Scylla Open Source 3.0

Scylla is an open source NoSQL database that offers the horizontal scale-out and fault-tolerance of Apache Cassandra, but delivers 10X the throughput and consistent, low single-digit latencies. Implemented from scratch in C++, Scylla's close-to-the-hardware design significantly reduces the number of database nodes you require and self-optimizes to dynamic workloads and various hardware combinations.

CONTENTS

INTRODUCTION

With the release of Scylla Open Source 3.0, we've introduced a rich set of new features for more efficient querying, reduced storage requirements, lower repair times, and better overall database performance. Already the industry's most performant NoSQL database, Scylla now includes production-ready features that surpass the capabilities of Apache Cassandra.

Scylla Open Source 3.0 is now available for download

MATERIALIZED VIEWS

Material Views automate the tedious and inefficient chores created when an application maintains several tables with the same data organized differently. Data is divided into partitions that can be found by a partition key. Sometimes the application needs to find a partition or partitions by the value of another column. Doing this efficiently without scanning all of the partitions requires indexing.

People have been using Materialized Views, also calling them denormalization, for years as a client-side implementation. In those days, the application maintained two or more views and two or more separate tables with the same data but under a different partition key. Every time the application wanted to write data, it needed to write to both tables, and reads were done directly (and efficiently) from the desired table. However, ensuring any level of consistency between the data in the two or more views required complex and slow application logic.

Scylla's Materialized Views feature moves this complexity out of the application and into the servers. The implementation is faster (fewer round trips to the applications) and more reliable. This approach makes it much easier for applications to begin using multiple views into their data. The application just declares the additional views, Scylla creates the new view tables, and on every update to the base table the view tables are automatically updated as well. Writes are executed only on the base table directly and are automatically propagated to the view tables. Reads go directly to the view tables.

As usual, the Scylla version is compatible – in features and CQL syntax – with the Apache Cassandra version (where it is still in experimental mode).

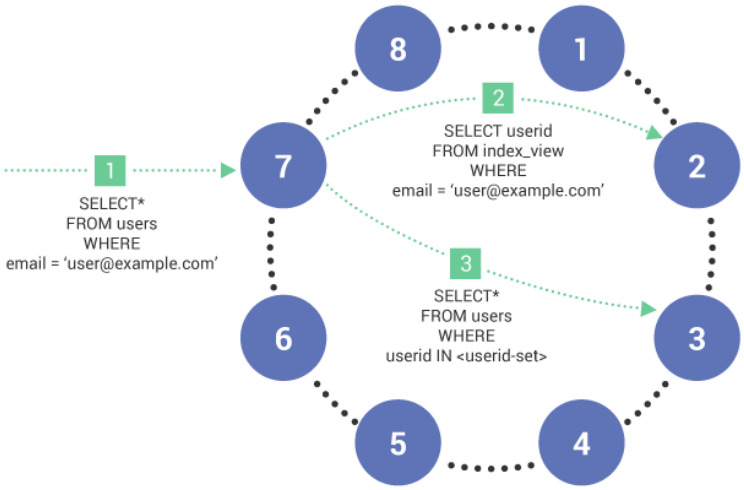
```
CREATE TABLE buildings (
     name text,
     city text,
     built int,
     Feet int,
     PRIMARY KEY (name)
);

CREATE MATERIALIZED VIEW building_by_city AS
     SELECT * FROM buildings
     WHERE city IS NOT NULL
     PRIMARY KEY (city, name);

SELECT * from building_by_city;
SELECT name, built, feet from building_by_city LIMIT 1;
```

GLOBAL SECONDARY INDEXES

Scylla Open Source 3.0 introduces production-ready global secondary indexes that can scale to any size distributed cluster — unlike the local-indexing approach adopted by Apache Cassandra. The secondary index uses a Materialized View index under the hood in order to make the index independent from the amount of nodes in the cluster. Secondary Indexes are (mostly) transparent to the application. Queries have access to all the columns in the table and you can add and remove indexes without changing the application. Secondary Indexes can also have less storage overhead than Materialized Views because Secondary Indexes need to duplicate only the indexed column and primary key, not the queried columns like with a Materialized View. For the same reason, updates can be more efficient with Secondary Indexes because only changes to the primary key and indexed column cause an update in the index view. In the case of a Materialized View, an update to any of the columns that appear in the view requires the backing view to be updated.



