP BY 25%

In this benchmark, we increase the capacity of the cluster by 25%:

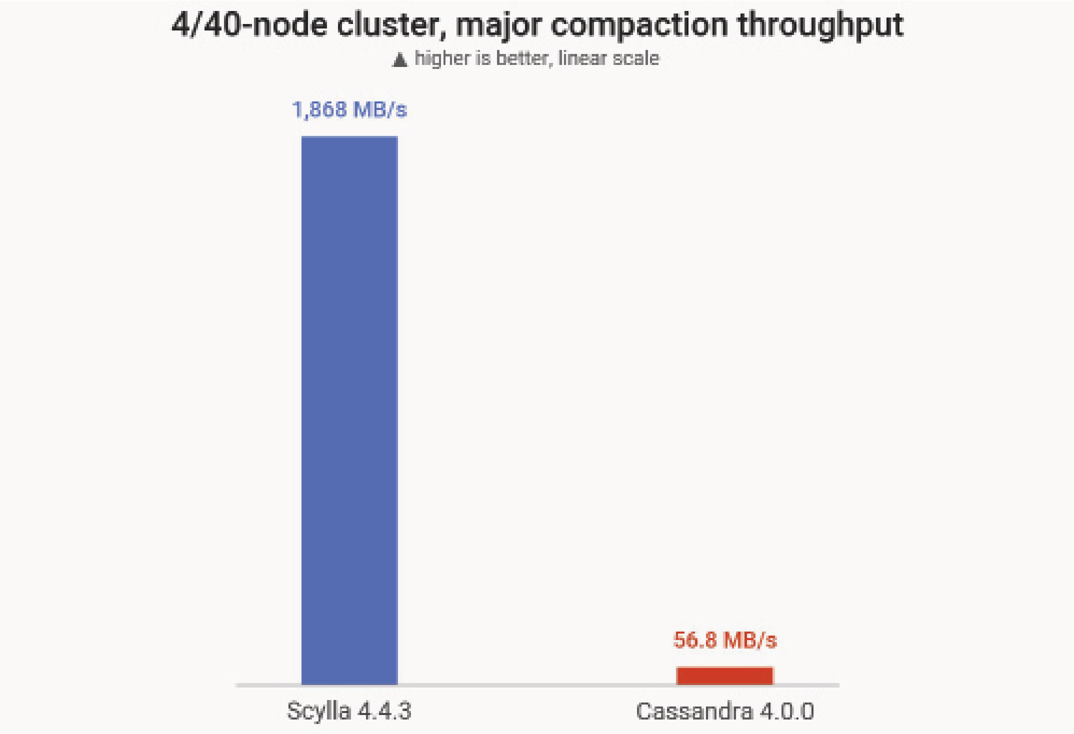
- By adding a single Scylla node to the cluster (from 4 nodes to 5)
- By adding 10 Cassandra nodes to the cluster (from 40 nodes to 50 nodes)



Metric	Scylla 4.4.3	Cassandra 4.0	Scylla 4.4 vs Cassandra 4.0
Add 25% capacity	1 hour, 29 minutes	16 hours, 54 minutes	**11x faster**

MAJOR COMPACTION

In this benchmark we measure the throughput of a major compaction. To compensate for Cassandra having 10 times more nodes (each having 1/10th of the data), this benchmark measures the throughput of a single Scylla node performing major compaction and the collective throughput of 10 Cassandra nodes performing major compactions concurrently.



Throughput of a major compaction at RF=1 (more is better). Scylla ran on a single i3.metal machine (72 vCPUs) and competed with a 10-node cluster of Cassandra 4 (10x i3.4xlarge machines; 160 vCPUs in total). Scylla can split this problem across CPU cores, which Cassandra cannot do, so – effectively – Scylla performed 32x better in this case.

	Scylla 4.4.3	Cassandra 4.0	Scylla 4.4 vs Cassandra 4.0
Major Compaction	1868 MB/s	56.8 MB/s	**32x faster**

SUMMARY

On identical hardware, Scylla 4.4.3 withstood up to 5x greater traffic and – in almost every scenario – offered lower latencies than Cassandra 4.0. We also demonstrated a specific use-case where choosing Scylla over Cassandra 4.0 would result in $170,000 monthly savings in hardware costs alone, without factoring in reduced administration costs or environmental impact.

