ROCKSANDRA UPDATE

Pengchao Wang @ Instagram

AGENDA

1 Pluggable Storage Engine and Rocksandra

2 Highlights of 2018~2019 improvements

3 Learnings from production

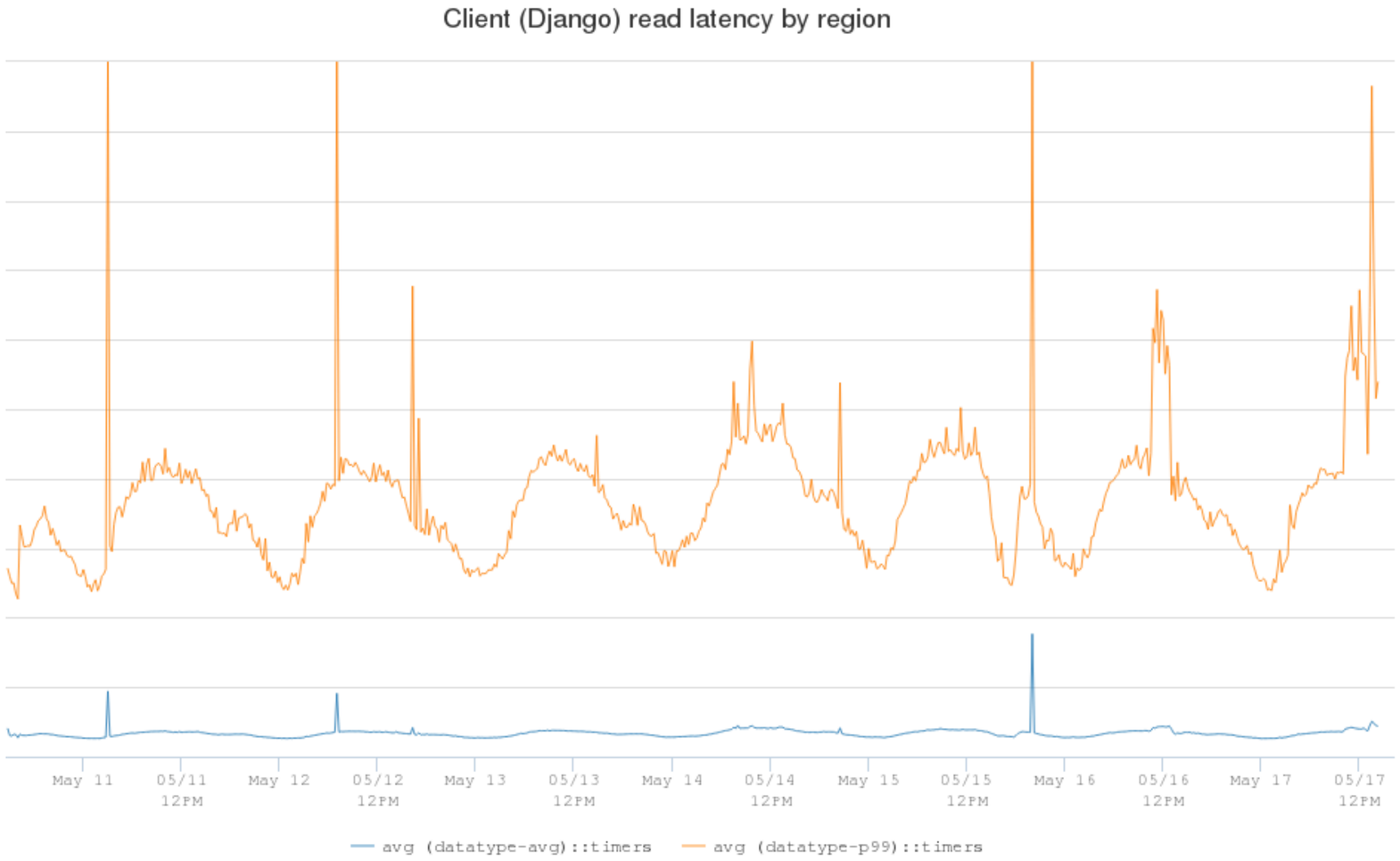
ROCKSANDRA

MOTIVATIONS

60ms

25ms

5ms



ROCKSANDRA

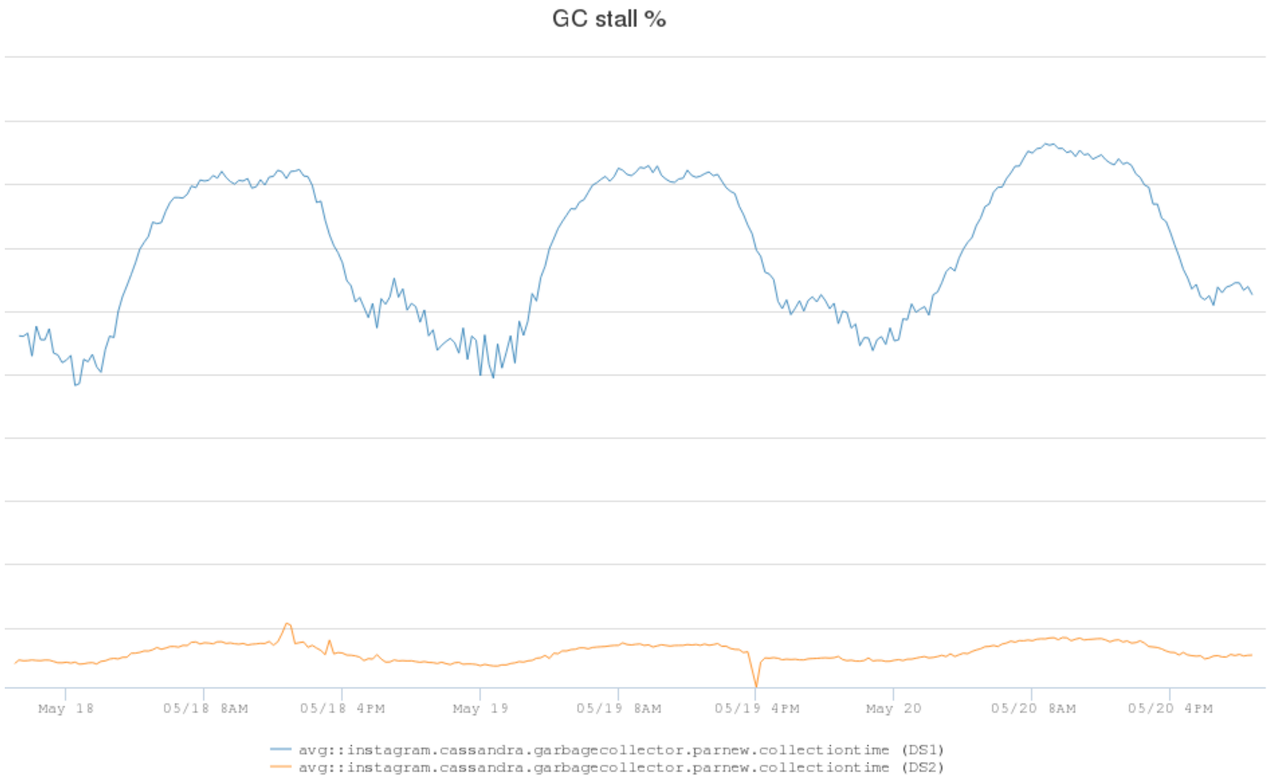
- Rocksandra is knew storage engine implemented on top of RocksDB for
  - Less JVM GC
  - Low tail latency
  - High throughput

