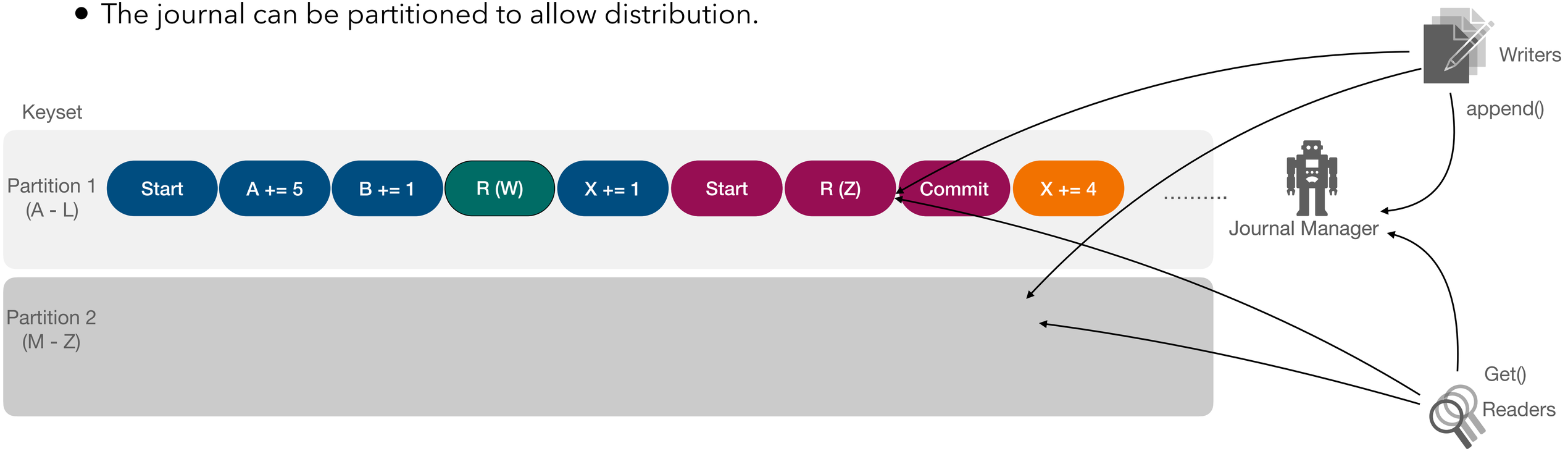
Persisting the Cache in journal based Data-stores

Managing Journal Size through truncation

Ayush Pandey
Dr. Annette Bieniusa
Dr. Marc Shapiro

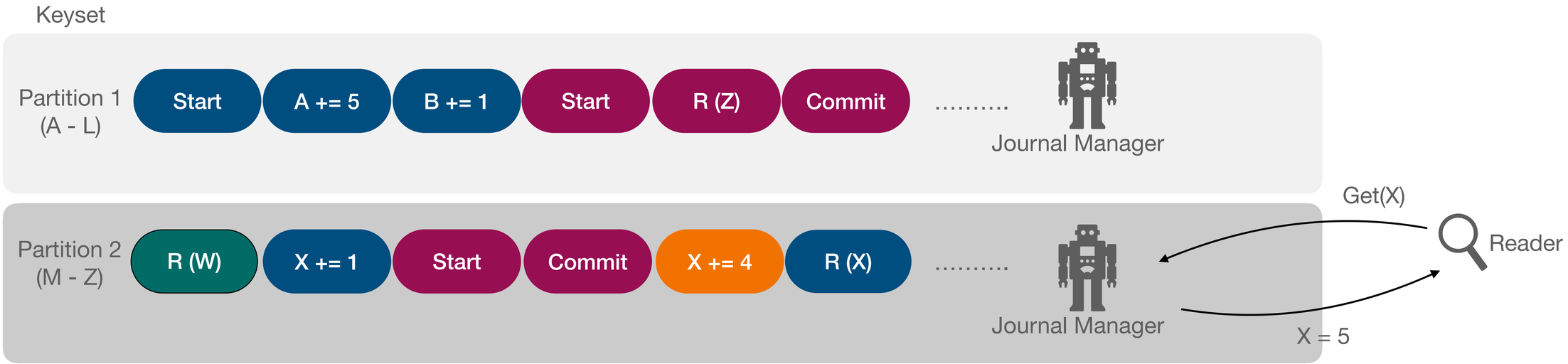
What is a journal?

