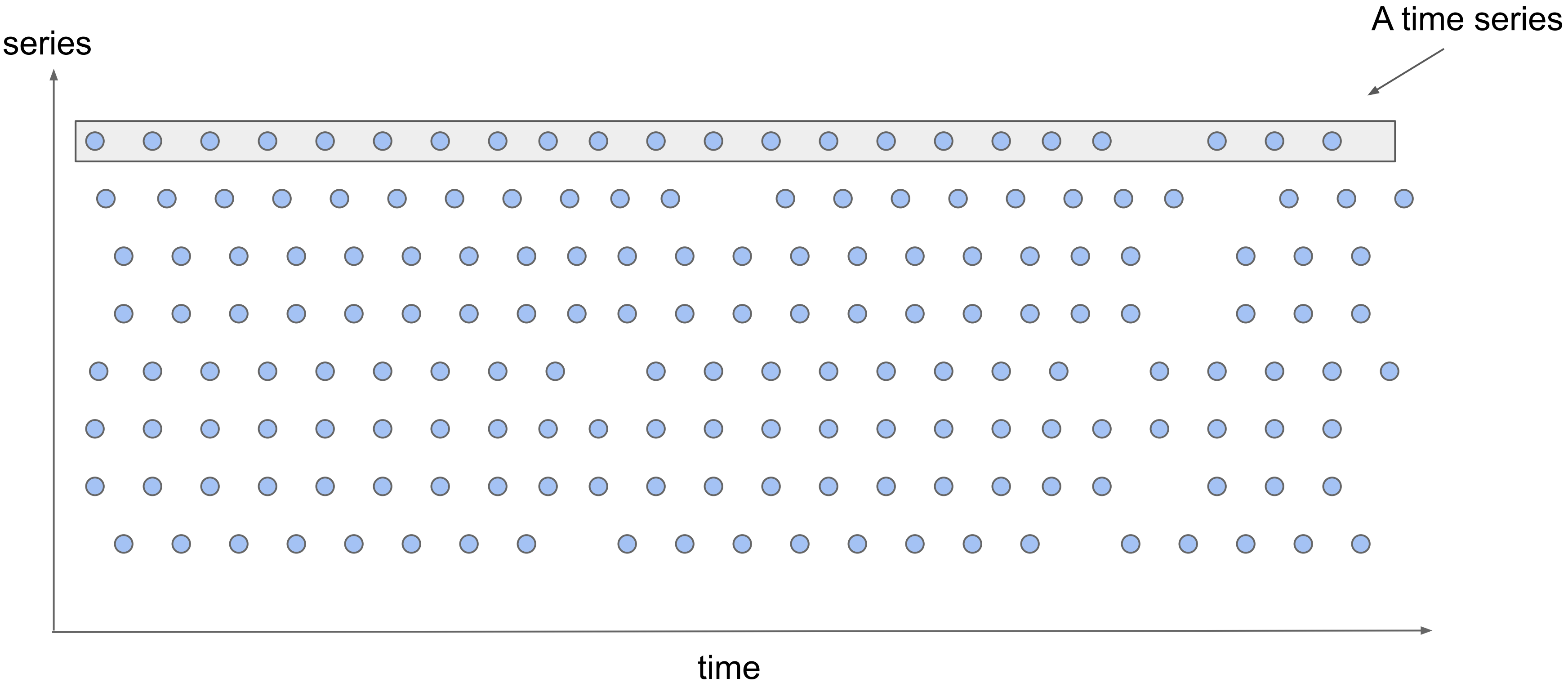
MVTO for isolation

A case for databases having lots of keys!
Specialised for time-series

Sreekar B
Gautham V

The problem



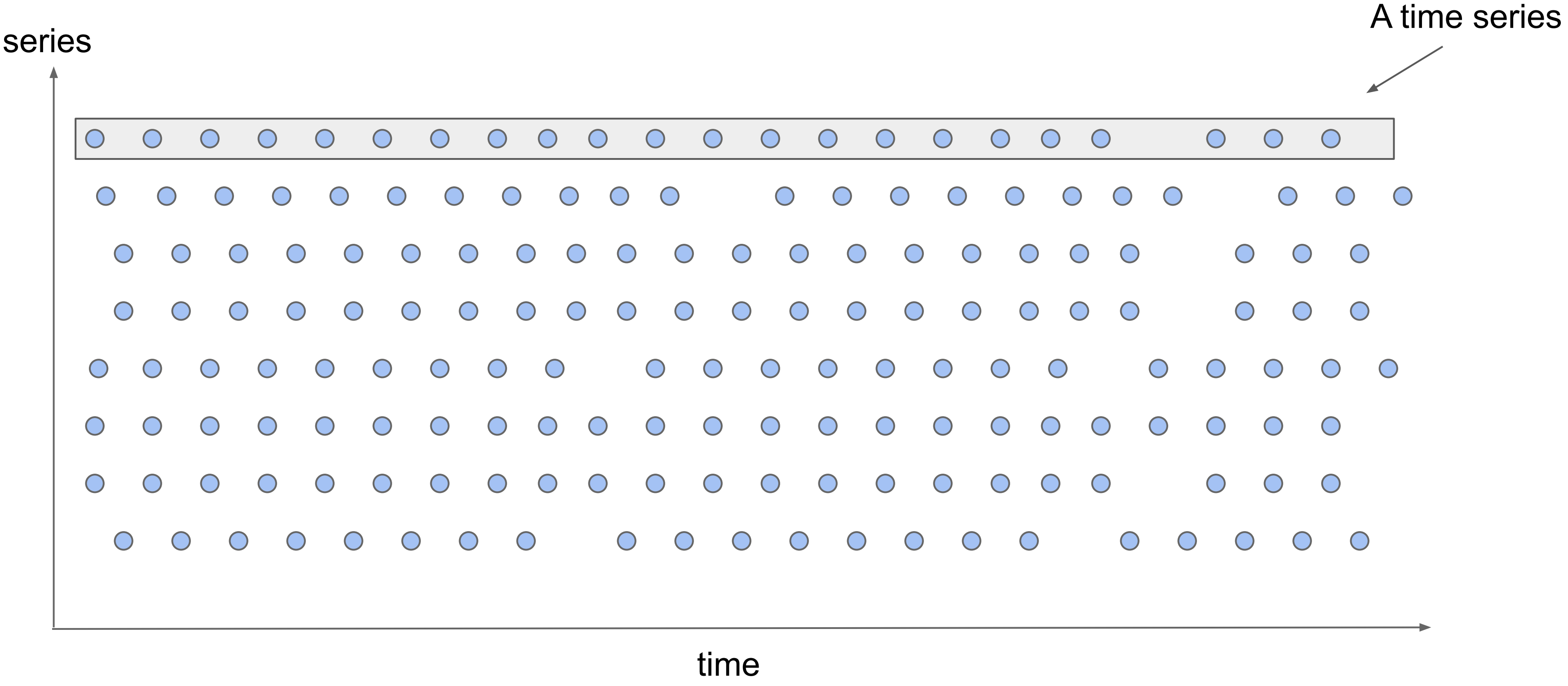
The problem

- You can have millions of series!
- You can have a single transaction write to MILLIONS of series!
- You cannot have a lock, and even a transaction manager, because with each