ROCKSANDRA

0.9%

0.5%

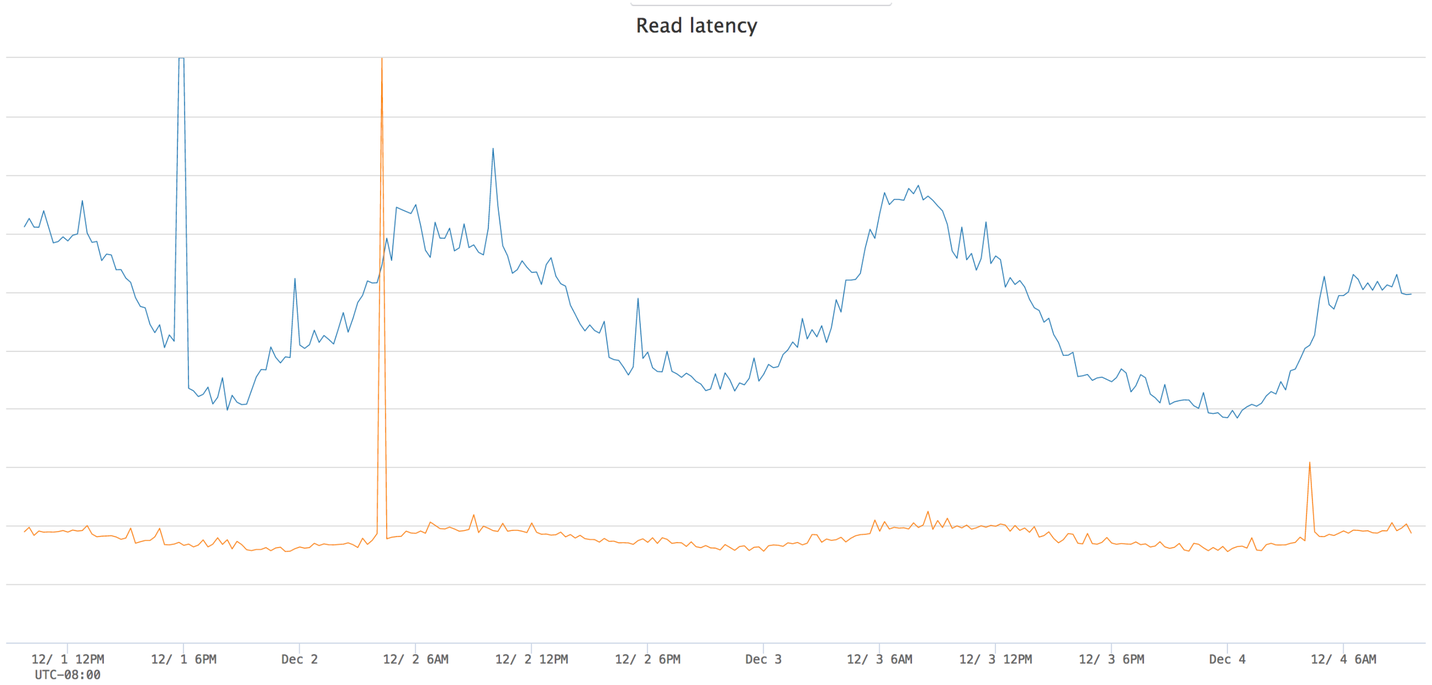
0.1%



P99 LATENCY

60ms

20ms



CURRENT STATUS IN INSTAGRAM

- 70% C* QPS is on Rocksandra (10s of millions)
- 100% on CPU and IO bound clusters
- Disk bound cluster is under migration

FEATURE PARITY

Features supported:

- Most of non-nested data types
- Table schema
- Point query
- Range query
- Mutations
- Timestamp
- TTL
- Deletions/Cell tombstones
- Streaming
- **Multi-partition query**
- **Snapshot**
- **Cleanup**
- **Truncate**
- **Partition deteltion ***
- **SSTableloader ***
- **Secondary indexes ***

Features will be supported later

- Nested data types
- Counters
- Range tombstone
- Materialized views
- SASI
- Row level tombstone
- Anti-entropy repair

