Token Management @ Scale

Jay Zhuang - Software Engineer @ Instagram

About me

Software Engineer @ Instagram Cassandra Team

Cassandra Committer

Before:

- Uber Cassandra Team
- Amazon AWS

Overview

1 Challenges

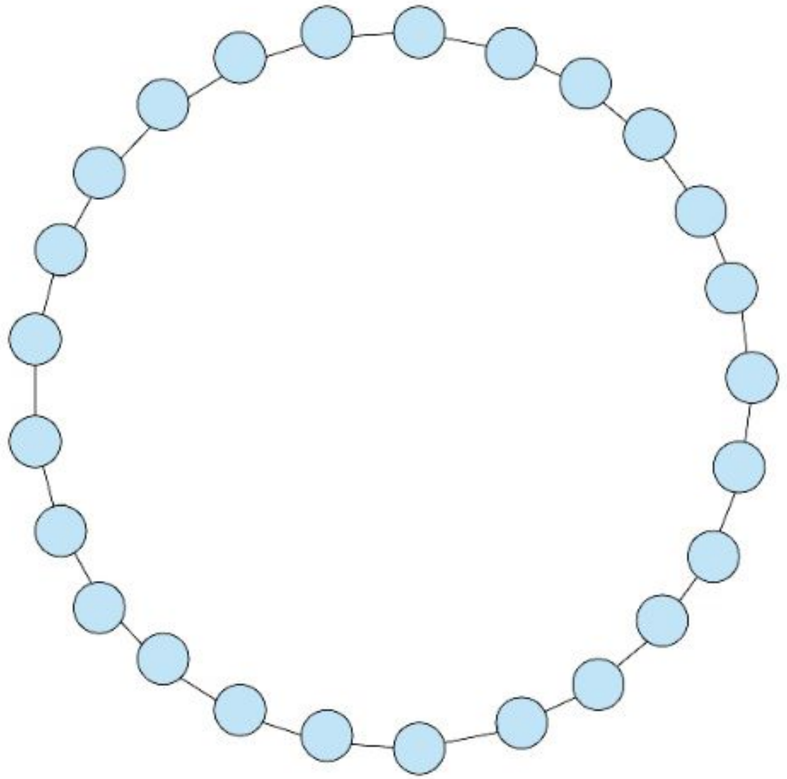
2 Improvements

3 Future Work

Challenges

Token Management & Gossip Protocol

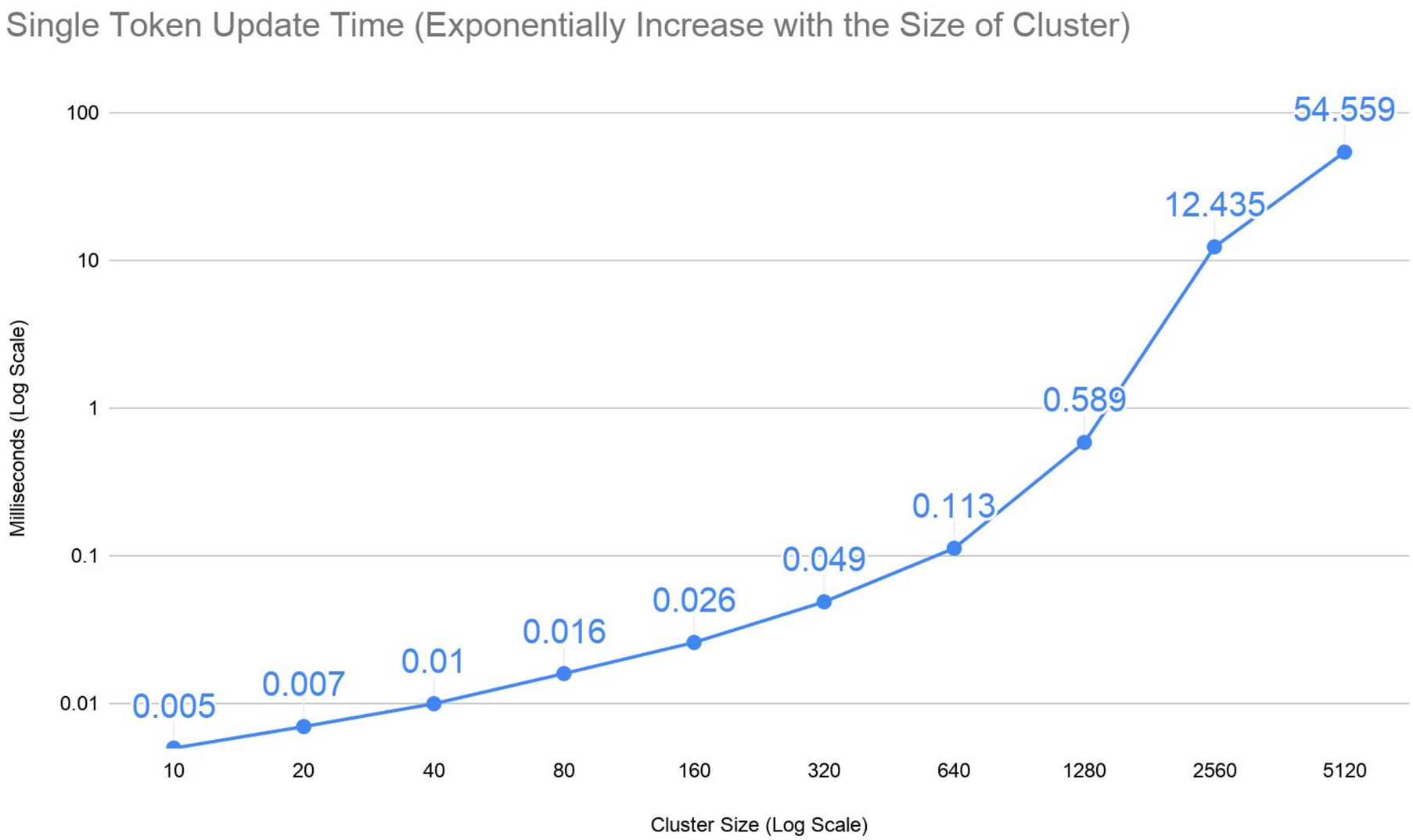
- Consistent Hashing Ring
- Gossip Protocol based Token Management
  - Adding New Node
  - Decommission Existing Node
  - Replacing Down Node
  - etc.
- Eventual Consistency