transaction touching millions of series, it’ll mostly be rollbacks.
- Transaction manager will also need to manage state on millions of objects.

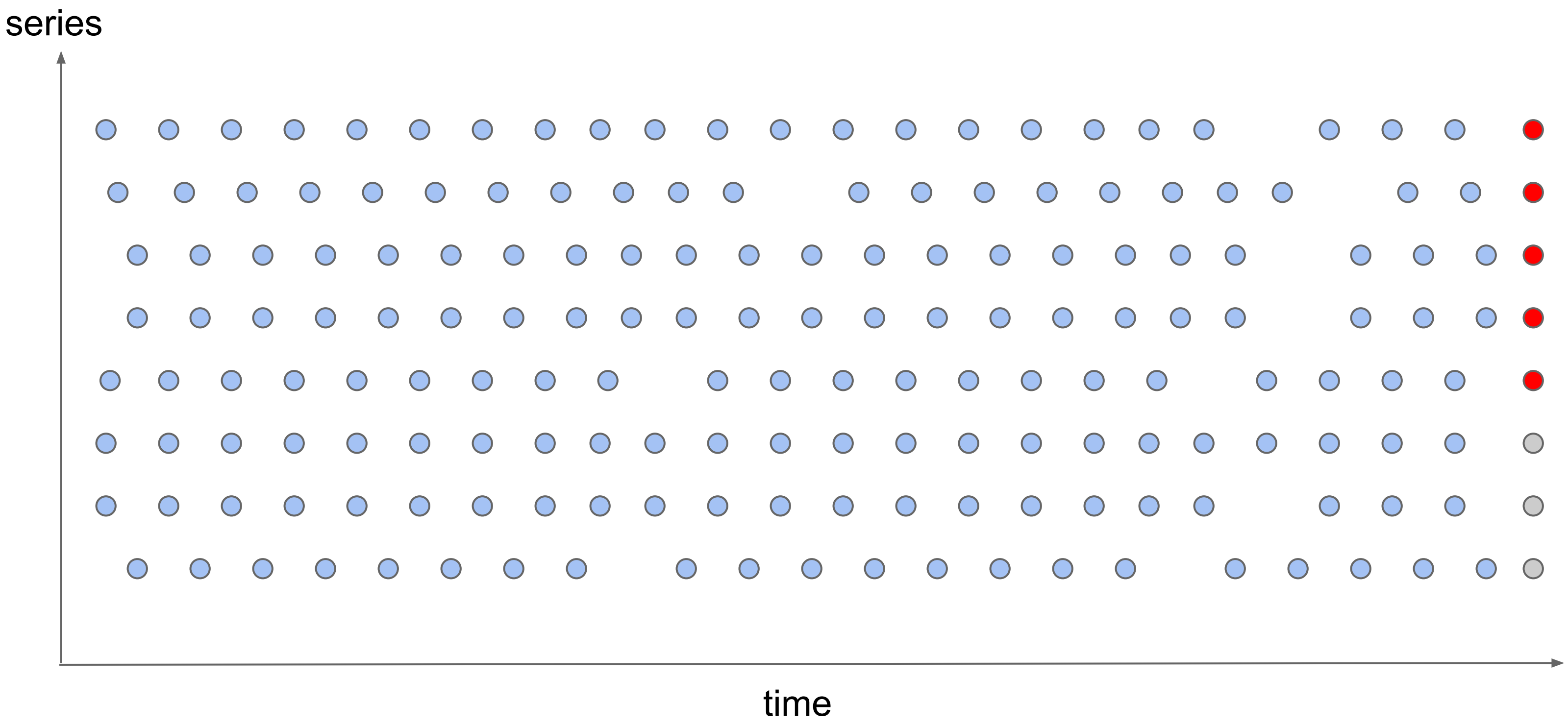
Solution

- A lock per series
- If a writer is writing to a series: lock, write, unlock
- If a reader is reading from a series: rlock, read, runlock
- Scales well, no one transaction is blocked for too long, perfect?