- Sequence of operations
- Written in a specific order by transactions.
- To be read by transactions to materialise objects.
- The journal can be partitioned to allow distribution.



Reading Objects / Materialisation

- Read queries lead to materialisations.
- To answer a specific read, the request is sent to the responsible partition.
- The journal manager filters the operations for that key.
- The materialiser uses these filtered operations and creates the object.



The problem with journals and filtering.

- Each transaction adds and reads some operations to and from the journal.
- The journal is, in the original AntidoteDB design, unbounded and contains all the operations.[1]
- We need to filter this large set of read operations to materialise the objects.
- The time required to filter is proportional to the size of the journal.[2]
- To reduce the time required to filter, reduce the size of the read set.

[1] From the start of the system.

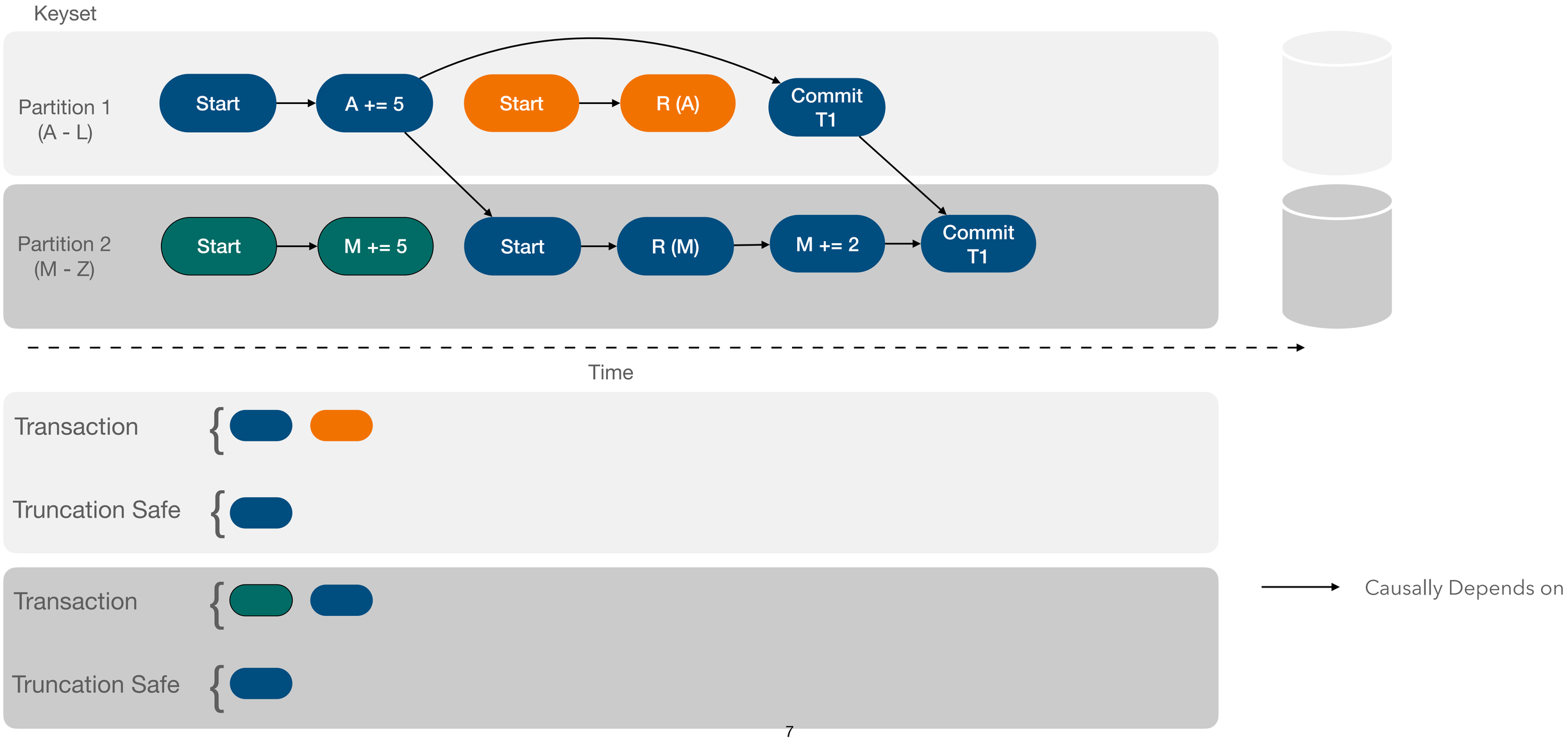
[2] Benchmarks done with a 16 partition deployment and 400 concurrent clients at max request rate

Truncate the journal !