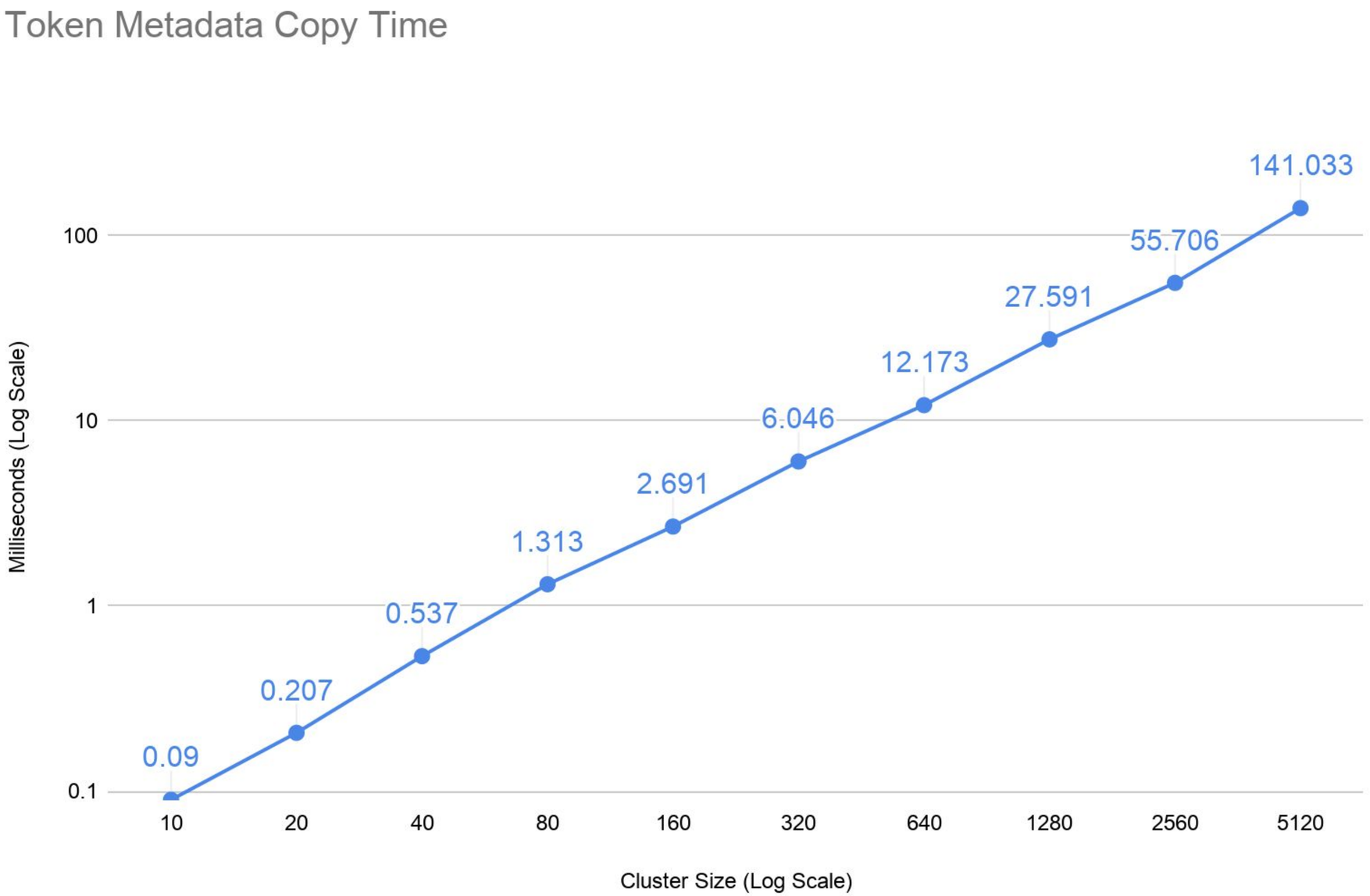
Challenges

- Large Number of Tokens
  - Expensive Token Operations
  - Inefficient Replicas Computation
- Gossip Convergence Time
- Strong Consistency