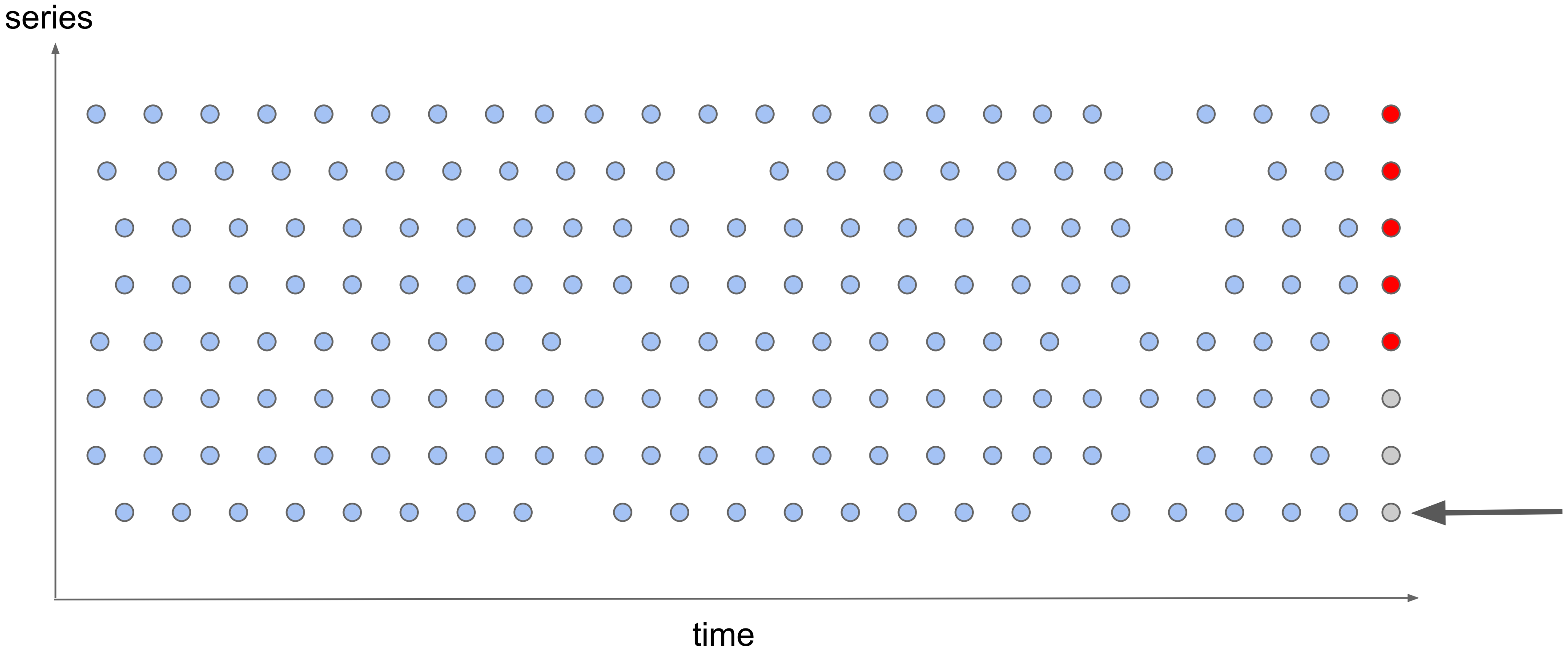
Problem: Isolation!