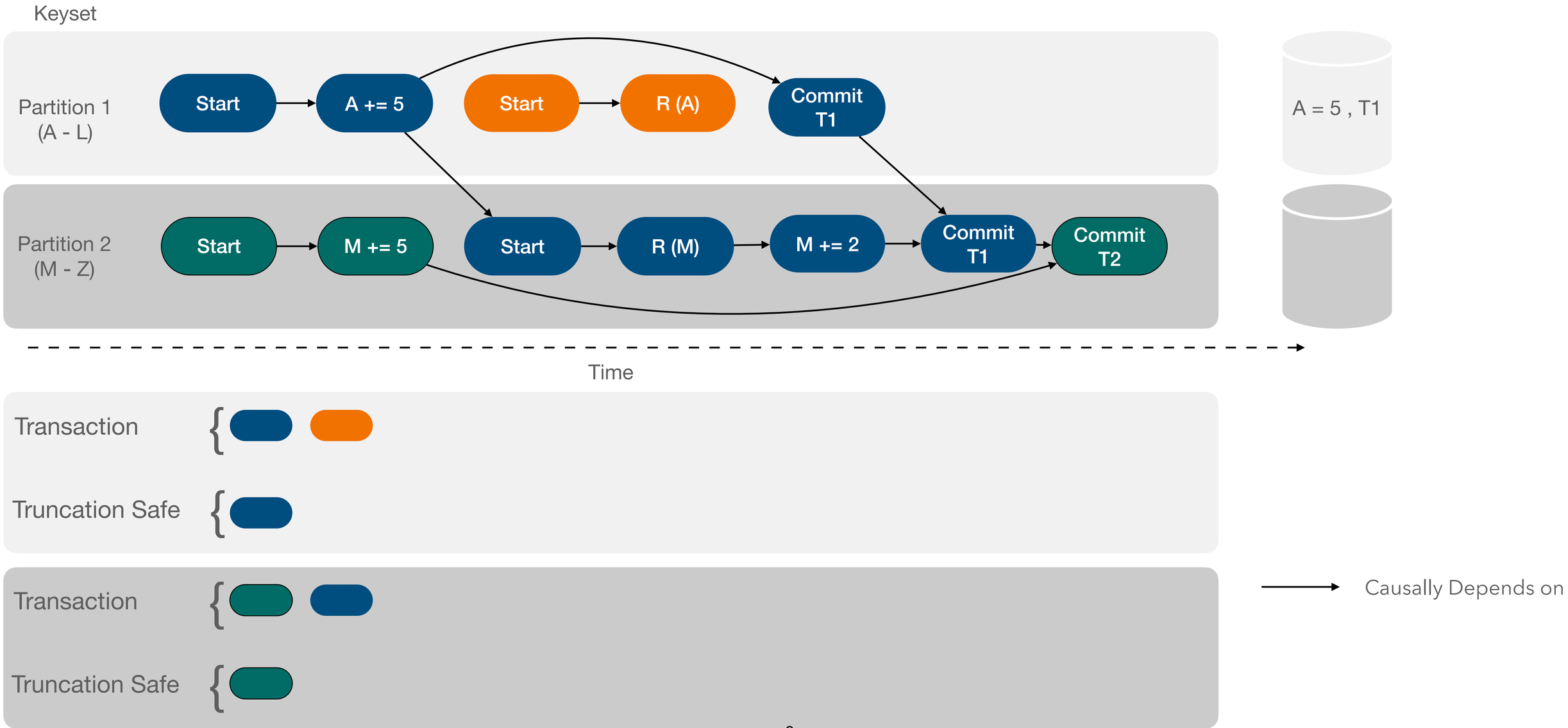
How to truncate a journal?