IMPROVEMENTS HIGHLIGHTS

IMPROVEMENTS HIGHLIGHTS

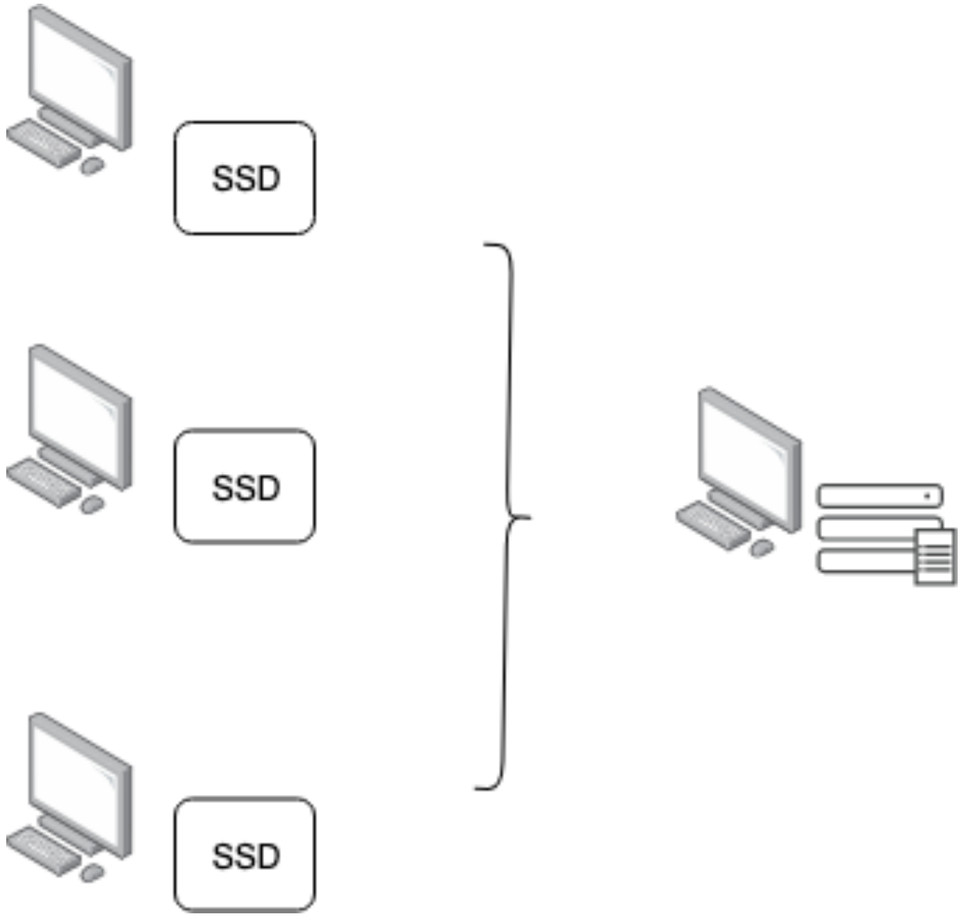
1 High density storage support

2 Fast Streaming

3 Fast Cleanup

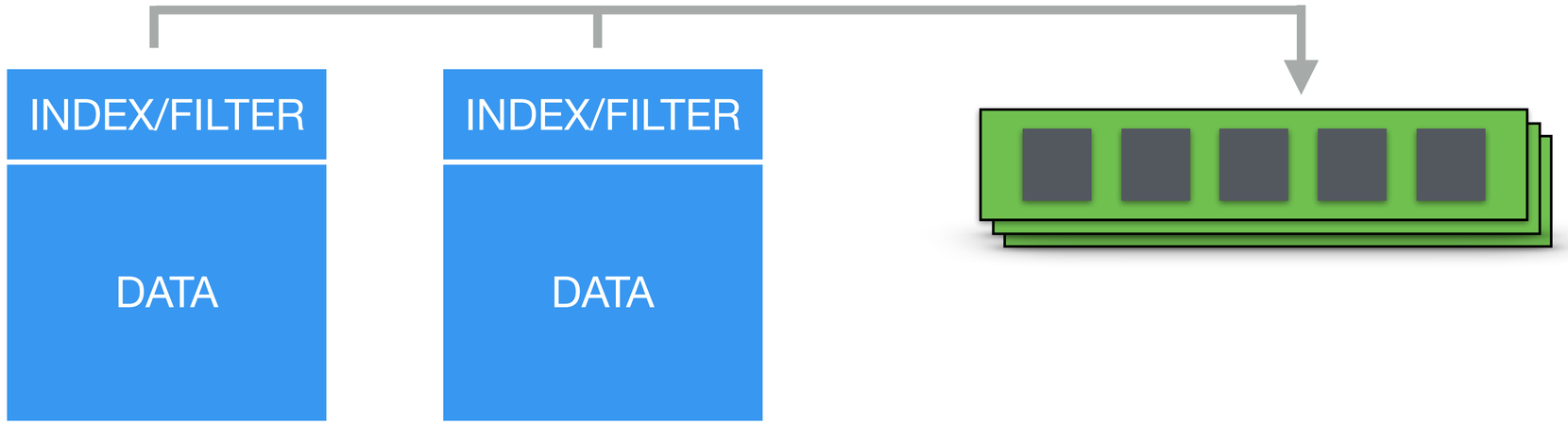
4 Space amplification improvement

HIGH DENSITY STORAGE



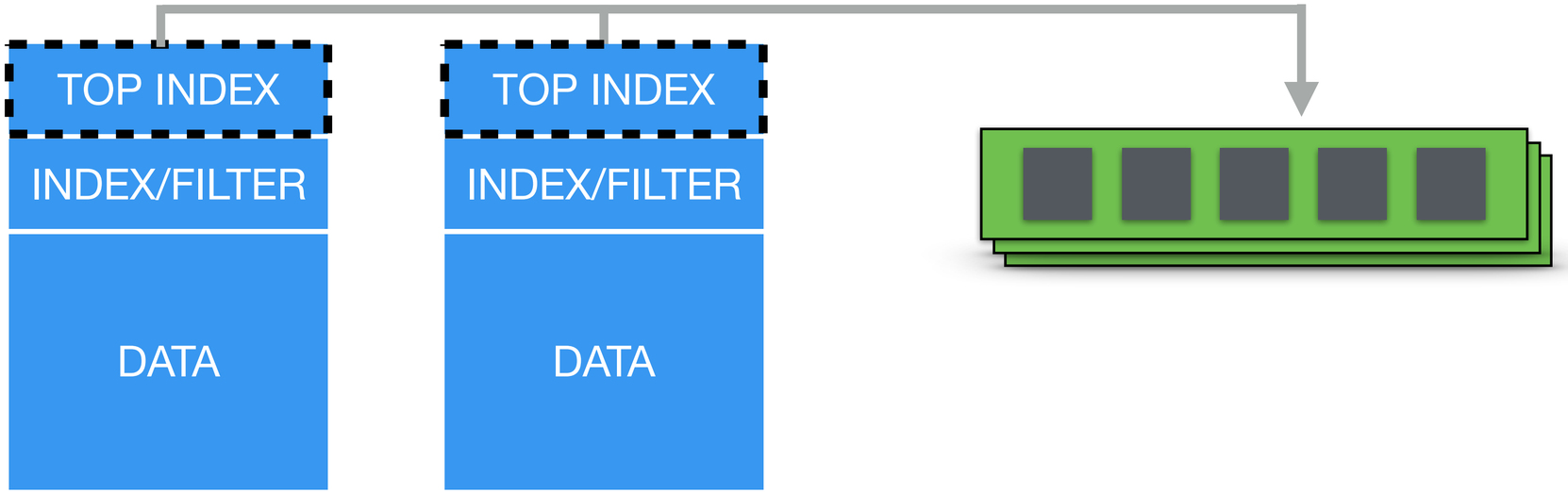
HIGH DENSITY STORAGE

Partitioned Index and Filter

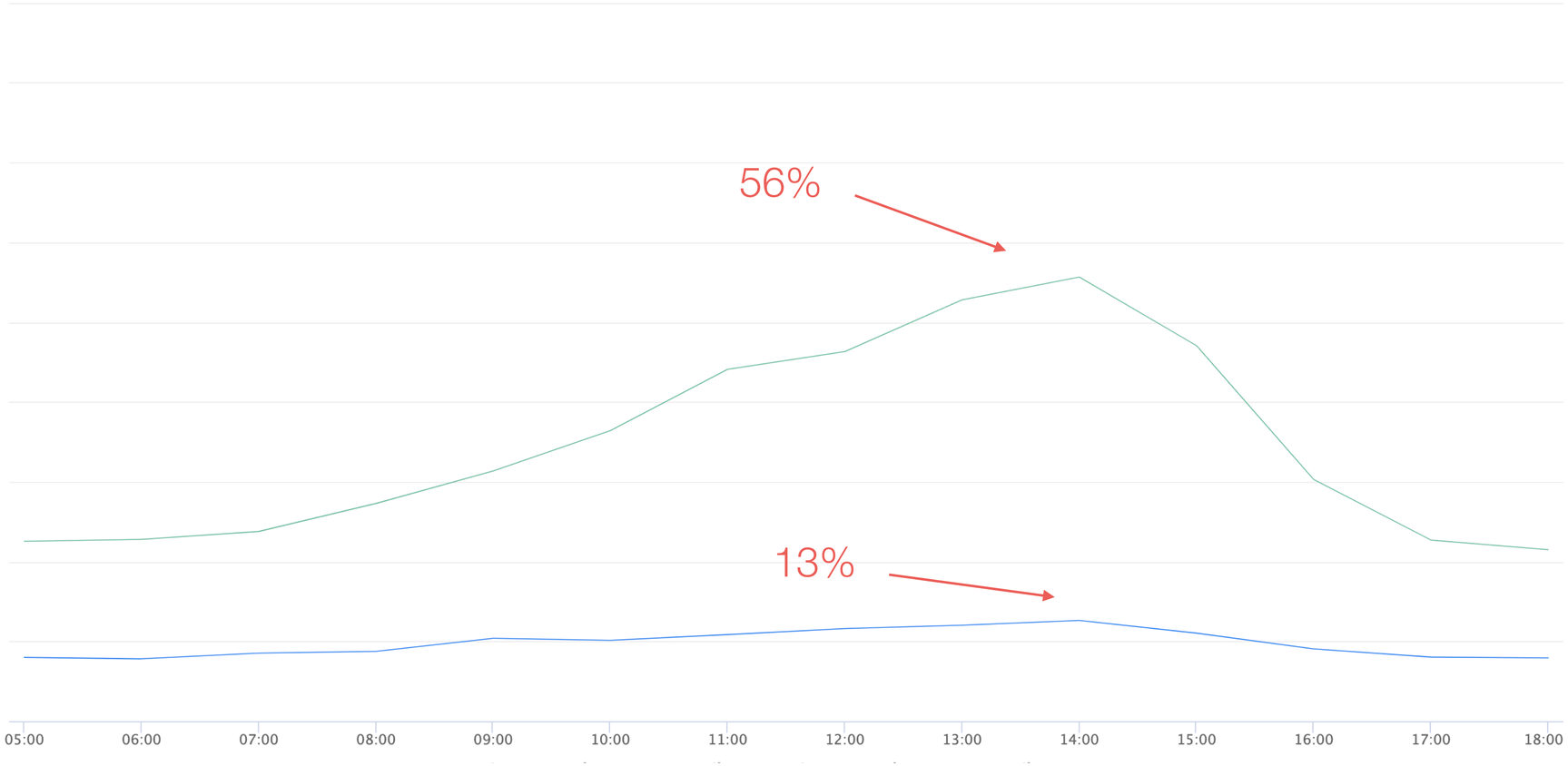


HIGH DENSITY STORAGE

Partitioned Index and Filter

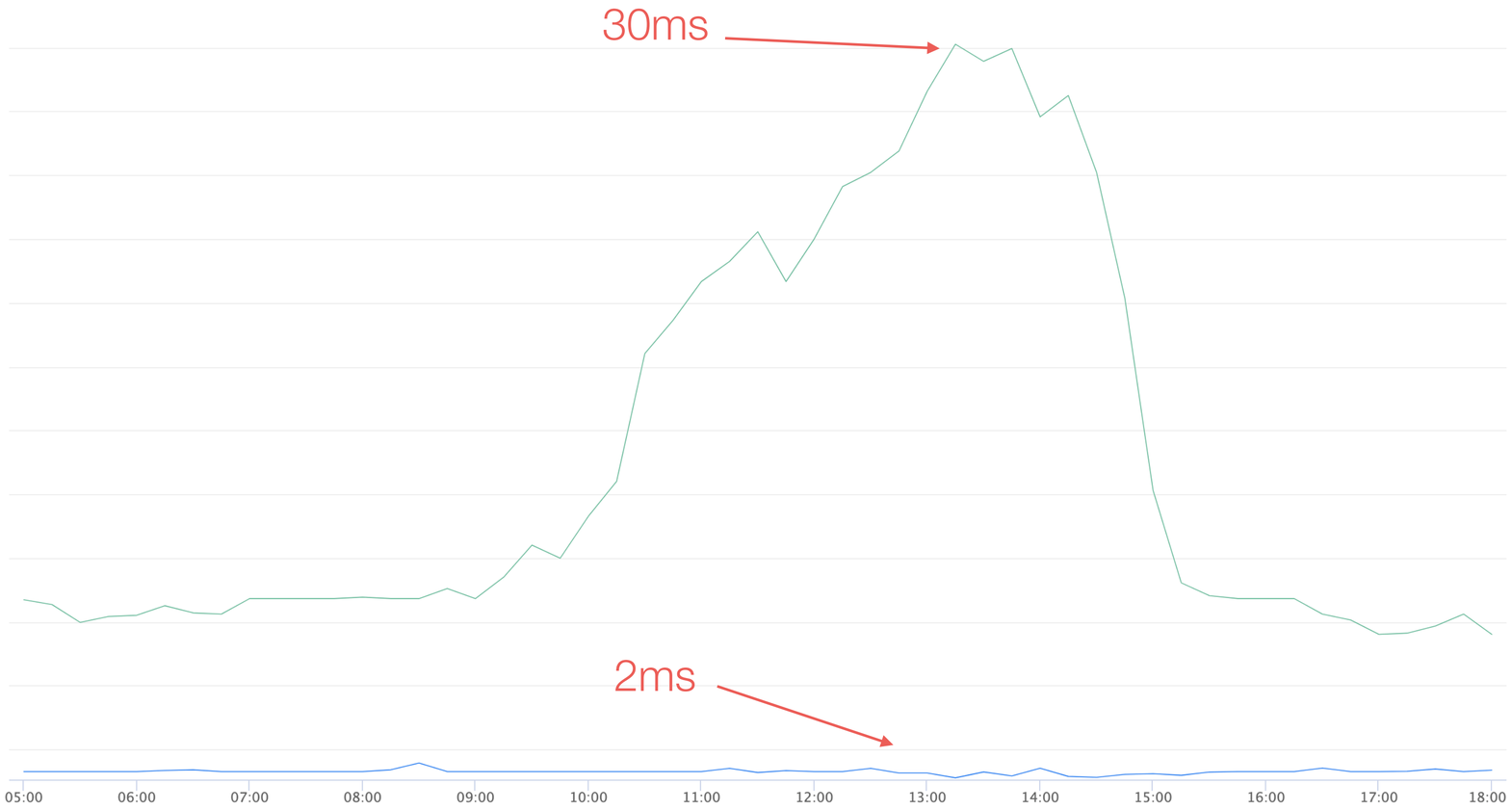


HIGH DENSITY STORAGE



CPU utilization (%)