Nonetheless, Cassandra 4.0 is a significant improvement over Cassandra 3.x. It has aptly piggy-backed on advancements to the JVM. Upgrading from Cassandra 3 to Cassandra 4 will benefit many use cases.

However, organizations that have already decided to upgrade from Cassandra 3.x should first consider all their options. Upgrading Cassandra involves backups, risk of downtime, and a sleepless night or two. Those who are determined to take this effort, should consider the return they will receive for their efforts — in terms of performance and cost savings. The benchmarks in this study demonstrate that Scylla is not only far more performant but also much more affordable. Additionally, all the information required to re-rerun these benchmarks is provided herein.

SUPPLEMENTARY INFORMATION

Here you can check out detailed results of latency/throughput benchmarks, JVM settings and cassandra.yaml from Cassandra 3 and Cassandra 4, as well as `cassandra-stress` invocations used to run benchmarks. Scylla used default configuration.

CASSANDRA 3.11 CONFIGURATION

JVM settings	JVM version: OpenJDK 8
	`-Xms48G`
	`-Xmx48G`
	`-XX:+UseG1GC`
	`-XX:G1RSetUpdatingPauseTimePercent=5`
	`-XX:MaxGCPauseMillis=500`
	`-XX:InitiatingHeapOccupancyPercent=70`
	`-XX:ParallelGCThreads=16`

cassandra.yaml	Only settings changed from the default configuration are mentioned here.
	`disk_access_mode: mmap_index_only`
	`row_cache_size_in_mb: 10240`
	`concurrent_writes: 128`
	`file_cache_size_in_mb: 2048`
	`buffer_pool_use_heap_if_exhausted: true`
	`disk_optimization_strategy: ssd`
	`memtable_flush_writers: 4`
	`trickle_fsync: true`
	`concurrent_compactors: 16`
	`compaction_throughput_mb_per_sec: 960`
	`stream_throughput_outbound_megabits_per_sec: 7000`

CASSANDRA 4.0 CONFIGURATION

<table>
<tr><td>JVM settings</td><td>JVM version: OpenJDK 16
<pre>
-Xmx70G
-Xmx70G
-XX:ConcGCThreads=16
-XX:+UseZGC
-XX:ConcGCThreads=16
-XX:ParallelGCThreads=16
-XX:+UseTransparentHugePages
-verbose:gc
-Djdk.attach.allowAttachSelf=true
-Dio.netty.tryReflectionSetAccessible=true
</pre></td></tr>
<tr><td>cassandra.yaml</td><td>Only settings changed from the default configuration are mentioned here.
<pre>
disk_access_mode: mmap_index_only
row_cache_size_in_mb: 10240
concurrent_writes: 128
file_cache_size_in_mb: 2048
buffer_pool_use_heap_if_exhausted: true
disk_optimization_strategy: ssd
memtable_flush_writers: 4
trickle_fsync: true
concurrent_compactors: 16
compaction_throughput_mb_per_sec: 960
stream_throughput_outbound_megabits_per_sec: 7000
</pre>
In major compaction benchmarks, the parameter <code>compaction_throughput_mb_per_sec</code> was set to <code>0</code> to make sure the compaction was not throttled.</td></tr>
</table>

CASSANDRA-STRESS PARAMETERS

Only the important facts and options are mentioned below.

- Scylla's Shard-aware Java driver was used.
- Background loads were executed in the loop (so `duration=5m` is not a problem).
- `REPLICATION_FACTOR` is 3 (except for major compaction benchmark).
- `COMPACTION_STRATEGY` is `SizeTieredCompactionStrategy` unless stated otherwise.
- `loadgenerator_count` is the number of generator machines (3 for "3 vs 3" benchmarks, 15 for "4 vs 40").
- `BACKGROUND_LOAD_OPS` is 1000 in major compaction, 25000 in other benchmarks.
- `DURATION_MINUTES` is 10 for memory-intensive benchmarks, 30 for other benchmarks.