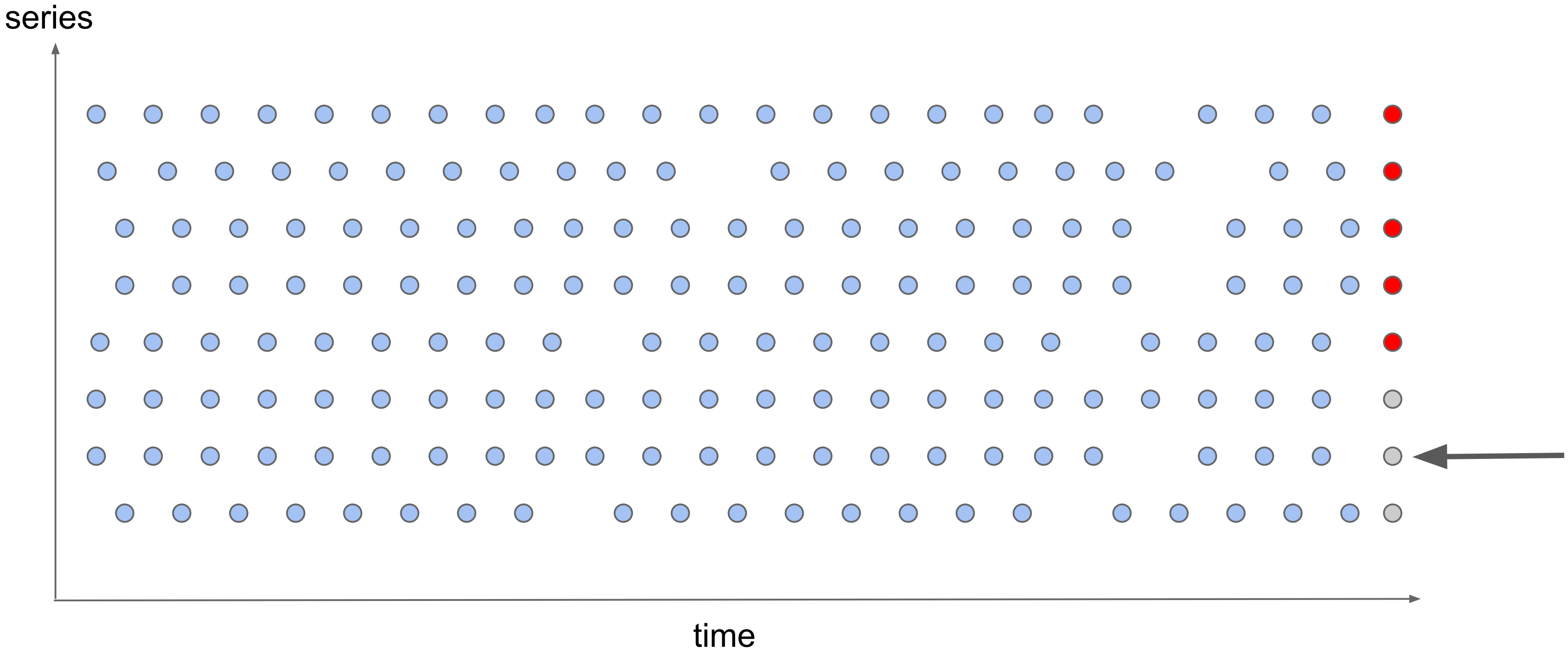
Problem: Isolation



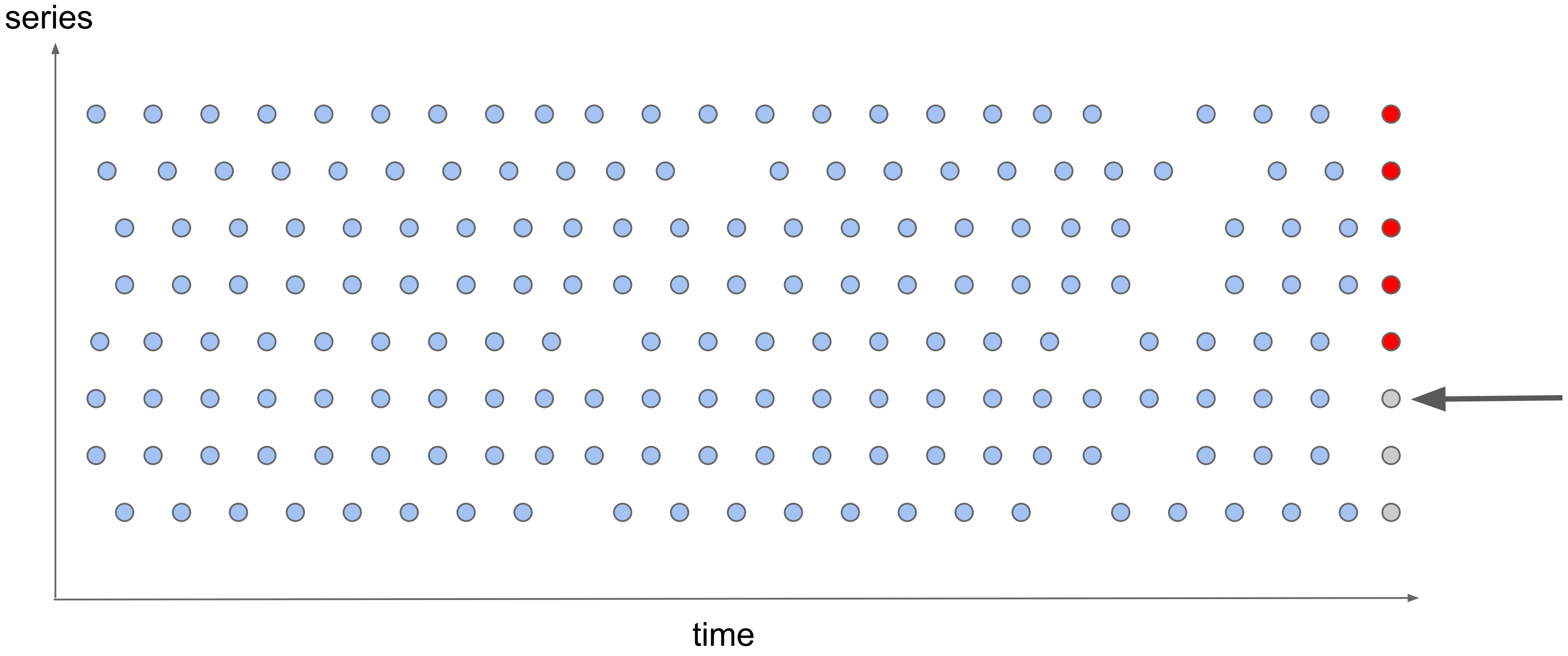
Problem: Isolation



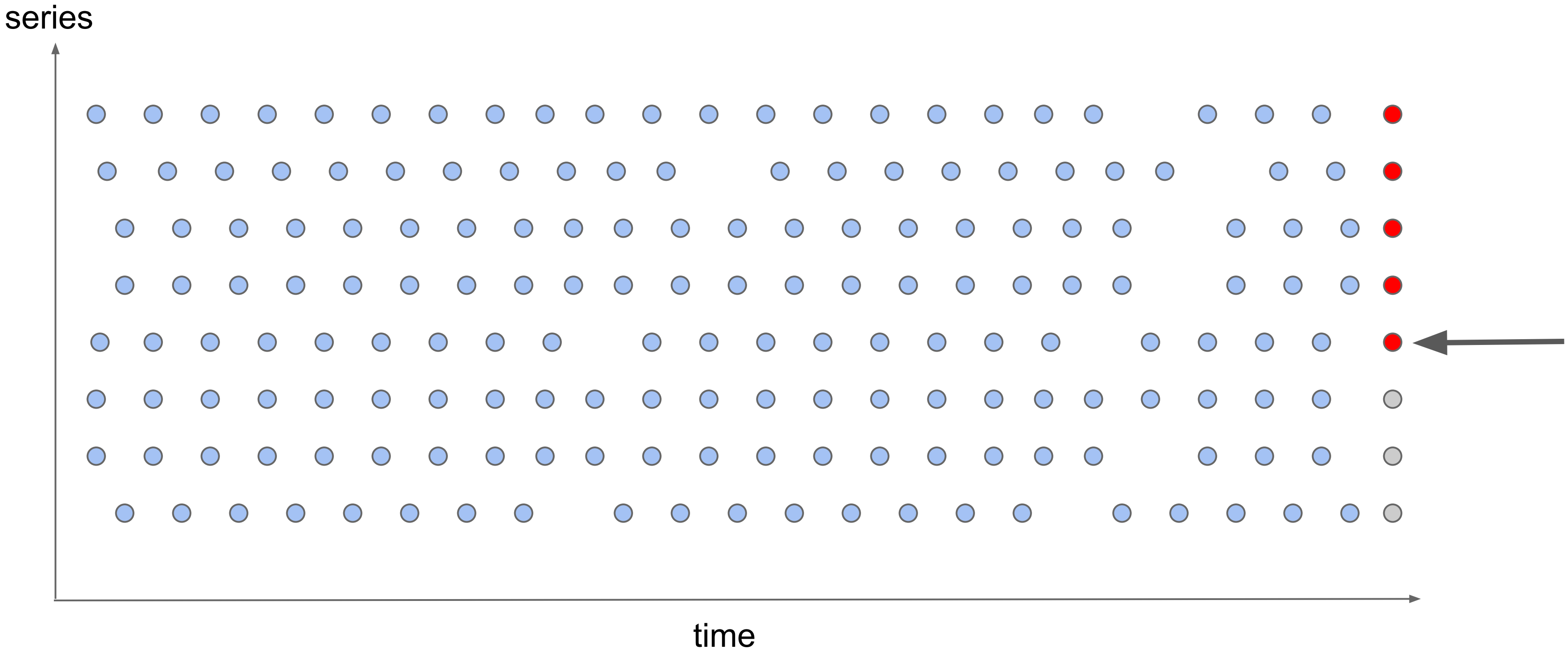
Problem: Isolation