As always, the decision whether to use Secondary Indexes or Materialized Views really depends on the requirements of your application. If you need maximum performance and are likely to query a specific set of columns, you should use Materialized Views. However, if the application needs to query different sets of columns, Secondary Indexes are a better choice because they can be added and removed with less storage overhead depending on application needs.

Global secondary indexes minimize the amount of data retrieved from the database, providing many benefits:

- Results are paged and customizable
- Filtering is supported to narrow result sets
- Keys, rather than data, are denormalized Supports more general-purpose use cases than Materialized Views

ALLOW FILTERING

Allow filtering is a way to make a more complex query, returning only a subset of matching results. Because the filtering is done on the server, this feature also reduces the amount of data transferred over the network between the cluster and the application. Such filtering may incur processing impacts to the Scylla cluster. For example, a query might require the database to filter an extremely large data set before returning a response. By default, such queries are prevented from execution, returning the following message:

```
Bad Request: Cannot execute this query as it might involve data filtering and thus may have
unpredictable performance.
```

Unpermitted queries include those that restrict:

- Non-partition key fields
- Parts of primary keys that are not prefixes
- Partition keys with something other than an equality relation (though you can combine SI with ALLOW FILTERING to support inequalities; >= or <=; see below)
- Clustering keys with a range restriction and then by other conditions (see this blog)

However, in some cases (usually due to data modeling decisions), applications need to make queries that violate these basic rules. Starting with Scylla Open Source 3.0, queries can be appended with the ALLOW FILTERING keyword to bypass this restriction and utilize server-side filtering.

The benefits of filtering include:

- Cassandra query compatibility
- Spark-Cassandra connector query compatibility
- Query flexibility against legacy data sets

NEW FILE FORMAT

Scylla Open Source 3.0 introduces support for a more performant storage format (SSTable), which is not only compatible with Apache Cassandra 3.x but also reduces storage volume by as much as 3X. The older 2.x format used to duplicate the column name next to each cell on disk. The new format eliminates the duplication and the column names are stored once, within the schema.

The newly introduced format is identical to that used by Apache Cassandra 3.x, while remaining backward-compatible with prior Scylla SSTable formats. New deployments of Scylla Open Source 3.0 will automatically use the new format, while existing files remain unchanged.

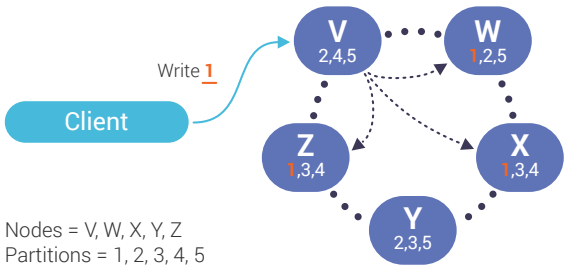
This new storage format delivers important benefits, including:

- Can read existing Apache Cassandra 3.x files when migrating
- Faster than previous versions
- Reduced storage footprint of up to 66%, depending on the data model used Range delete support

HINTED HANDOFF

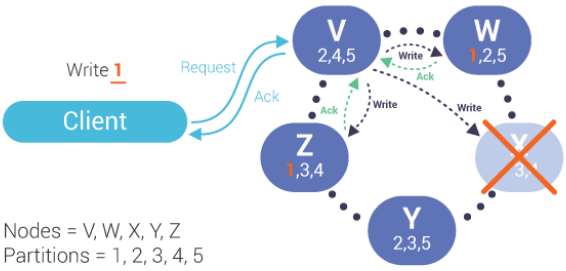
Hinted handoffs are designed to help when any individual node is temporarily unresponsive due to heavy write load, network weather, hardware failure, or any other factor. Hinted handoffs also help in the event of short-term network issues or node restarts, reducing the time for scheduled repairs, and resulting in higher overall performance for distributed deployments. Originally introduced as an experimental feature in Scylla Open Source 2.1, hinted handoffs are another production-ready feature in Scylla Open Source 3.0.