<table>
<tr><td>Inserting data</td><td><pre><code>write cl=QUORUM
-schema “replication(strategy=SimpleStrategy,replication_
factor={REPLICATION_FACTOR})” “compaction(strategy={COMPACTI
ON_STRATEGY})”
-mode native cql3</code></pre>
<code>threads</code> and <code>throttle</code> parameters were chosen for each DB separately, to ensure 3TB or 40TB were inserted quickly, but to provide headroom for minor compactions and avoid timeouts/large latencies.
In case of “4 vs 40” benchmarks additional parameter <code>maxPending=1024</code> was used. </td></tr>
<tr><td>Background load for replace node</td><td><pre><code>mixed ratio(write=1,read=1)
duration=5m
cl=QUORUM
-pop dist=UNIFORM(1..{ROW_COUNT})
-mode native cql3
-rate “threads=700 throttle={BACKGROUND_LOAD_OPS
// loadgenerator_count}/s”</code></pre></td></tr>
<tr><td>Background load for new nodes / major compaction</td><td><pre><code>mixed ratio(write=1,read=1)
duration=5m
cl=QUORUM
-pop dist=UNIFORM(1..{ROW_COUNT})
-mode native cql3
-rate “threads=700 fixed={BACKGROUND_LOAD_OPS
// loadgenerator_count}/s”</code></pre></td></tr>
<tr><td>Cache warmup in Gaussian latency / throughput</td><td><pre><code>mixed ratio(write=0,read=1)
duration=180m
cl=QUORUM -pop dist=GAUSSIAN(1..
{ROW_COUNT},{GAUSS_CENTER},{GAUSS_SIGMA})
-mode native cql3
-rate “threads=500 throttle=35000/s” -node {cluster_string}')</code></pre></td></tr>
<tr><td>Latency / throughput - Gaussian</td><td><pre><code>duration={DURATION_MINUTES}m
cl=QUORUM
-pop dist=GAUSSIAN(1..{ROW_COUNT},{GAUSS_CENTER},{GAUSS_SIGMA})
-mode native cql3
“threads=500 fixed={rate // loadgenerator_count}/s”</code></pre></td></tr>
<tr><td>Latency / throughput - uniform / in-memory</td><td><pre><code>duration={DURATION_MINUTES}m
cl=QUORUM
-pop dist=UNIFORM(1..{ROW_COUNT})
-mode native cql3
-rate “threads=500 fixed={rate // loadgenerator_count}/s”</code></pre>
In case of “4 vs 40” benchmarks additional parameter <code>maxPending=1024</code> was used. </td></tr>
</table>

DETAILED RESULTS - THREE I3.4XLARGE NODES

GAUSSIAN DISTRIBUTION - MIXED WORKLOAD - 50% READS AND 50% WRITES

Scylla 4.4.3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	Write	0.68	0.66	0.81	1.03
	Read	1	1.01	1.24	2.05
20k/s	Write	0.66	0.6	0.75	2.37
	Read	1.04	0.99	1.26	3.56
30k/s	Write	0.69	0.6	0.8	2.53
	Read	1.09	1	1.4	4.37
40k/s	Write	0.83	0.65	0.97	2.62
	Read	1.29	1.11	1.78	4.56
50k/s	Write	0.97	0.76	1.86	3.02
	Read	1.64	1.32	3.05	5.1
60k/s	Write	1.37	1.05	2.45	3.97
	Read	2.39	1.95	4.41	6.47
70k/s	Write	1.5	1.15	2.6	4.75
	Read	2.5	2.02	4.5	7.54
80k/s	Write	3.55	1.77	3.7	7.37
	Read	5.11	3.06	6.61	13.54
90k/s	**Could not keep up with the load**				

Cassandra 4

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	Write	1.18	1.09	1.4	3.04
	Read	2.19	1.82	3.29	7.78
20k/s	Write	1.48	1.16	1.73	4.39
	Read	2.68	2.01	4.18	10.08
30k/s	Write	1.96	1.53	3.03	8.03
	Read	3.63	2.73	6.2	17.06
40k/s	Write	27.19	3.85	58.65	414.71
	Read	31.28	7.13	66.29	421.27
50k/s	**Could not keep up with the load**				