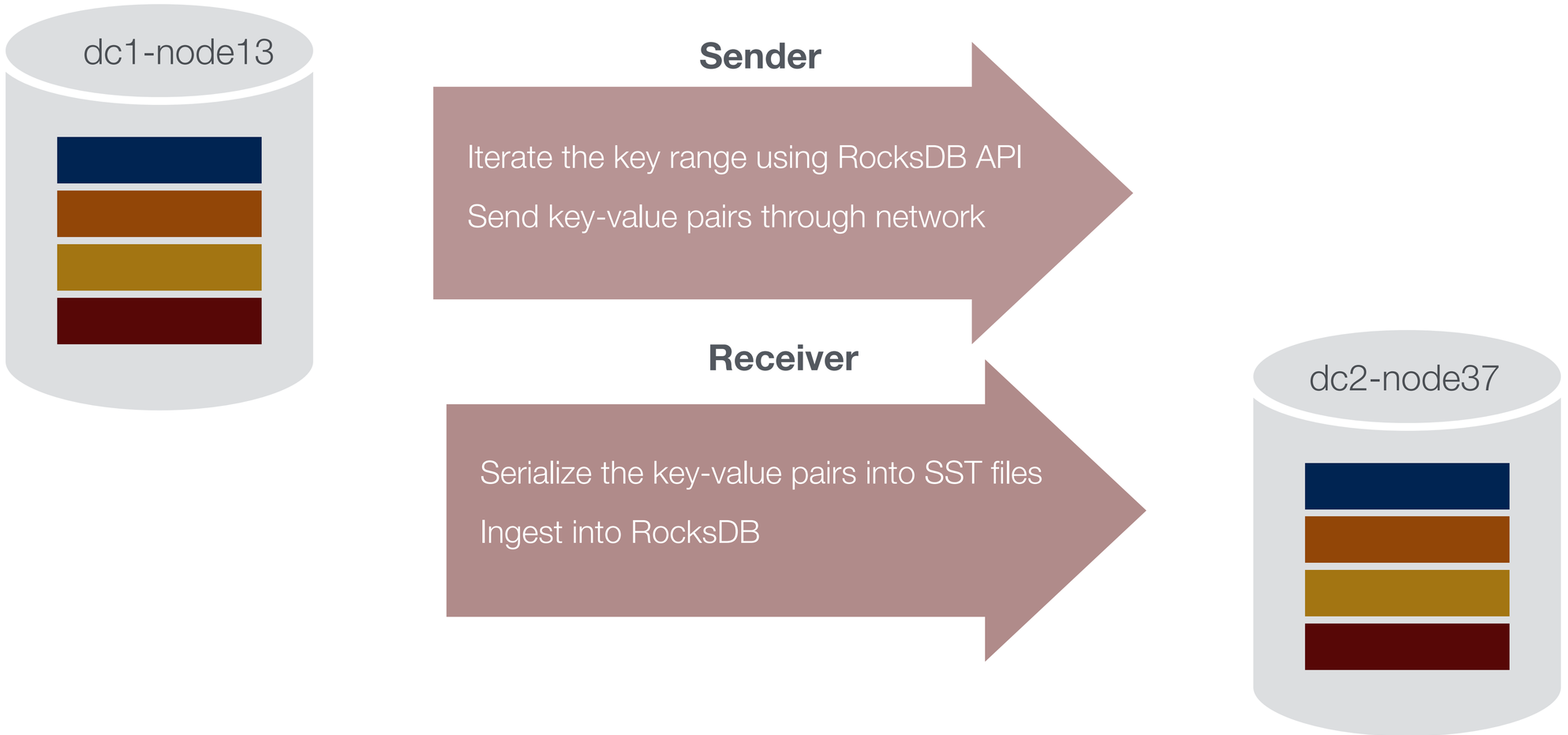
HIGH DENSITY STORAGE



P99 Read Latency (ms)