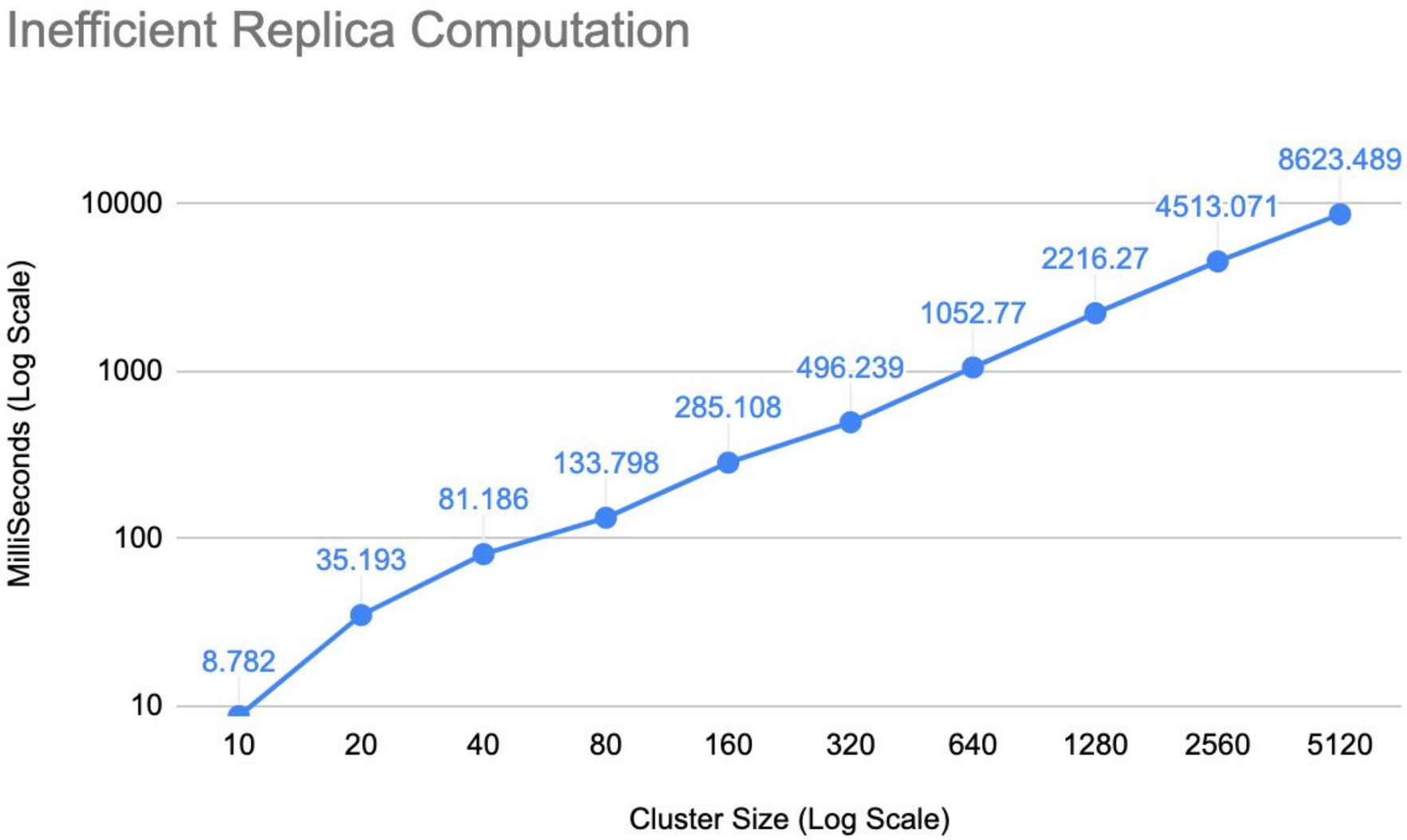
Expensive Token Metadata Update



Expensive Token