Cassandra 3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	Write	9.77	1.18	2.24	266.6
	Read	13	1.95	6.34	275.51
20k/s	Write	28.03	1.46	77.4	451.41
	Read	30.06	2.56	85.33	457.7
30k/s	Write	100.69	3.69	360.45	731.38
	Read	104.51	7.02	365.95	738.72
40k/s	**Could not keep up with the load**				

UNIFORM DISTRIBUTION (DISK-INTENSIVE) - WRITES WORKLOAD - ONLY WRITES

Scylla 4.4.3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	0.71	0.67	0.82	2.1
20k/s	0.69	0.61	0.74	2.2
30k/s	0.69	0.6	0.73	2.19
40k/s	0.76	0.63	0.83	2.23
50k/s	0.93	0.68	1.96	2.72
60k/s	0.97	0.72	1.95	2.81
70k/s	1.19	0.8	2.13	3.71
80k/s	2.61	2.4	2.79	5.61
90k/s	3.17	2.56	2.98	5.96
100k/s	3.1	2.56	2.97	6.05
110k/s	3.31	2.58	3.07	6.28
120k/s	2.75	2.44	2.93	6.07
130k/s	3.12	2.56	3.1	6.56
140k/s	3.12	2.55	3.11	6.89
150k/s	3.51	2.36	3.02	7.2
160k/s	4.1	2.49	3.18	7.92
170k/s	5.72	2.78	3.73	9.67
180k/s	7.87	2.99	4.58	39.32
190k/s	**Could not keep up with the load**			

Cassandra 4

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	0.9	0.88	1.07	1.25
20k/s	0.9	0.86	1.04	1.34
30k/s	1.31	0.92	1.18	3.87
40k/s	2.27	1.15	2.5	23.28
50k/s	38.13	2.26	136.31	494.67
60k/s	**Could not keep up with the load**			

Cassandra 3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	3.44	0.91	1.13	79.3
20k/s	12.12	0.94	1.29	325.58
30k/s	27.99	1.14	88.34	423.36
40k/s	89.41	2.38	347.34	740.82
50k/s	**Could not keep up with the load**			

UNIFORM DISTRIBUTION (LOW CACHE HIT RATIO) - READS WORKLOAD - ONLY READS

Scylla 4.4.3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	1.11	1.09	1.28	1.67
20k/s	1.13	1.07	1.32	1.98
30k/s	1.51	1.14	1.57	2.78
40k/s	1.59	1.3	1.97	3.63
50k/s	2.38	1.58	2.6	5.27
60k/s	3.74	2.09	3.75	7.87
70k/s	8.99	2.65	5.59	195.3
80k/s	38.98	4.56	121.11	504.63
90k/s	**Could not keep up with the load**			

Cassandra 4

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	1.43	1.35	1.67	4.01
20k/s	1.6	1.42	1.92	6.05
30k/s	2.14	1.71	2.66	10.85
40k/s	4.5	2.42	6.42	53.22
50k/s	**Could not keep up with the load**			

Cassandra 3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	2.46	1.35	1.7	16.33
20k/s	3.1	1.52	2.81	29.51
30k/s	6.07	2.11	6.25	117.05
40k/s	**Could not keep up with the load**			

UNIFORM DISTRIBUTION - MIXED WORKLOAD - 50% READS AND 50% WRITES

Scylla 4.4.3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	Write	0.71	0.68	0.84	1.62
	Read	1.17	1.11	1.32	3.21
20k/s	Write	0.73	0.64	0.83	2.19
	Read	1.36	1.1	2.4	4.23
30k/s	Write	0.79	0.65	0.92	2.18
	Read	1.46	1.15	2.74	4.37
40k/s	Write	0.92	0.71	1.08	2.3
	Read	1.59	1.27	2.59	4.37
50k/s	Write	1.26	0.83	1.74	2.96
	Read	2.15	1.56	3.39	5.27
60k/s	Write	2.43	0.99	1.99	4.36
	Read	3.38	1.95	3.87	6.47
70k/s	Write	4.11	1.54	2.91	6.21
	Read	5.68	3.22	5.69	10.5