Based on the replication factor (RF), the co-ordinator attempts to write to RF nodes



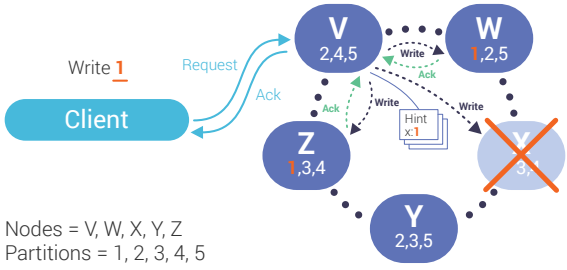
Nodes = V, W, X, Y, Z
Partitions = 1, 2, 3, 4, 5

If one node is down, acknowledgments are only returned from two nodes



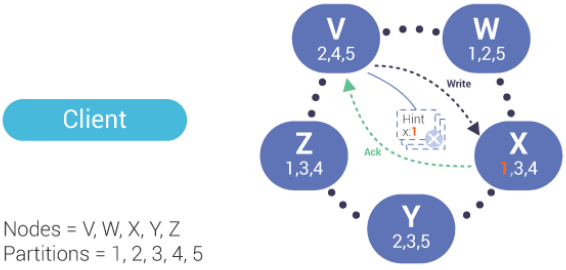
Nodes = V, W, X, Y, Z
Partitions = 1, 2, 3, 4, 5

The co-ordinator will write and store a hint for the missing node



Nodes = V, W, X, Y, Z
Partitions = 1, 2, 3, 4, 5

Once the down node comes up, the co-ordinator will replay the hint for that node. After the coordinator receives an acknowledgement of the write, the hint is deleted.



Nodes = V, W, X, Y, Z
Partitions = 1, 2, 3, 4, 5

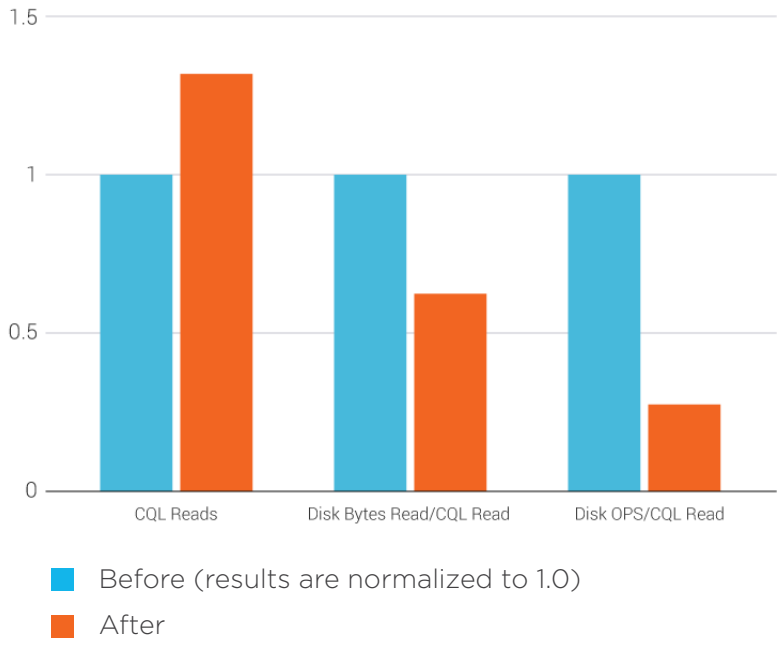
Technically, a 'hint' is a record of a write request held by the coordinator until an unresponsive replica node comes back online. When a write is deemed successful but one or more replica nodes fail to acknowledge it, Scylla will write a hint that is replayed to those nodes when they recover. Once the node becomes available again, the write request data in the hint is written to the replica node.

Hinted handoffs deliver the following benefits:

- Minimizes the difference between data in the nodes when nodes are down — whether for scheduled upgrades or for all-too-common intermittent network issues.
- Reduces the amount of data transferred during repair.
- Reduces the chances of checksum mismatch (during read-repair) and thus improves overall latency.

FULL, MULTI-PARTITION SCAN IMPROVEMENTS

Scylla Open Source 3.0 builds on earlier improvements by extending stateful paging to support range scans as well. As opposed to other partition queries, which read a single partition or a list of distinct partitions, range scans read all of the partitions that fall into the range specified by the client. Since the precise number and identity of partitions in a given range cannot be determined in advance, the query must read data from all nodes containing data for the range.



scylladb.com/2018/11/01/more-efficient-range-scan-paging-with-scylla-3-0/

To improve range scan paging, Scylla Open Source 3.0 introduces a new control algorithm for reading all data belonging to a range from all shards, which caches the intermediate streams on each of the shards and directs paged queries to the matching, previously used, cached results. The new algorithm is essentially a multiplexer that combines the output of readers opened on affected shards into a single stream. The readers are created on-demand when the partition scan attempts to read from the shard. To ensure that the read won't stall, the algorithm uses buffering and read-ahead.