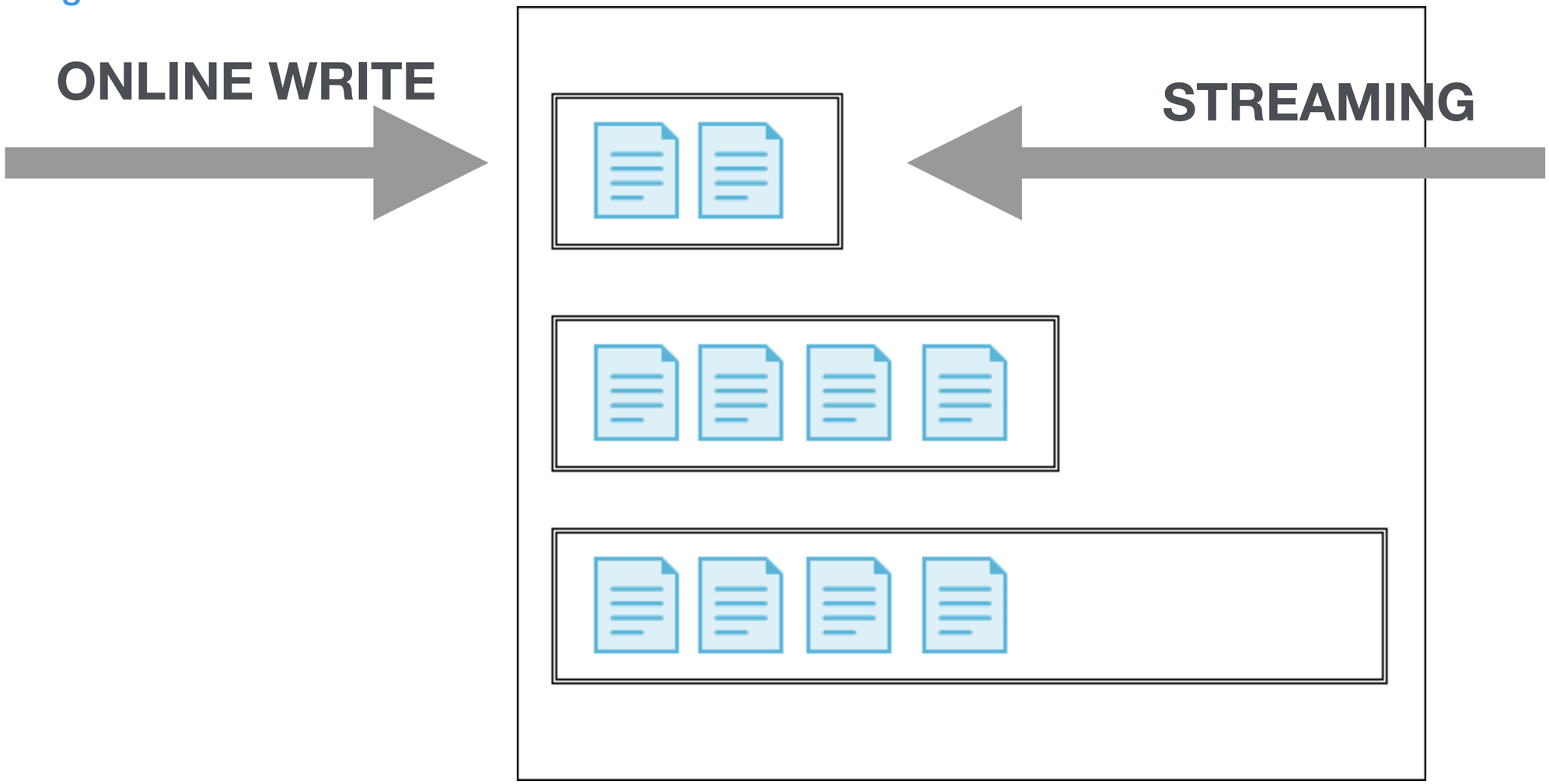
FAST STREAMING

Rocksandra Streaming



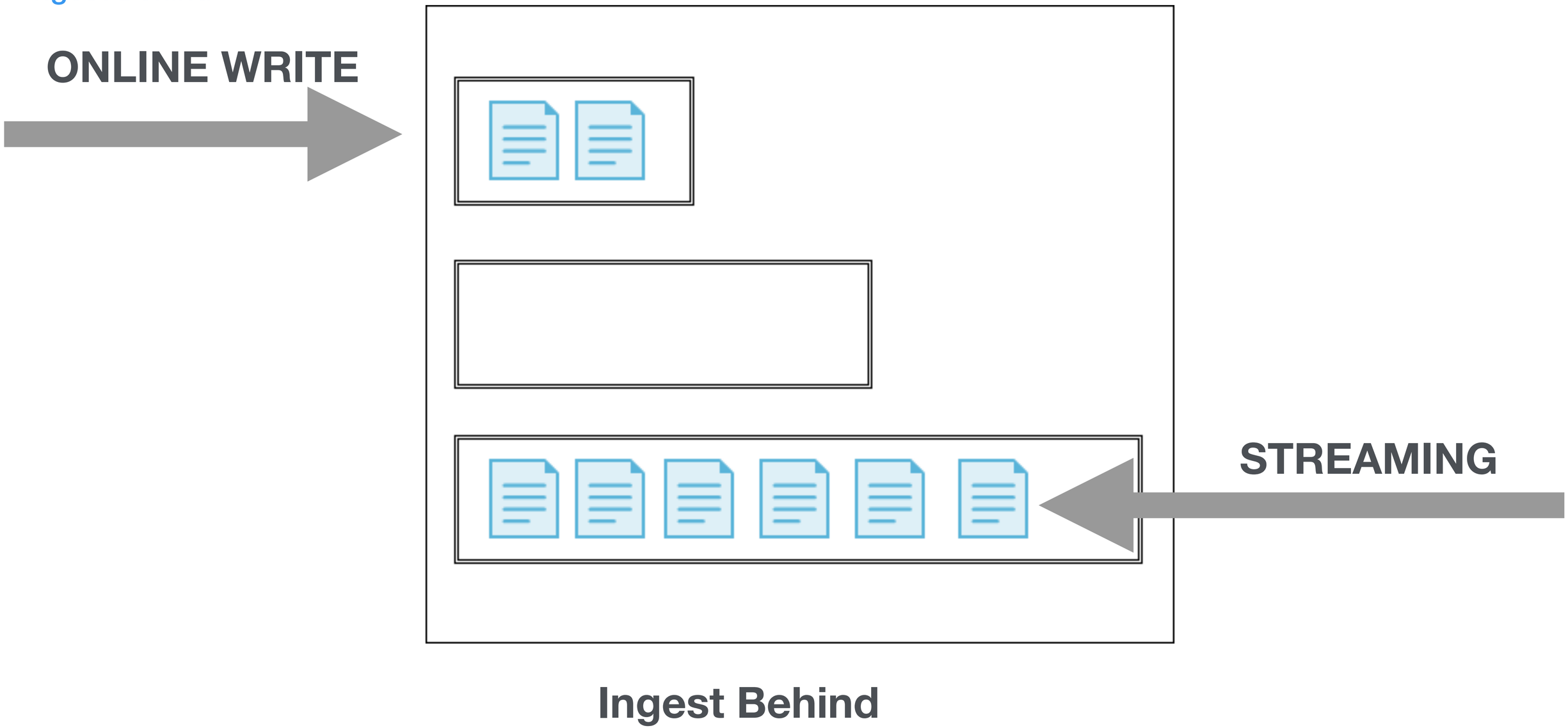
FAST STREAMING

Ingest Behind

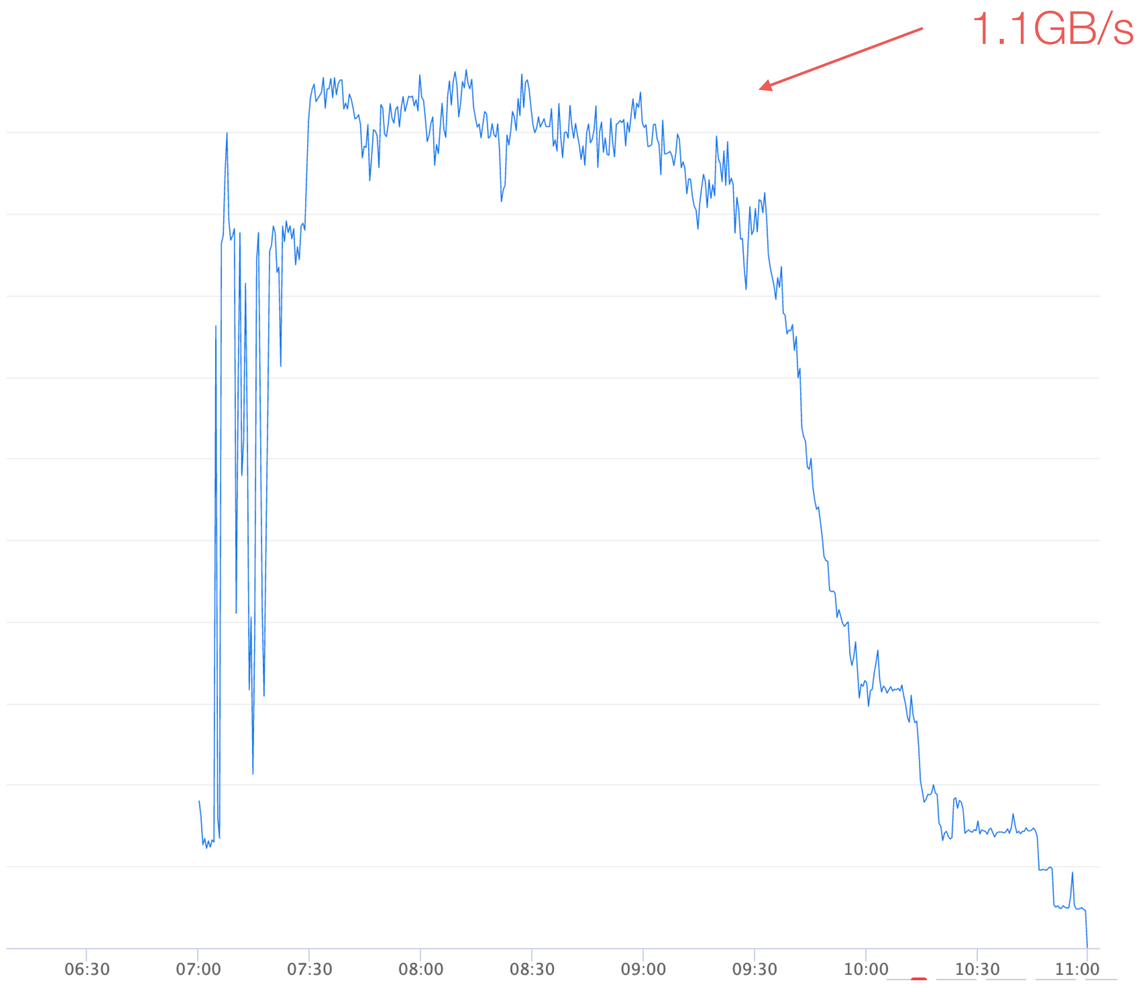


FAST STREAMING

Ingest Behind

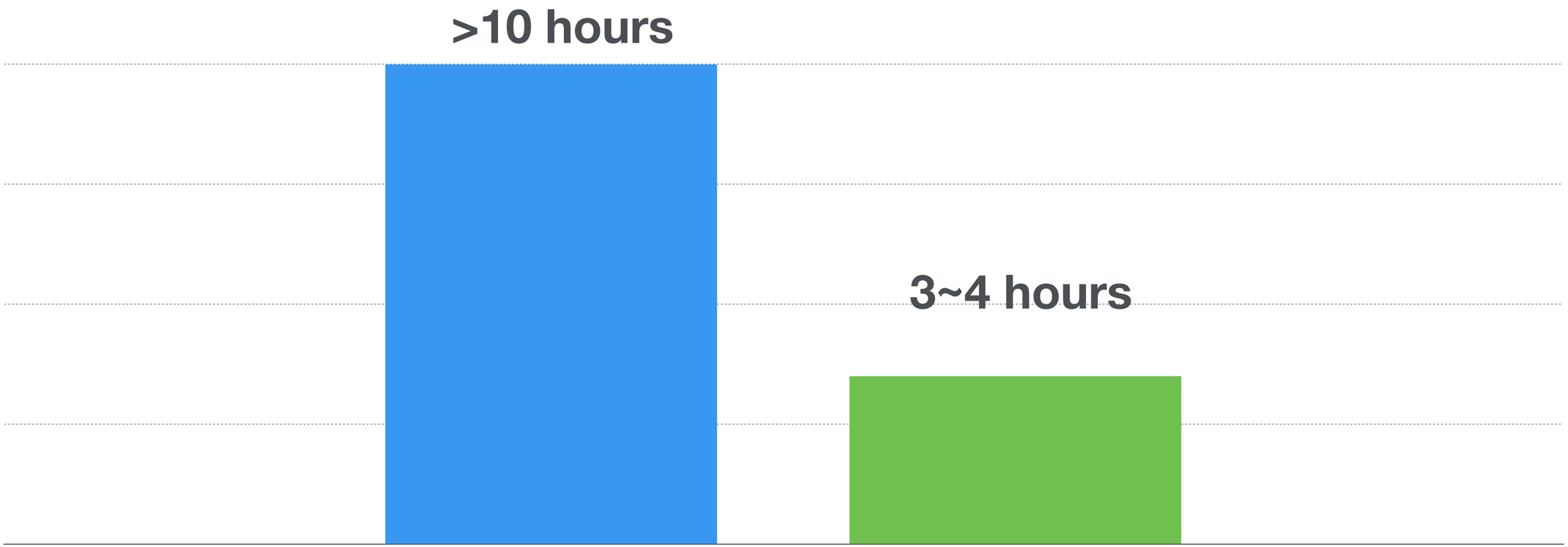


FAST STREAMING

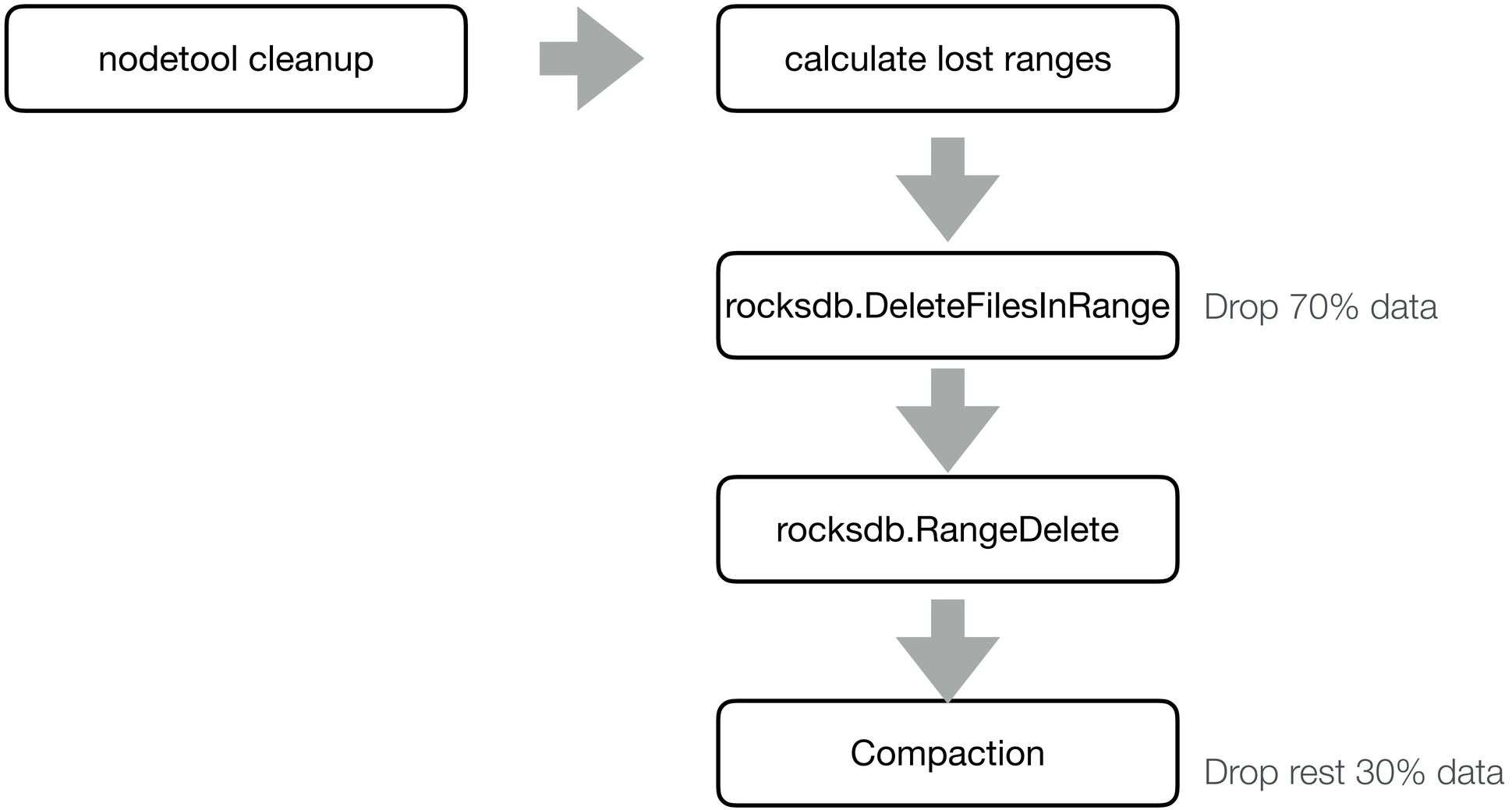


Streaming Incoming Bytes

FAST STREAMING



FAST CLEANUP



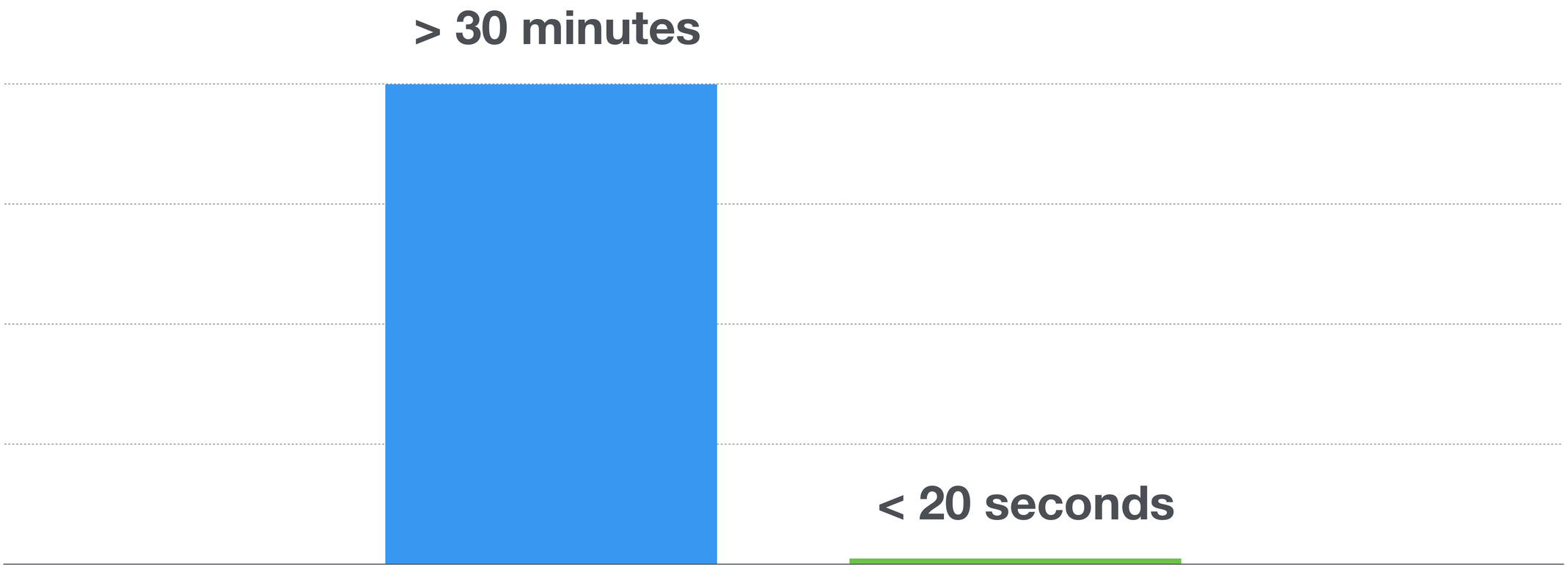
FAST CLEANUP

DeleteFilesInRange

FAST CLEANUP

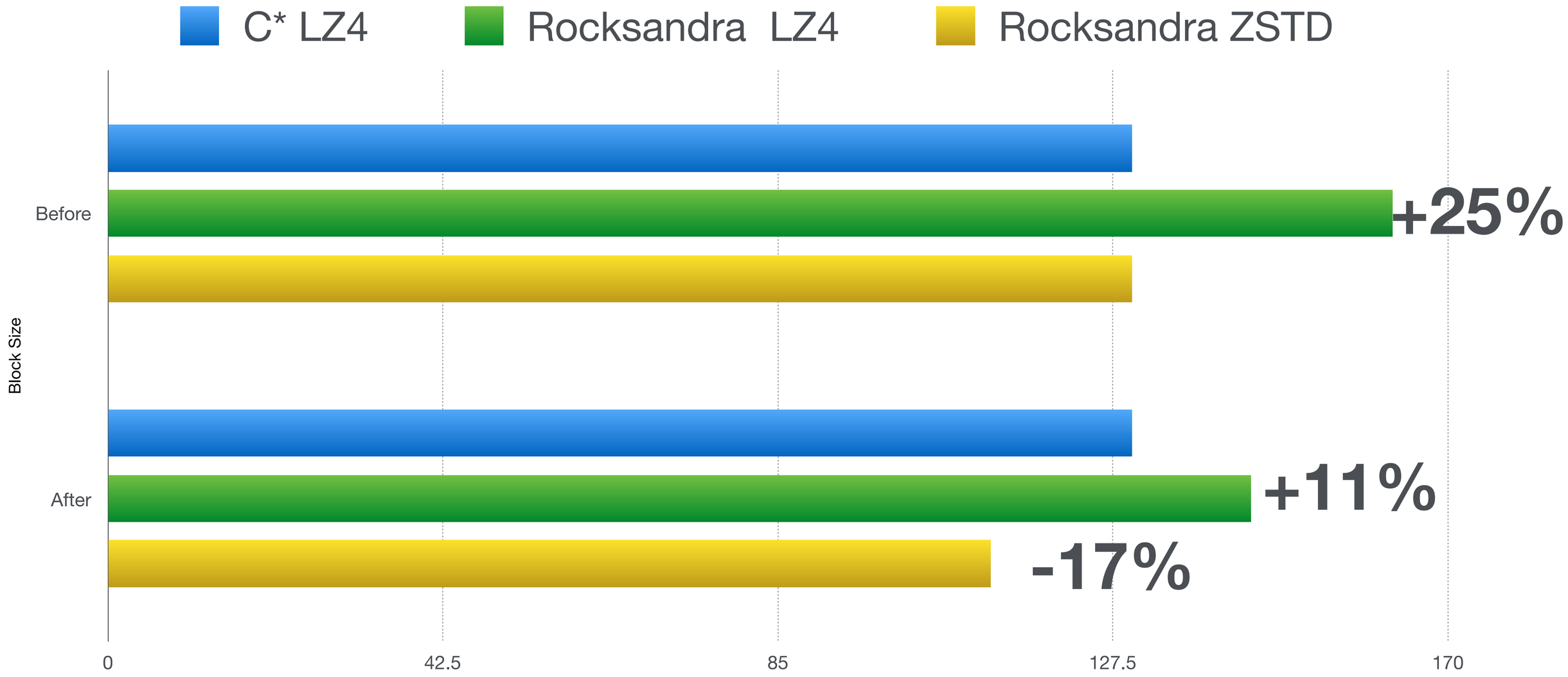
DeleteFilesInRange

FAST CLEANUP



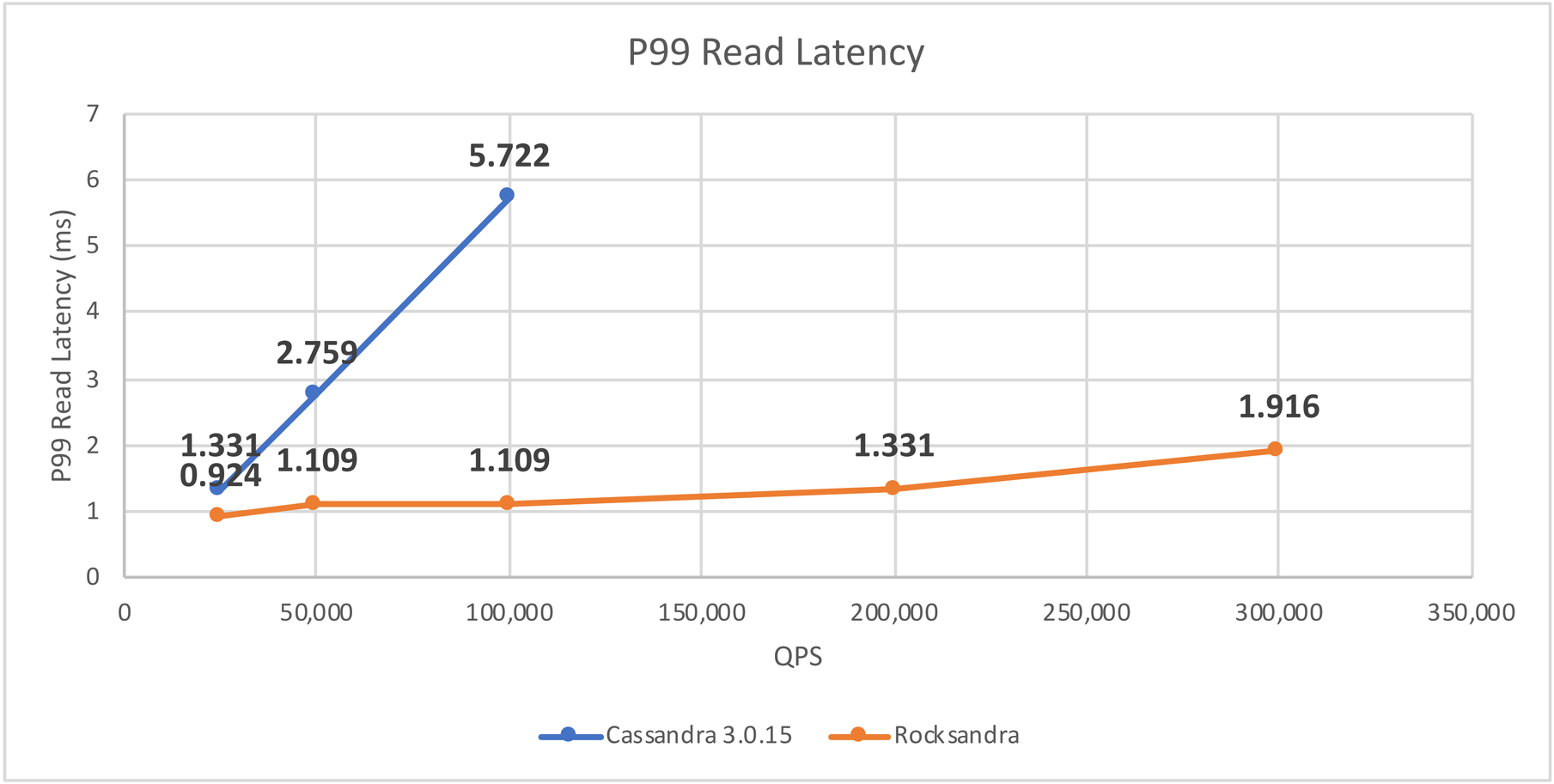
SPACE AMPLIFICATION

Closing the Gap

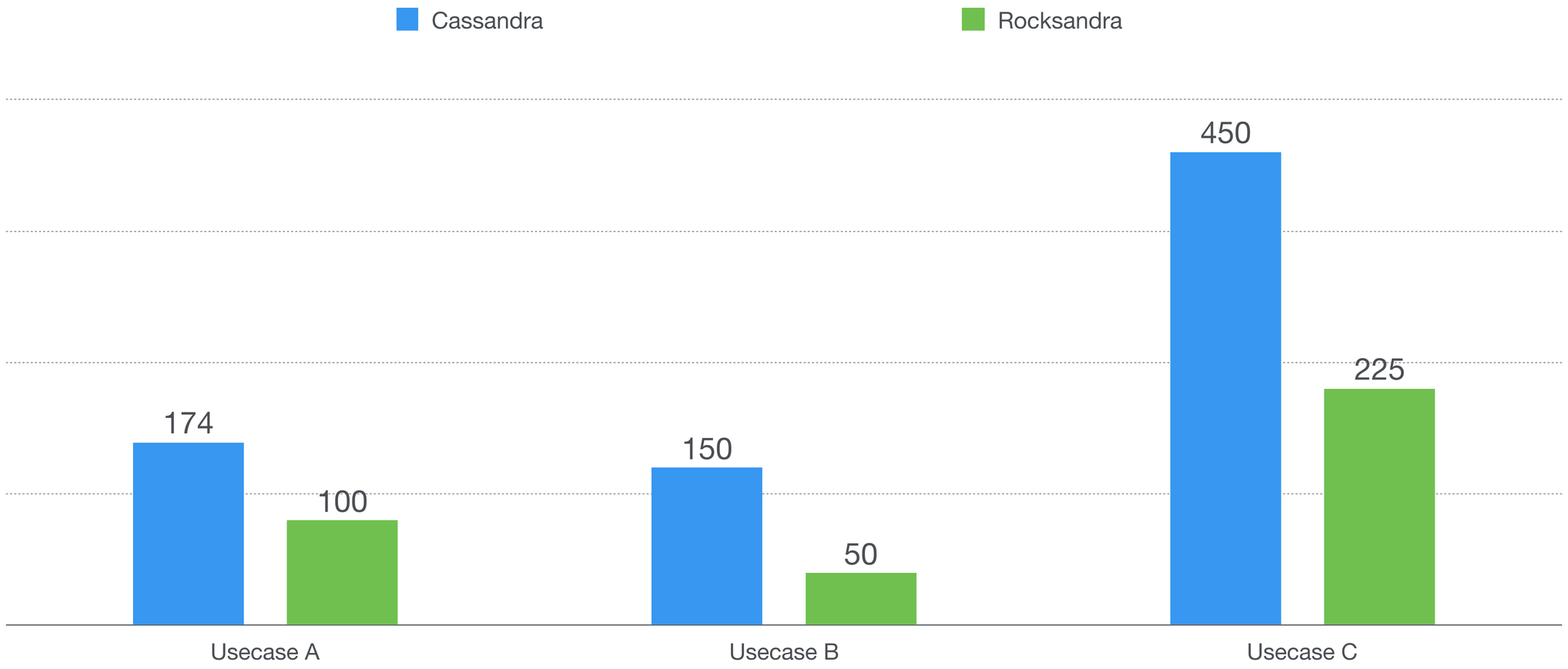


2 YEARS LEARNINGS IN PRODUCTION

LEARNING: THROUGHPUT



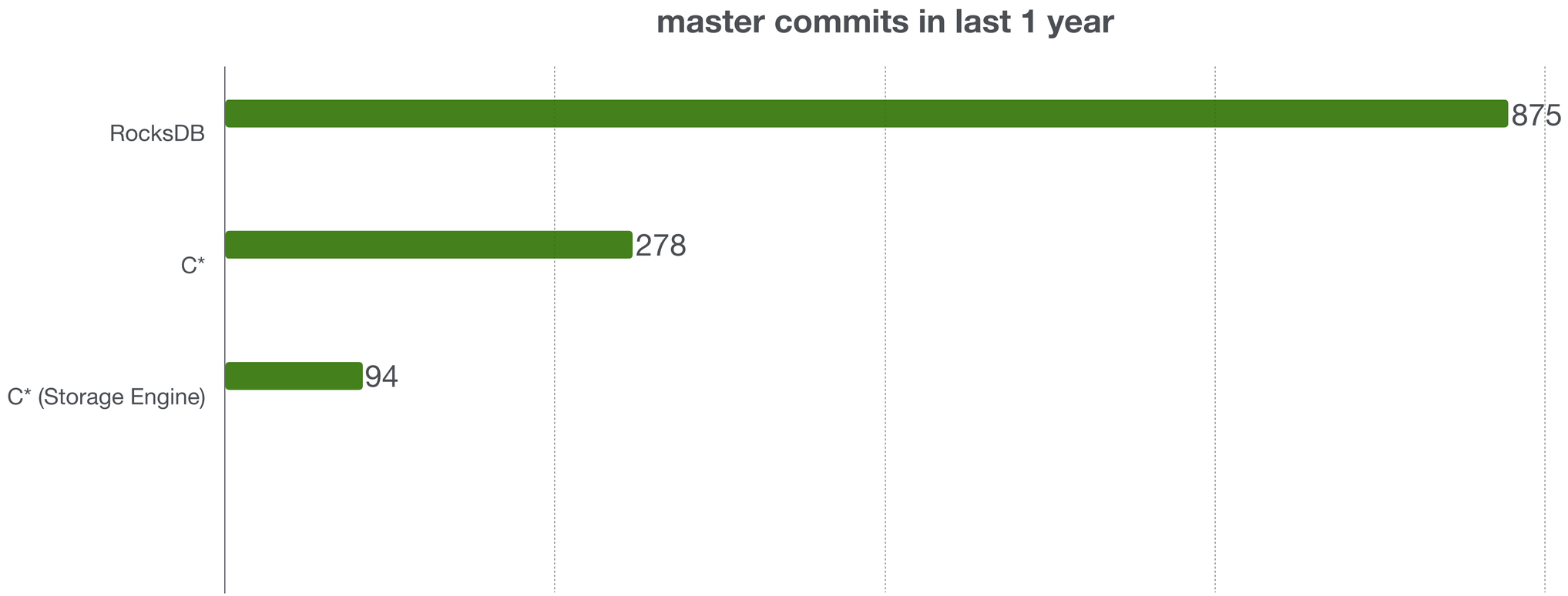
LEARNING: HIGH THROUGHPUT = LESS $$$



LEARNING: OPERATION COST LOWER

- Faster streaming (**doubled throughput**)
- Faster cleanup (**hours vs seconds**)
- Faster major compaction (**days vs hours**)