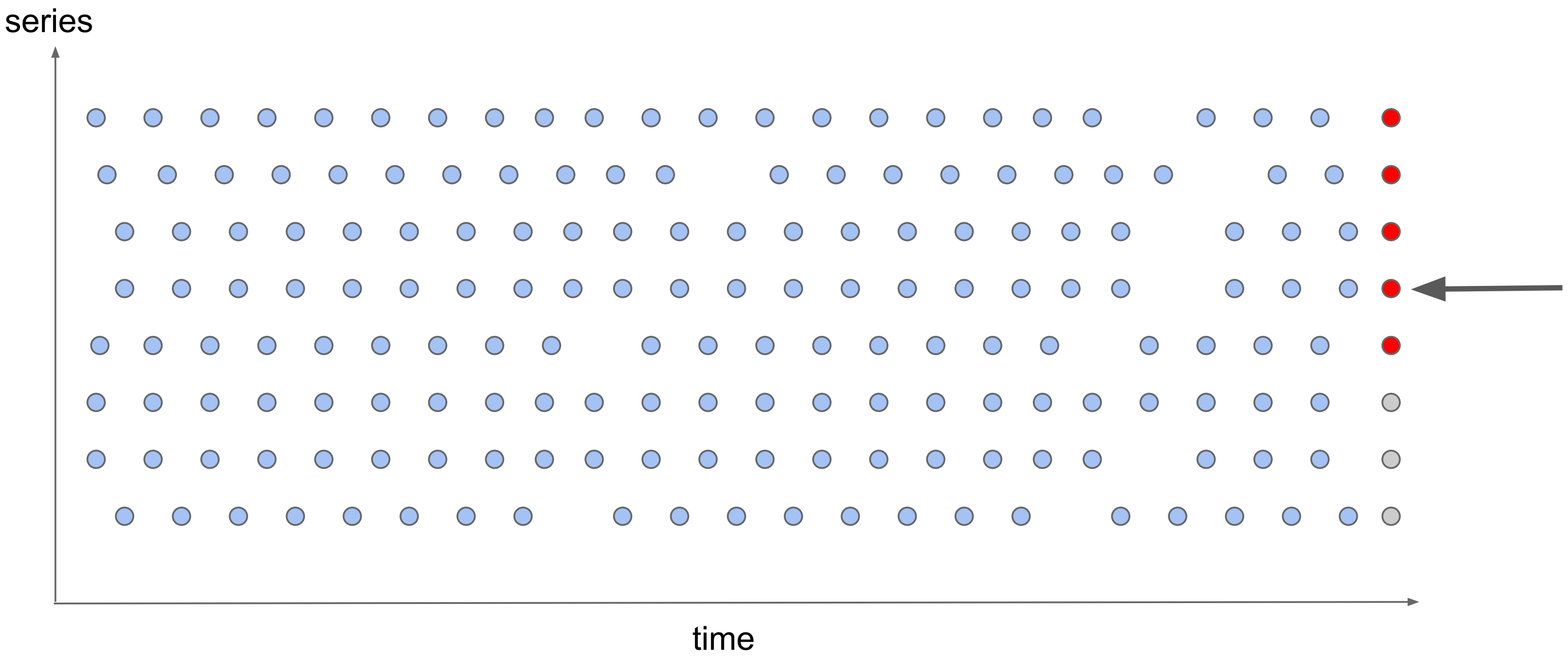
Problem: Isolation



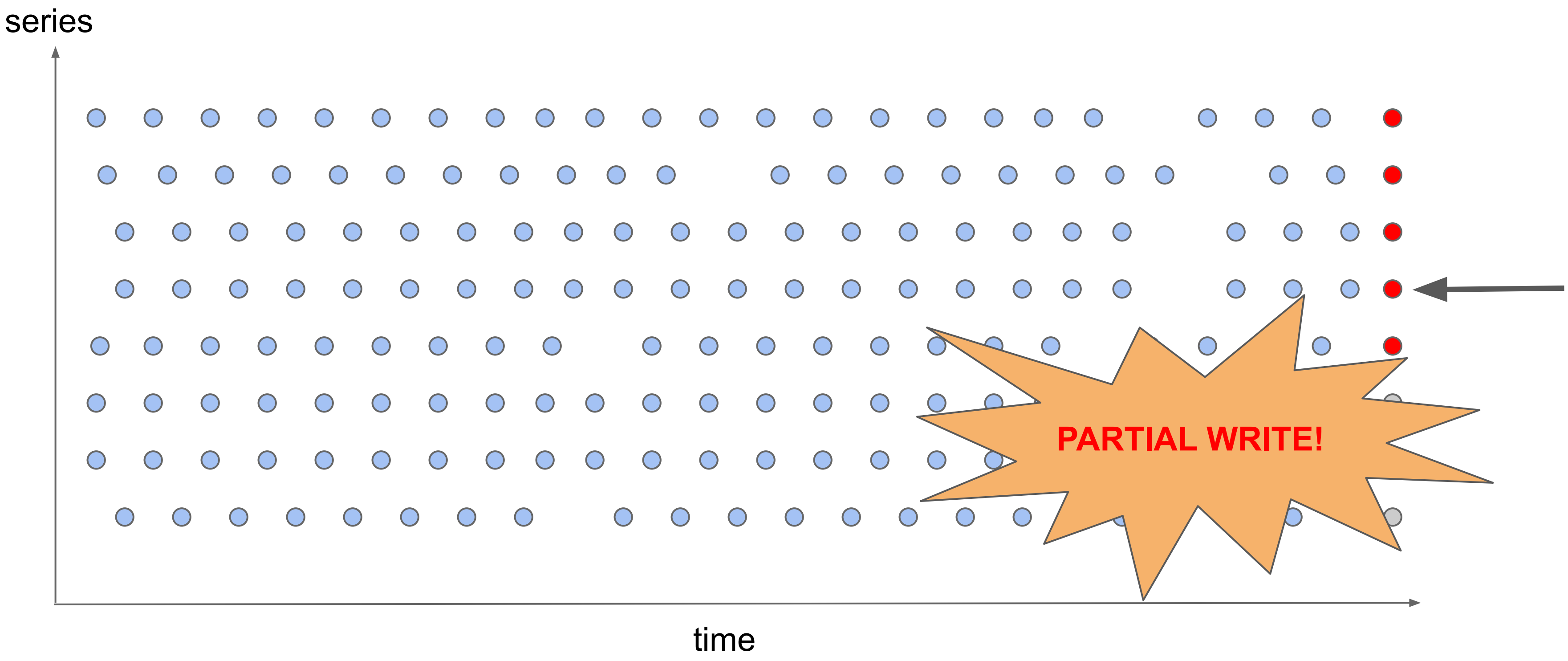
Problem: Isolation



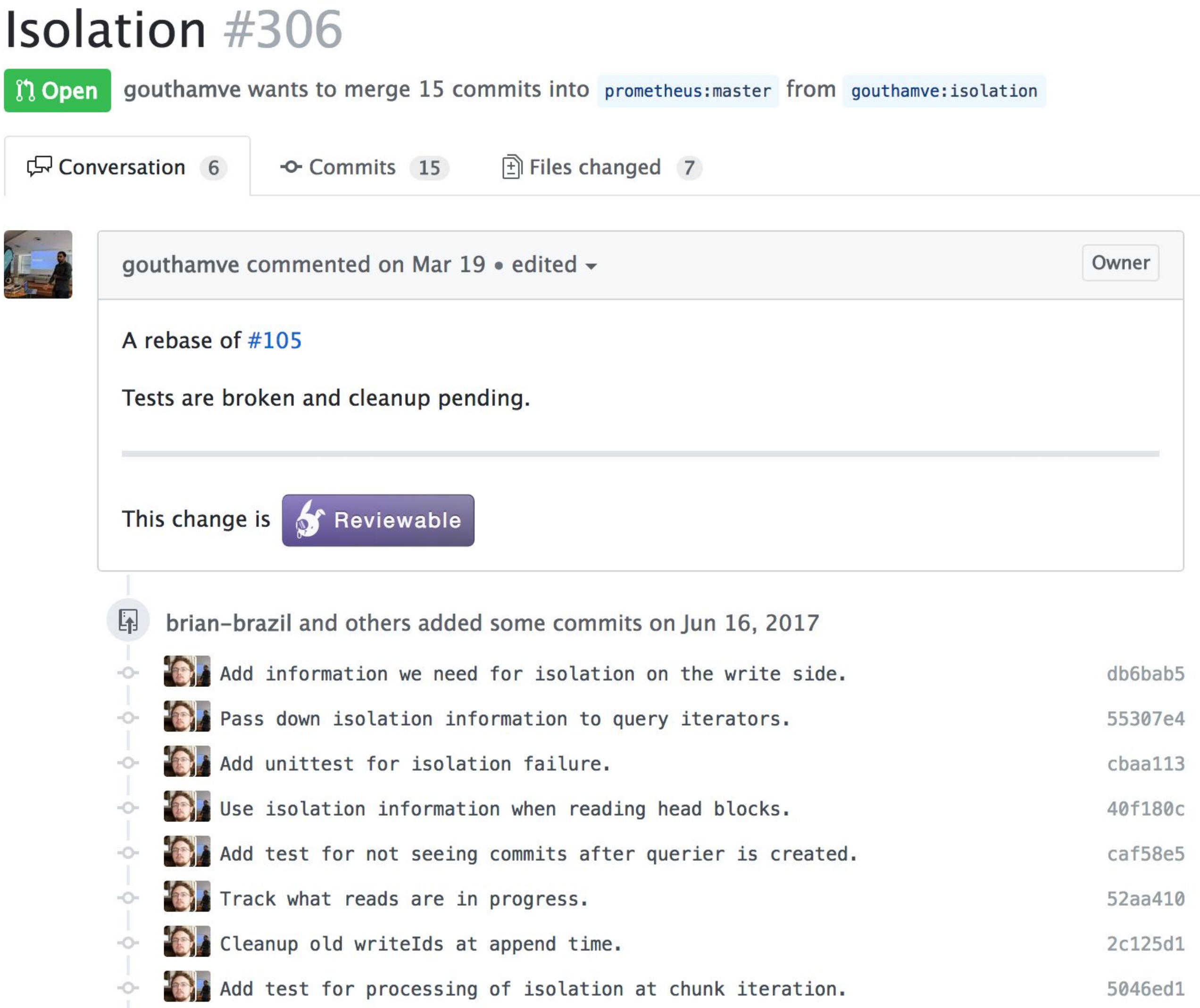
Problem: Isolation



Problem: Isolation



Solution



Isolation #306

Open gouthamve wants to merge 15 commits into `prometheus:master` from `gouthamve:isolation`

gouthamve commented on Mar 19 • edited

A rebase of #105

Tests are broken and cleanup pending.