- Journal truncation is destructive.
- We need to maintain causal dependencies between transaction snapshots.
- To ensure safe truncation, there are invariants that need to be maintained.
- Conditions on **times of interest** during the operation of the database.
- Truncated operations need to be persisted (Check-pointed).

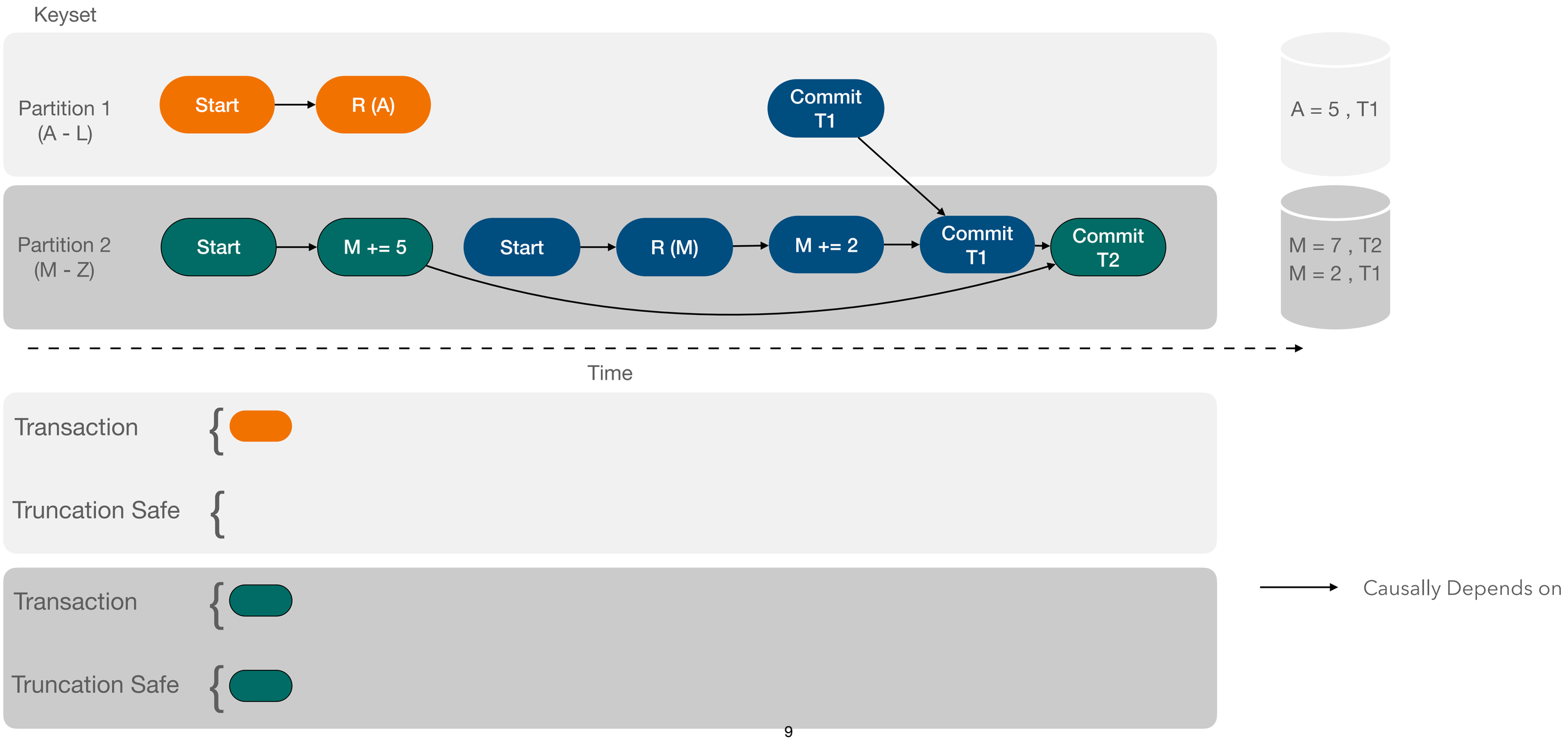
Truncation Visualised



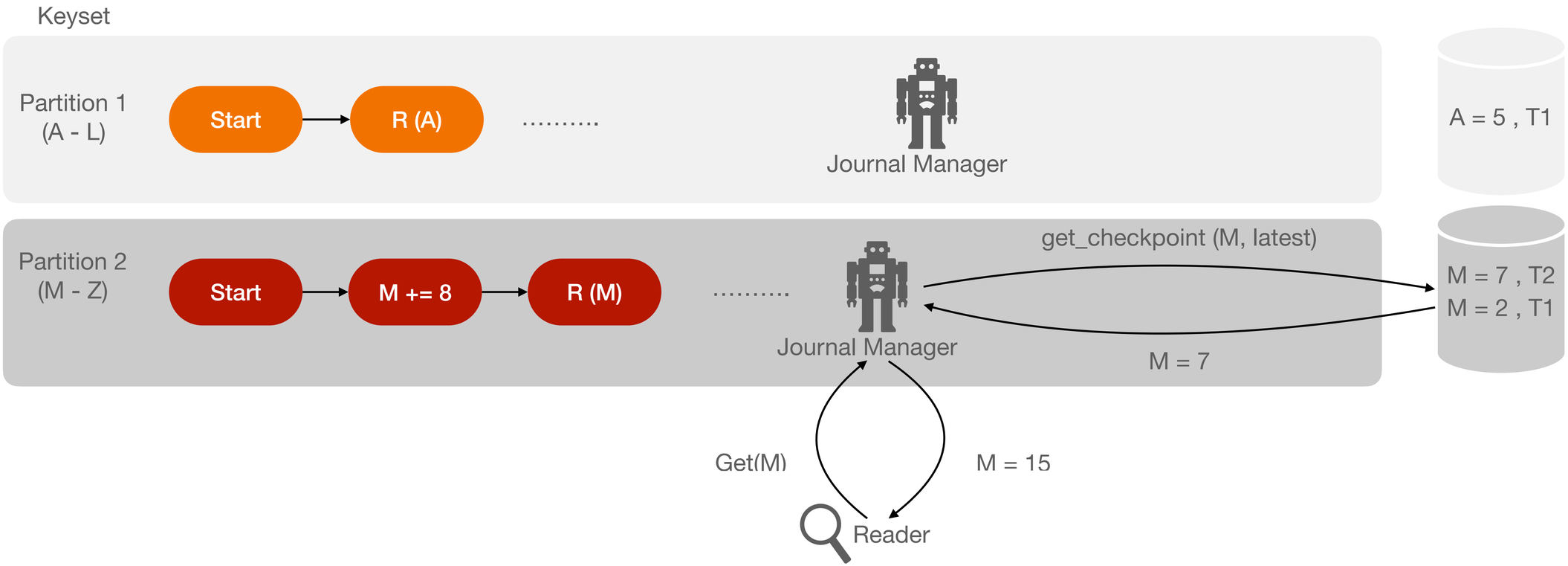
Truncation Visualised



Truncation Visualised



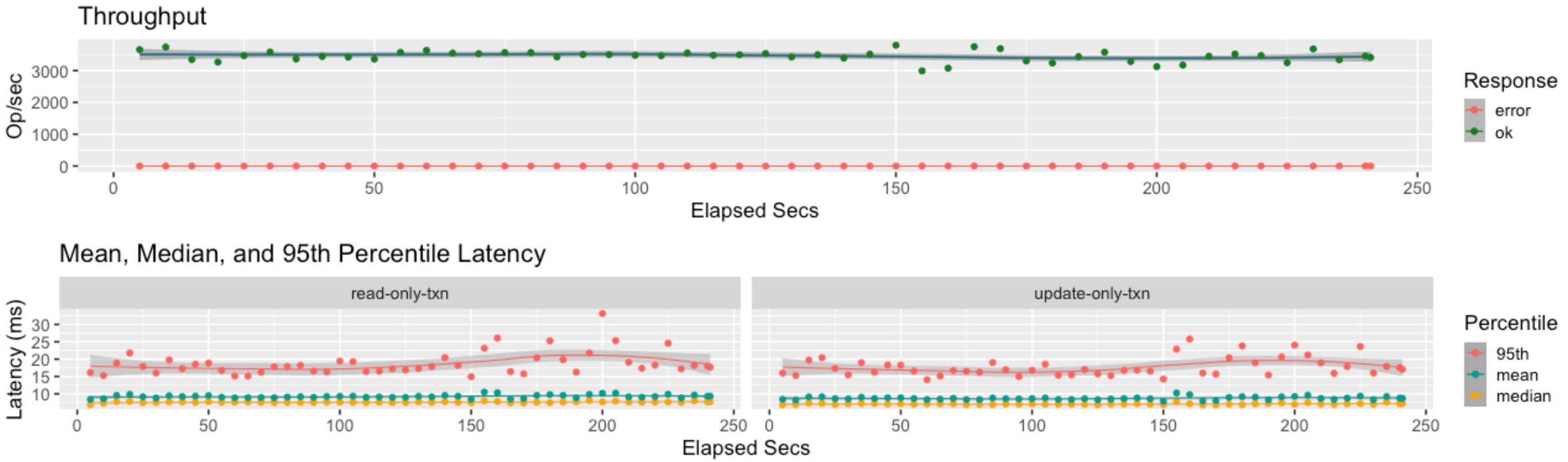