- Faster LSM tree shaping (**days vs hours**)
  - https://rocksdb.org/blog/2015/07/23/dynamic-level.html

LEARNING: ENJOY AN ACTIVE COMMUNITY



SUMMARY

- It has been 2 years since we start, time fly fast!
- We got huge benefits from running Rocksandra in instagram
  - higher efficiency
  - lower operating costs
- We are keep working on improving it!

FUTURE WORK

- Add more missing features
- Rebase to Cassandra 4.0
- More contribution back to RocksDB codebase, make RocksDB more java/ cassandra friendly

CASSANDRA-13474

Cassandra / CASSANDRA-13474

Cassandra pluggable storage engine

Details

Type:	New Feature	Status:	OPEN
Priority:	Normal	Resolution:	Unresolved
Component/s:	Legacy/Core	Fix Version/s:	None
Labels:	None		

Description

Instagram is working on a project to significantly reduce Cassandra's tail latency, by implementing a new storage engine on top of RocksDB, named Rocksandra.

We started a prototype of single column (key-value) use case, and then implemented a full design to support most of the data types and data models in Cassandra, as well as streaming.

After a year of development and testing, we have rolled out the Rocksandra project to our internal deployments, and observed 3-4X reduction on P99 read latency in general, even more than 10 times reduction for some use cases.

We published a blog post about the wins and the benchmark metrics on AWS environment. https://engineering.instagram.com/open-sourcing-a-10x-reduction-in-apache-cassandra-tail-latency-d64f86b43589

I think the biggest performance win comes from we get rid of most Java garbages created by current read/write path and compactions, which reduces the JVM overhead and makes the latency to be more predictable.

We are very excited about the potential performance gain. As the next step, I propose to make the Cassandra storage engine to be pluggable (like Mysql and MongoDB), and we are very interested in providing RocksDB as one storage option with more predictable performance, together with community.

Design doc for pluggable storage engine: https://docs.google.com/document/d/1suZlvhzgB6NIyBNpM9nxoHxz_Ri7qAm-UEO8v8AIFsc/edit

Issue Links

relates to	CASSANDRA-13476 RocksDB based storage engine	OPEN

Sub-Tasks

1.	Pluggable storage engine design	OPEN	Dikang Gu
2.	Refactor streaming	RESOLVED	Blake Eggleston
3.	Refactor repair	RESOLVED	Blake Eggleston
4.	Refactor read path	PATCH AVAIL...	Dikang Gu
5.	Refactor write path	RESOLVED	Blake Eggleston
6.	Refactor compaction	OPEN	*Unassigned*
7.	Refactor Keyspace/CFS operations	OPEN	*Unassigned*
8.	Refactor metrics	OPEN	*Unassigned*
9.	Refactor Indexes	OPEN	*Unassigned*
10.	Abstract storage engine API from Keyspace/CFS	OPEN	*Unassigned*
11.	Refactor Schema/Metadata	OPEN	*Unassigned*
12.	Refactor commitlog	OPEN	*Unassigned*

https://github.com/Instagram/cassandra/tree/rocks_3.0

EVERYTHING OPEN SOURCED

And Need Your Help!