This change is Reviewable

brian-brazil and others added some commits on Jun 16, 2017

Add information we need for isolation on the write side.	db6bab5
Pass down isolation information to query iterators.	55307e4
Add unittest for isolation failure.	cbaa113
Use isolation information when reading head blocks.	40f180c
Add test for not seeing commits after querier is created.	caf58e5
Track what reads are in progress.	52aa410
Cleanup old writeIds at append time.	2c125d1
Add test for processing of isolation at chunk iteration.	5046ed1

MVTO

- Each sample has a version attached to it

- When a reader tries to read, it will see if it’s **legal,** to read it, and reads an older value if not.

GC

- We do GC the normal way and we have a twist on it as well :)
- Now millions of series == 100s of millions of versions == lots of memory!
- We cannot do periodic GC as it might be too frequent or too late.
- We do GC on write!

GC

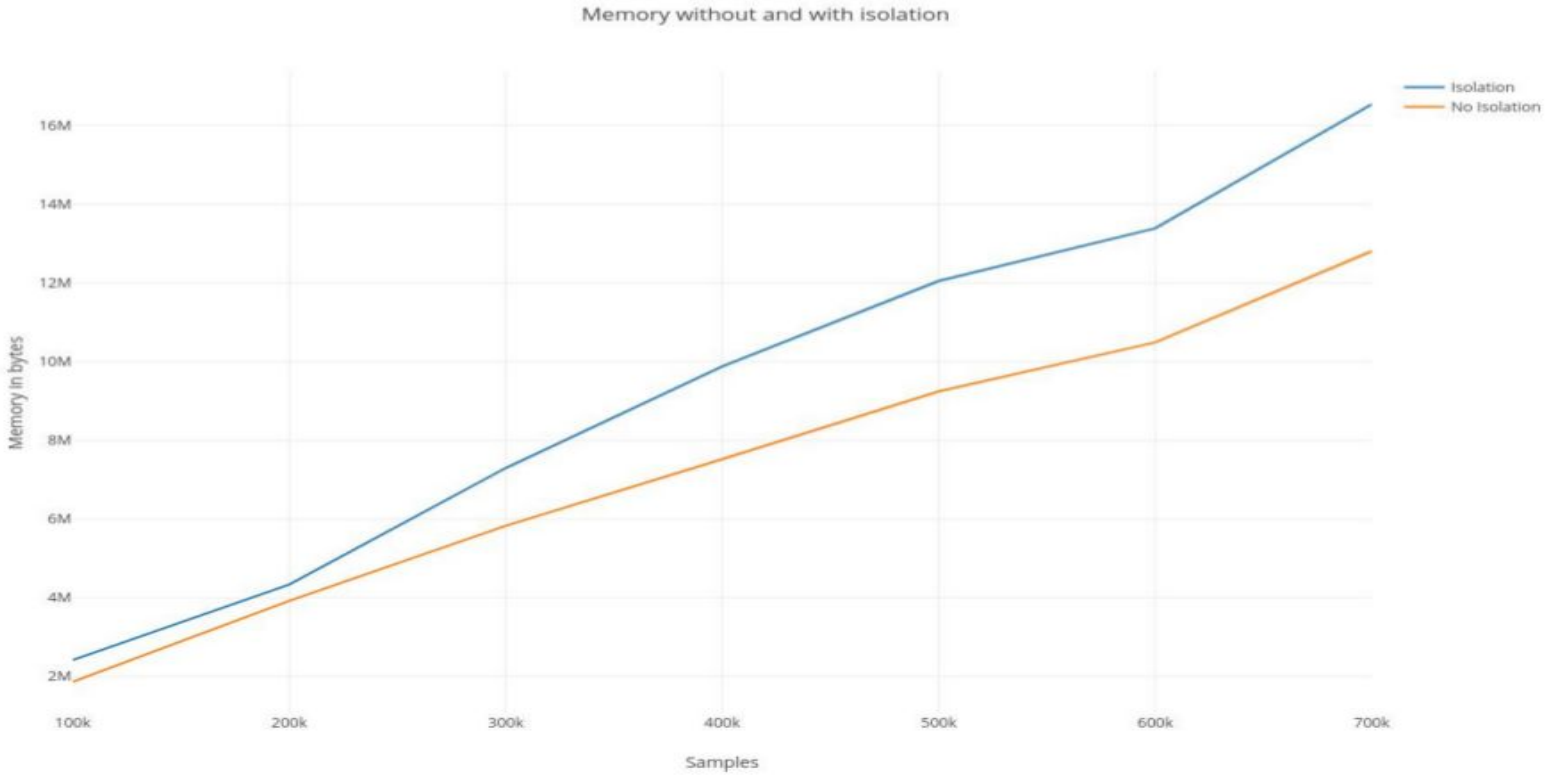
- When we create appender, we get the oldest active transaction’s id.

- When we write to series, we delete the versions lower than the above transaction id.