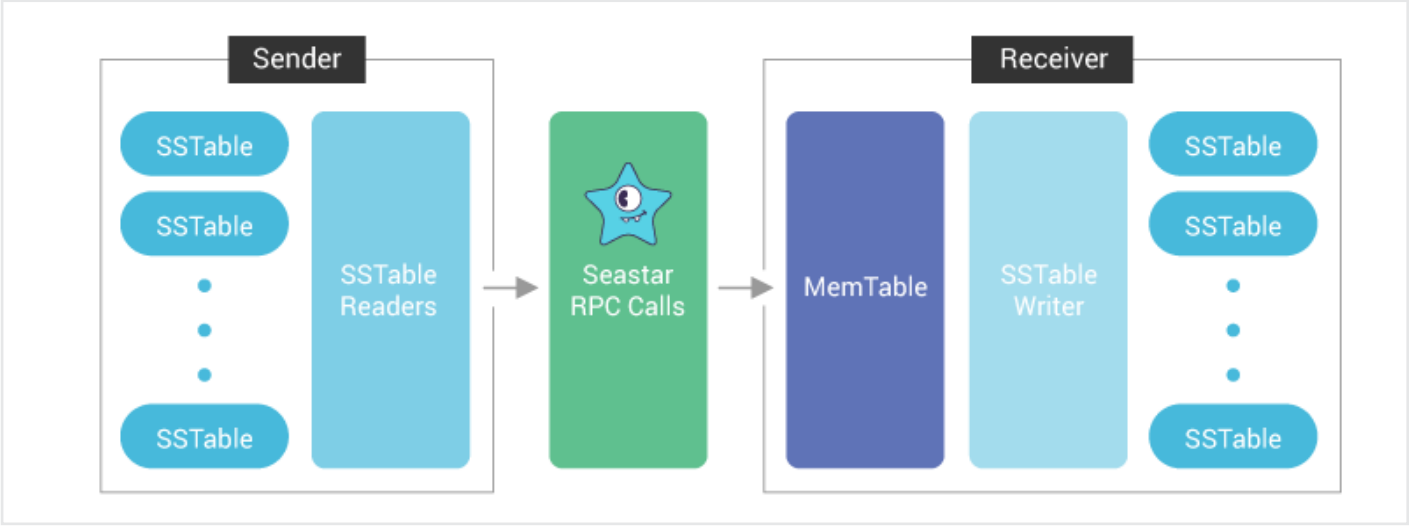
Benefits include:

- Improved system responsiveness
- Throughput of range scans improved by as much as 30%
- Amount of data read from the disk reduced by as much as 40%
- Disk operations lowered by as much as 75%