Reading from a truncated journal



Implementation at a glance

- Introducing a new library: **Gingko**.
- Restructuring the code into daemons and simplifying AntidoteDB:
  - Materialisation (Fill daemon)
  - Journal management (Gingko OP Log)
  - Journal Indexing (Index Daemon)
  - Checkpointing (Checkpoint Daemon)
  - Object Cache (Cache Daemon)
- Truncation is implemented via the Checkpoint Daemon.

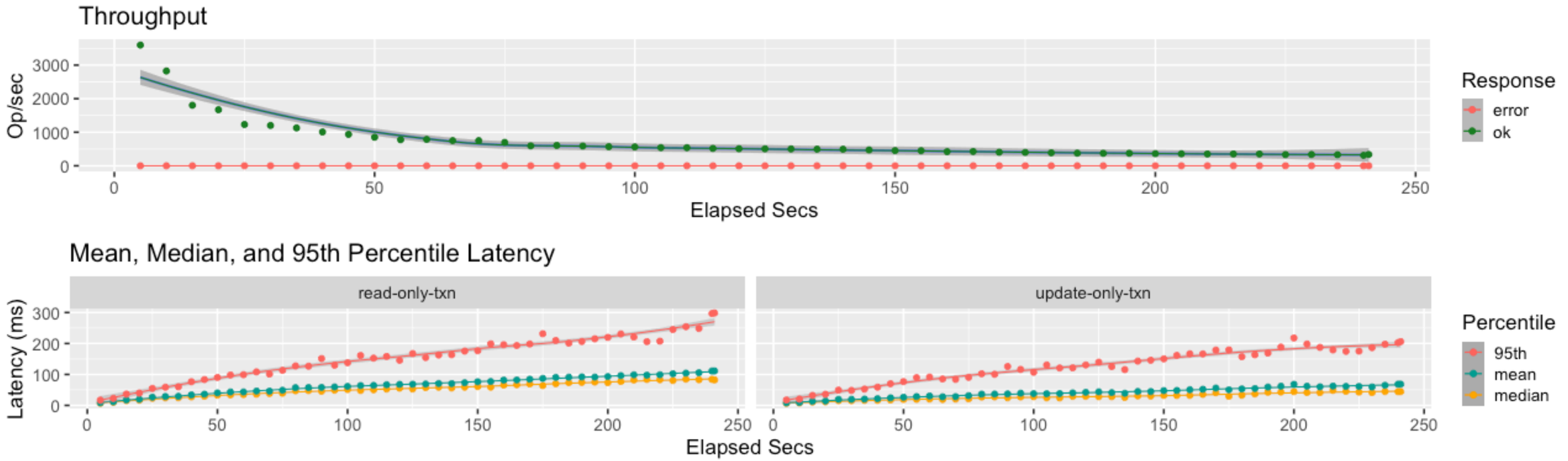
Performance Benchmarks



Baseline Performance of the new Implementation

Benchmark on an 8 core 3.2 Ghz consumer grade CPU With 16 GB RAM.

Performance Benchmarks



Observed decrease in throughput without Truncation

Benchmark on an 8 core 3.2 Ghz consumer grade CPU With 16 GB RAM.

Conclusion

- In journal based data-stores, Journal grows with operational time.
- We need the journal to be bounded to manage persistent space requirement.
- Journal can be bounded by truncating the oldest records.
- Truncation needs to be done with transaction starts and commits in consideration.

Thank you !

New Architecture of a single DC in AntidoteDB