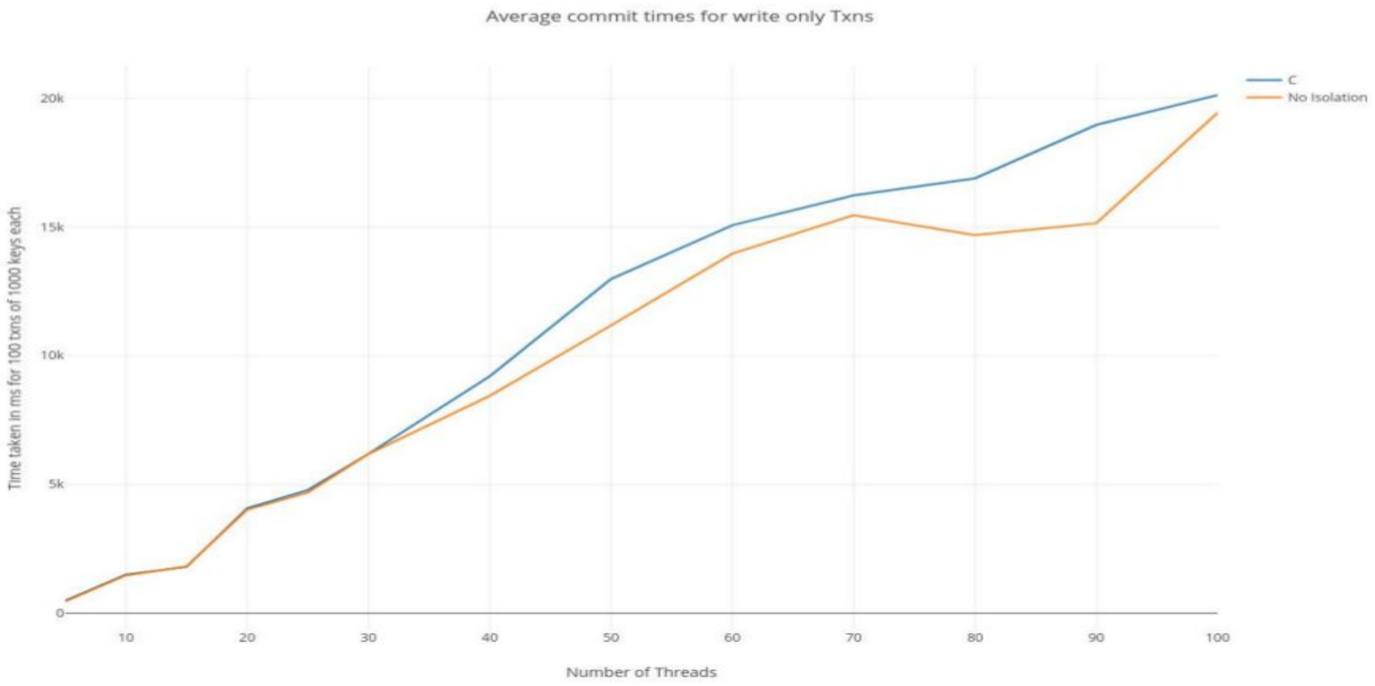
- This means two things:
  * Active series will have just a couple of versions, or so.
  * Periodic GC is required as some series might have turned inactive (receive no writes).

- We evaluated doing it on reads too, but dropped it due to performance over-head. Reads tend to touch much more series than writes. And we want reads to be faster.

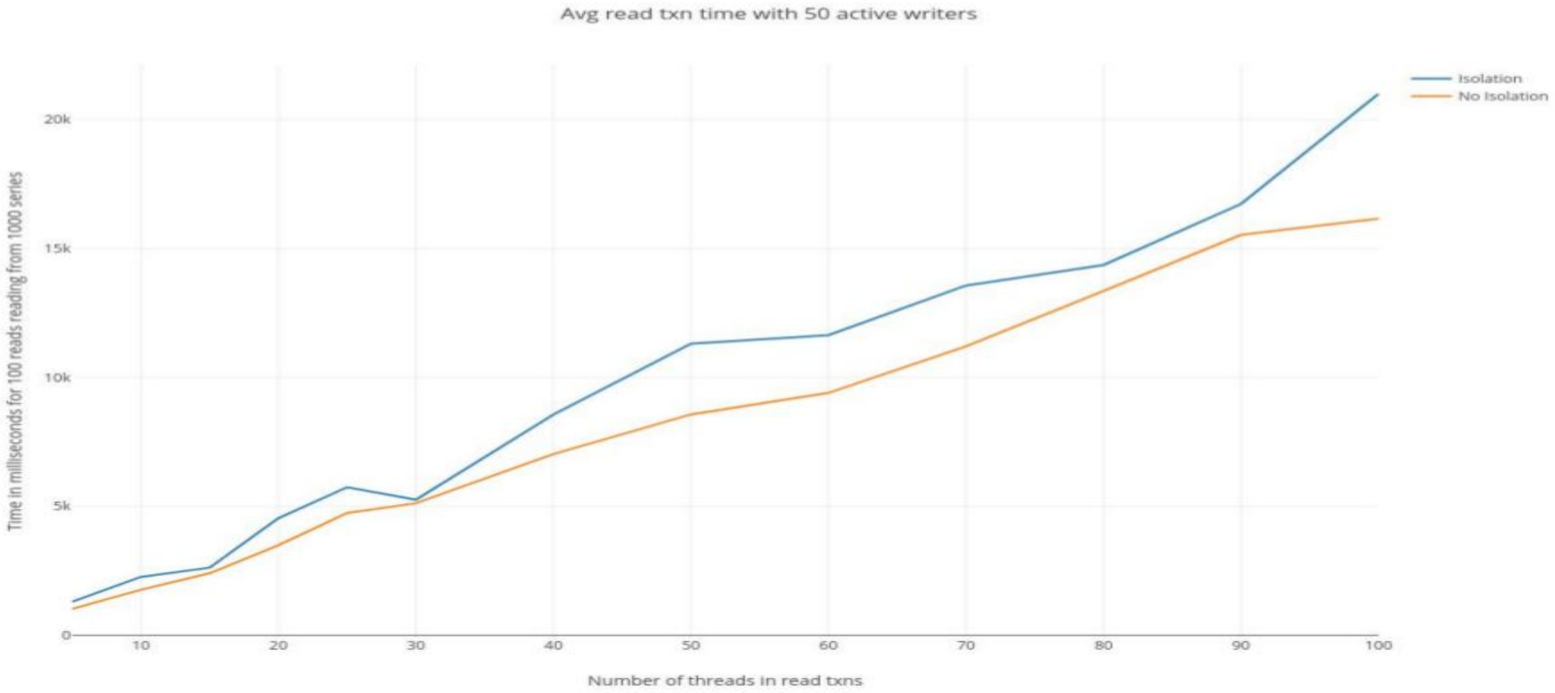
Memory Consumed



Throughput of Write Transactions



Throughput of Read Transactions


Avg read txn time with 50 active writers
Isolation
No Isolation
Time in milliseconds for 100 reads reading from 1000 series
5k
10k
15k
20k
Number of threads in read txns