80k/s	Write	4.54	1.96	3.68	7.5
	Read	6.64	4.02	7.18	13.53
90k/s	Write	59.32	2.99	224.66	773.85
	Read	67.48	6.28	248.64	827.85
100k/s	**Could not keep up with the load**				

Cassandra 4

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	Write	0.92	0.89	1.07	1.24
	Read	1.41	1.37	1.65	2.24
20k/s	Write	0.97	0.9	1.08	1.53
	Read	1.59	1.43	1.86	5.03
30k/s	Write	1.45	0.99	1.32	4.98
	Read	2.38	1.65	2.81	11.41
40k/s	Write	9.85	1.56	6.16	257.95
	Read	12.10	2.94	11.98	265.03
50k/s	**Could not keep up with the load**				

Cassandra 3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
10k/s	Write	4.52	0.89	1.08	141.16
	Read	5.41	1.36	1.71	133.23
20k/s	Write	17.67	0.94	1.44	391.12
	Read	18.45	1.51	6.16	388.24
30k/s	Write	48.88	1.3	216.14	553.12
	Read	50.53	2.3	217.84	556.79
40k/s	Write	252.97	146.28	687.34	1182.79
	Read	257.50	150.86	693.11	1188.04
50k/s	**Could not keep up with the load**				

UNIFORM DISTRIBUTION (MEMORY-INTENSIVE) - WRITES WORKLOAD - ONLY WRITES

Scylla 4.4.3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	0.61	0.59	0.7	0.83
40k/s	0.72	0.6	0.74	2.65
60k/s	0.82	0.66	0.87	2.74
80k/s	0.9	0.72	1.04	2.93
100k/s	1.31	0.85	2.6	3.26
120k/s	1.46	0.97	2.67	3.88
140k/s	2.21	2.42	3.06	5.37
160k/s	2.07	2.29	2.95	5.16
180k/s	2.33	2.41	3.13	6.06
200k/s	4.81	2.89	4.18	7.96
220k/s	**Could not keep up with the load**			

Cassandra 4

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	0.87	0.84	1	1.25
40k/s	1.24	0.99	1.38	6.93
60k/s	**Could not keep up with the load**			

Cassandra 3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	7.12	0.86	1.07	244.97
40k/s	56.24	1.29	259.26	546.83
60k/s	**Could not keep up with the load**			

UNIFORM DISTRIBUTION (MEMORY-INTENSIVE) - WRITES WORKLOAD - ONLY WRITES

Scylla 4.4.3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	0.62	0.6	0.71	0.84
40k/s	0.63	0.59	0.7	0.85
60k/s	0.68	0.63	0.78	1
80k/s	0.74	0.68	0.87	1.26
100k/s	0.84	0.74	0.97	1.96
120k/s	0.92	0.81	1.08	2.5
140k/s	1.02	0.89	1.23	3.15
160k/s	1.14	0.98	1.4	3.7
180k/s	1.29	1.11	1.61	4.51
200k/s	1.48	1.26	1.87	5.31
220k/s	1.71	1.48	2.2	6.14
240k/s	2.16	1.71	2.58	7.19
260k/s	7.55	2.06	3.98	188.09
280k/s	69.8	4.19	206.57	600.31
300k/s	272.79	183.5	669.52	1063.78
320k/s	**Could not keep up with the load**			

Cassandra 4

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	0.91	0.86	1.01	1.58
40k/s	1.35	1.05	1.57	7.2
60k/s	4.01	1.81	3.88	33.31
80k/s	18.95	5.99	18.06	463.99
100k/s	**Could not keep up with the load**			

Cassandra 3

Load	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	2.48	1.02	1.25	35.75
40k/s	8.93	1.65	3.48	191.89
60k/s	58.11	9.77	198.71	365.43
80k/s	**Could not keep up with the load**			