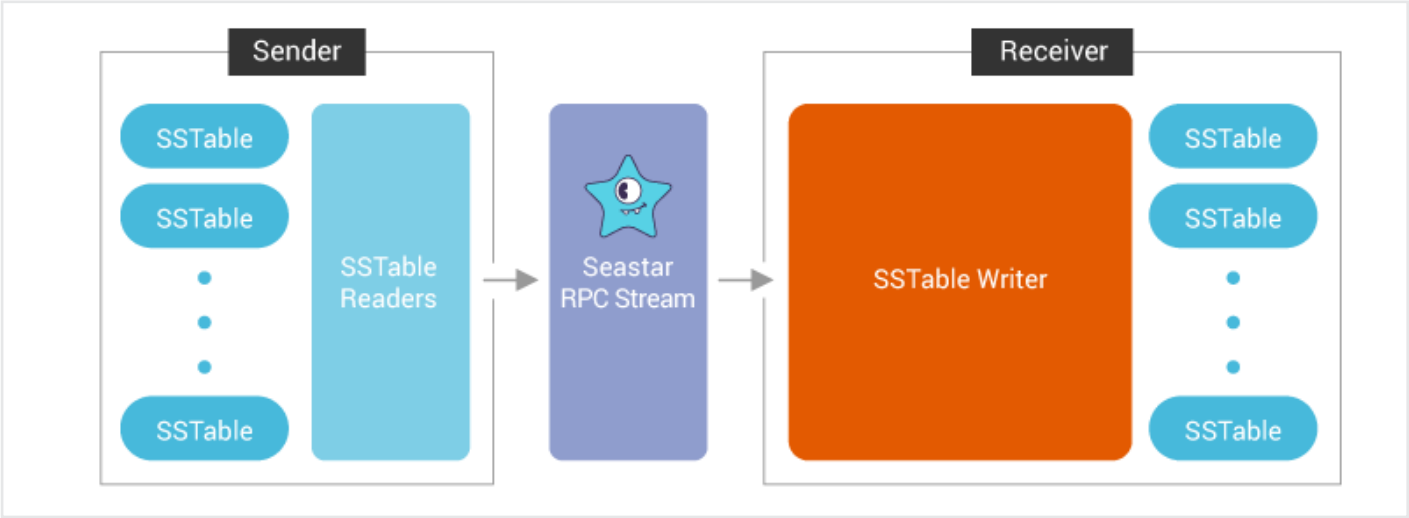
STREAMING IMPROVEMENTS

Streaming is used during node recovery to populate restored nodes with data replicated from running nodes. The Scylla streaming model reads data on one node, transmits it to another node, and then writes to disk. The sender creates SSTable readers to read the rows from SSTables on disk and sends them over the network. The receiver receives the rows from the network and writes them to a memtable. The rows in memtable are flushed into SSTables periodically or when the memtable is full.

Before



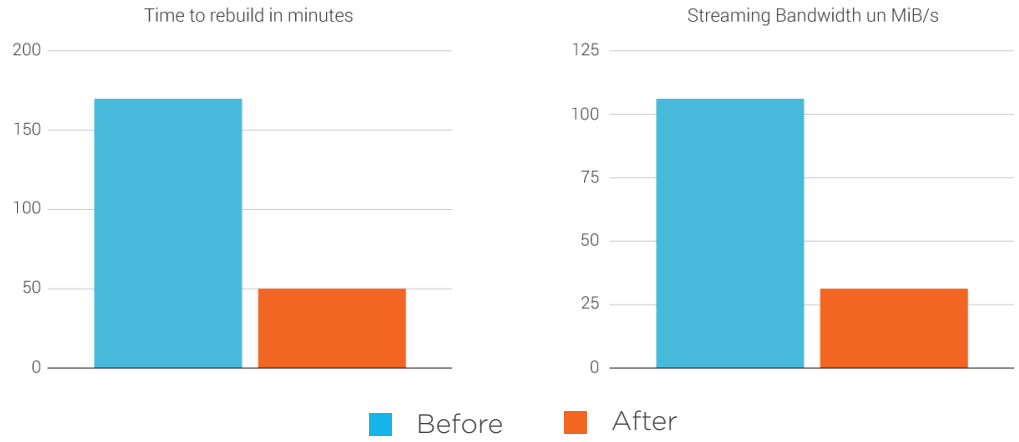
After



In Scylla Open Source 3.0, stream synchronization between nodes bypasses memtables, significantly reducing the time to repair, add and remove nodes. These improvements result in higher performance when there is a change in the cluster topology, improving streaming bandwidth by as much as 240% and reducing the time it takes to perform a "rebuild" operation by 70%.

Scylla's new streaming improvements provide the following benefits:

- Lower memory consumption. The saved memory can be used to handle your CQL workload instead.
- Better CPU utilization. No CPU cycles are used to insert and sort memtables.
- Bigger SSTables and fewer compactions.



RELEASE NOTES

You can read more details about Scylla Open Source 3.0 in the Release Notes.

Download Scylla Open Source 3.0 Now!

ABOUT SCYLLADB

Scylla is the real-time big data database. Fully compatible with Apache Cassandra, Scylla embraces a shared-nothing approach that increases throughput and storage capacity as much as 10X that of Cassandra. Comcast, Ola Cabs, Samsung, AdGear, IBM Compose, Grab, Snapfish, Intel, MediaMath, AppNexus, CERN, SAS, AppNexus, Samsung, Investing.com, L3 Technologies and many more leading companies have adopted Scylla to realize order-of-magnitude performance improvements and reduce hardware costs. ScyllaDB was founded by the team responsible for the KVM hypervisor and is backed by Bessemer Venture Partners, Innovation Endeavors, Wing Venture Capital, Qualcomm Ventures, TLV Partners, Magma Venture Partners, Western Digital Capital and Samsung Ventures.
For more information: ScyllaDB.com

SCYLLADB.COM

SCYLLA.

United States Headquarters
1900 Embarcadero Rd
Palo Alto, CA 94303 U.S.A.
Phone: +1 (650) 332-9582
Email: info@scylladb.com

Israel Headquarters
11 Galgalei Haplada
Herzelia, Israel