UNIFORM DISTRIBUTION - MIXED WORKLOAD - 50% READS AND 50% WRITES

Scylla 4.4.3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	Write	0.61	0.59	0.7	0.85
	Read	0.66	0.64	0.76	0.94
40k/s	Write	0.66	0.6	0.74	2.5
	Read	0.73	0.65	0.83	2.61
60k/s	Write	0.75	0.66	0.88	2.61
	Read	0.83	0.73	0.99	2.7
80k/s	Write	0.91	0.75	1.08	2.75
	Read	1	0.83	1.22	3.08
100k/s	Write	1.31	0.93	2.58	3.55
	Read	1.43	1.05	2.72	3.72
120k/s	Write	1.95	1.04	2.34	4.02
	Read	2.07	1.17	2.43	4.24
140k/s	Write	1.92	1.59	3.08	5.36
	Read	2.07	1.79	3.19	5.53
160k/s	Write	2.11	1.83	3.21	5.85
	Read	2.29	2.05	3.4	6.18
180k/s	Write	86.03	2.93	416.02	866.65
	Read	87.91	3.07	427.03	889.72
200k/s	**Could not keep up with the load**				

Cassandra 4

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	Write	0.85	0.81	0.96	1.25
	Read	1	0.94	1.13	2.02
40k/s	Write	1.79	0.96	1.41	9.05
	Read	2.16	1.2	1.98	11.94
60k/s	**Could not keep up with the load**				

Cassandra 3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
20k/s	Write	8.46	0.86	1.06	287.57
	Read	8.6	1.04	1.36	284.16
40k/s	Write	88.29	1.8	344.98	772.8
	Read	89.49	2.57	347.34	776.47
60k/s	**Could not keep up with the load**				

DETAILED RESULTS - "4 VS 40" BENCHMARK

MIXED WORKLOAD - 50% READS AND 50% WRITES

Scylla 4.4.3

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
50k/s	Write	1	0.96	1.2	1.77
	Read	1.7	1.63	2.04	2.93
100k/s	Write	1.06	0.96	1.47	2.34
	Read	1.96	1.76	2.82	4.34
150k/s	Write	0.8	0.72	1.02	2.17
	Read	1.46	1.29	1.98	3.69
200k/s	Write	0.8	0.68	0.99	2.23
	Read	1.46	1.16	2.88	4.23
250k/s	Write	0.99	0.82	1.85	2.72
	Read	1.81	1.39	3.31	4.84
300k/s	Write	1.15	0.97	1.98	3.12
	Read	2.05	1.62	3.55	5.1
350k/s	Write	1.41	1.19	2.27	3.79
	Read	2.45	2	3.94	5.73
400k/s	Write	1.79	1.52	2.71	4.43
	Read	2.94	2.55	4.47	6.45
450k/s	Write	2.08	1.87	3.07	5.14
	Read	3.39	3.13	5.19	7.56
500k/s	Write	2.53	2.39	3.68	6.09
	Read	4.25	4.03	6.33	9.64

550k/s	Write	2.83	2.62	4.12	6.94
	Read	4.82	4.48	7.27	11.71
600k/s	Write	3.08	2.72	4.49	8.28
	Read	5.41	4.69	8.24	15.64
650k/s	**Could not keep up with the load**				

Cassandra 4

Load	Operation	Mean latency (ms)	Median latency (ms)	90% latency (ms)	99% latency (ms)
50k/s	Write	0.92	0.9	1.07	1.24
	Read	1.41	1.37	1.64	2.83
100k/s	Write	0.86	0.83	0.97	1.14
	Read	1.35	1.29	1.56	2.93
150k/s	Write	0.87	0.82	0.95	1.18
	Read	1.38	1.3	1.6	3.27
200k/s	Write	1.93	0.86	1	1.57
	Read	2.47	1.36	1.73	3.78
250k/s	Write	0.96	0.9	1.08	2.7
	Read	1.59	1.44	1.96	4.6
300k/s	Write	1.07	0.96	1.21	3.59
	Read	1.84	1.57	2.39	6.42
350k/s	Write	1.22	1.03	1.42	4.33
	Read	2.23	1.78	3.22	9.16
400k/s	Write	1.43	1.16	1.86	5.37
	Read	2.78	2.11	4.58	12.43
450k/s	Write	1.96	1.36	2.64	7.93
	Read	3.74	2.59	6.19	18.73
500k/s	Write	3.33	1.67	3.94	19.09
	Read	5.78	3.27	8.88	35.88
550k/s	Write	7.49	2.35	7.29	166.46
	Read	11.24	4.65	16.42	171.57
600k/s	Write	96.88	8.17	256.51	1131.41
	Read	102.16	15.38	262.67	1136.66
650k/s	**Could not keep up with the load**				

LATENCY / THROUGHPUT - 3 X I3.4XLARGE

Benchmark	Scylla 4.4	Cassandra 4.0
Data: 1TB per node **Workload: 50% W/R** **Distribution: Uniform** **Low cache hit ratio**	Max throughput: 90k/s p99th latency < 10ms: 60k/s	Max throughput: 40k/s p99th latency < 10ms: 20k/s
Data: 1TB per node **Workload: 100% W** **Distribution: Uniform** **Disk-intensive workload**	Max throughput: 180k/s p99th latency < 10ms: 170k/s	Max throughput: 50k/s p99th latency < 10ms: 30k/s
Data: 1TB per node **Workload: 100% R** **Distribution: Uniform** **Low cache hit ratio**	Max throughput: 80k/s p99th latency < 10ms: 60k/s	Max throughput: 40k/s p99th latency < 10ms: 20k/s
Data: 1TB per node **Workload: 50% W/R** **Distribution: Uniform** **High cache hit ratio**	Max throughput: 180k/s p99th latency < 10ms: 160k/s	Max throughput: 40k/s p99th latency < 10ms: 20k/s
Data: 1TB per node **Workload: 100% W** **Distribution: Uniform** **Memory-intensive workload**	Max throughput: 200k/ p99th latency < 10ms: 200k/s	Max throughput: 40k/s p99th latency < 10ms: 40k/s
Data: 1TB per node **Workload: 100% R** **Distribution: Uniform** **High cache hit ratio**	Max throughput: 300k/s p99th latency < 10ms: 240k/s	Max throughput: 80k/s p99th latency < 10ms: 40k/s
Data: 1TB per node **Workload: 50% W/R** **Distribution: Gaussian** **Medium cache hit ratio**	Max throughput: 80k/s p99th latency < 10ms: 70k/s	Max throughput: 40k/s p99th latency < 10ms: 10k/s

Note: throughput numbers are reported with 10k/s granularity (in some cases 20k/s).

Administrative operations - 3 x i3.4xlarge		
Benchmark	**Scylla 4.4**	**Cassandra 4.0**
Add single node **Data: 1TB per node**	36 minutes 56 seconds	107 minutes 1 second
	Adding a node on Scylla is about 3 times faster	
Double cluster size **Data: 1TB per node**	94 minutes 57 seconds	238 minutes 21 seconds
	Doubling cluster size on Scylla is about 2.5 times faster	
Replace a node **Data: 1TB per node**	54 minutes 19 seconds	208 minutes 46 seconds
	Replacing node on Scylla is about 4 times faster	

4 vs 40		
Benchmark	**Scylla 4.4 (4x i3.metal)**	**Cassandra 4.0 (40x i3.4xlarge)**
Yearly cost **AWS us-east-2 region**	On-demand: $174,919.68 1 year reserved instances, all-upfront: $111,424.00	On-demand: $437,299.20 1 year reserved instances, all-upfront: $278,560.00
Latency / throughput **Data: 40TB altogether** **Workload: 50% W/R** **Distribution: Uniform** **Low cache hit ratio**	Max throughput: 600k/s p99th latency < 10ms: 350k/s	Max throughput: 600k/s p99th latency < 10ms: 350k/s
25% capacity increase **Data: 40TB altogether**	1 hour 29 minutes 29 seconds	16 hours 54 minutes 52 seconds
	Increasing capacity is about 11 times faster on Scylla	

ABOUT SCYLLADB

Scylla is the real-time big data database. API-compatible with Apache Cassandra and Amazon DynamoDB, Scylla embraces a shared-nothing approach that increases throughput and storage capacity as much as 10X. Comcast, Discord, Disney+ Hotstar, Grab, Medium, Starbucks, Ola Cabs, Samsung, IBM, Investing.com and many more leading companies have adopted Scylla to realize order-of-magnitude performance improvements and reduce hardware costs. Scylla's database is available as an open source project, an enterprise edition and a fully managed database as a service. ScyllaDB was founded by the team responsible for the KVM hypervisor. For more information: ScyllaDB.com

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