Metadata Copy

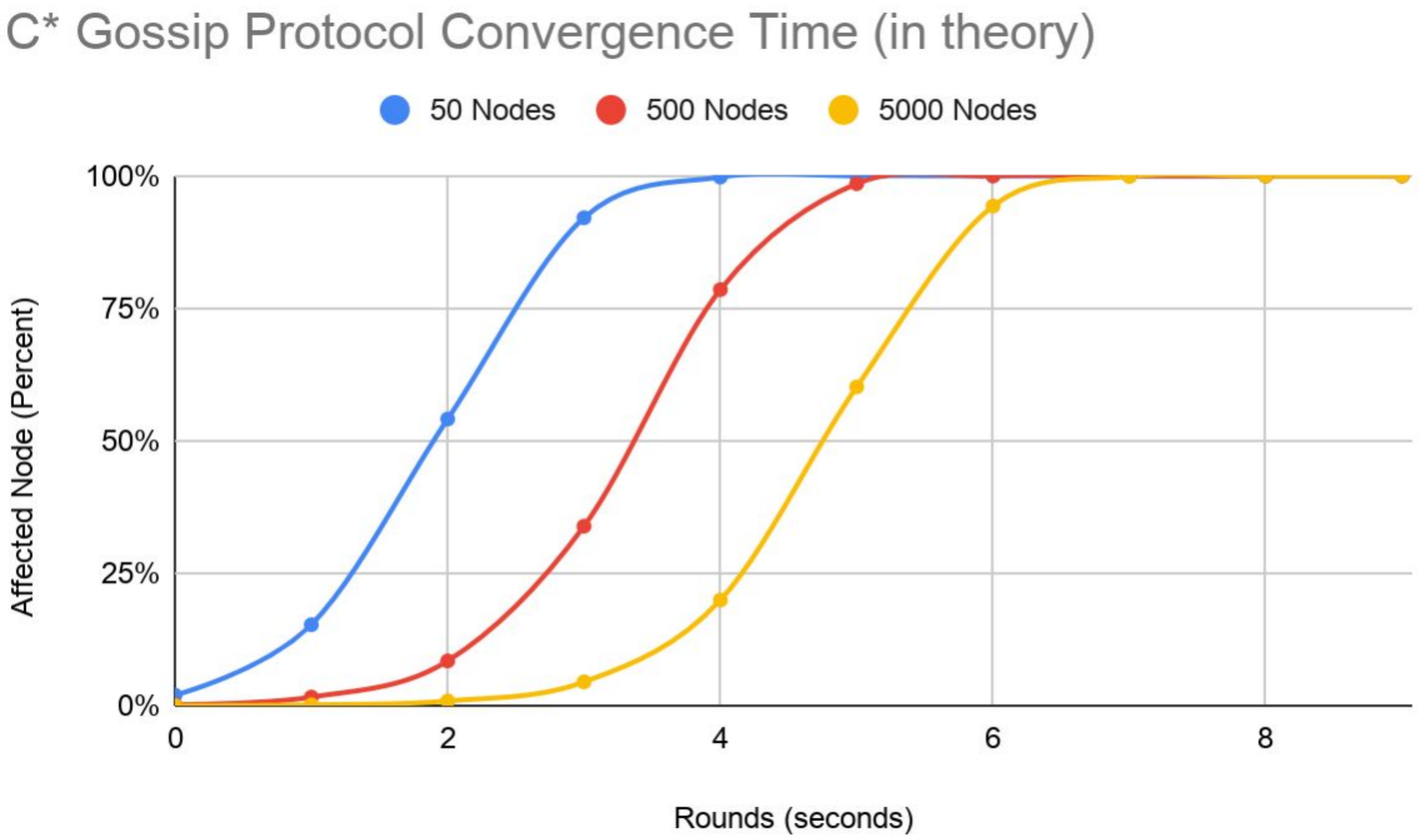


Inefficient Replicas Computation

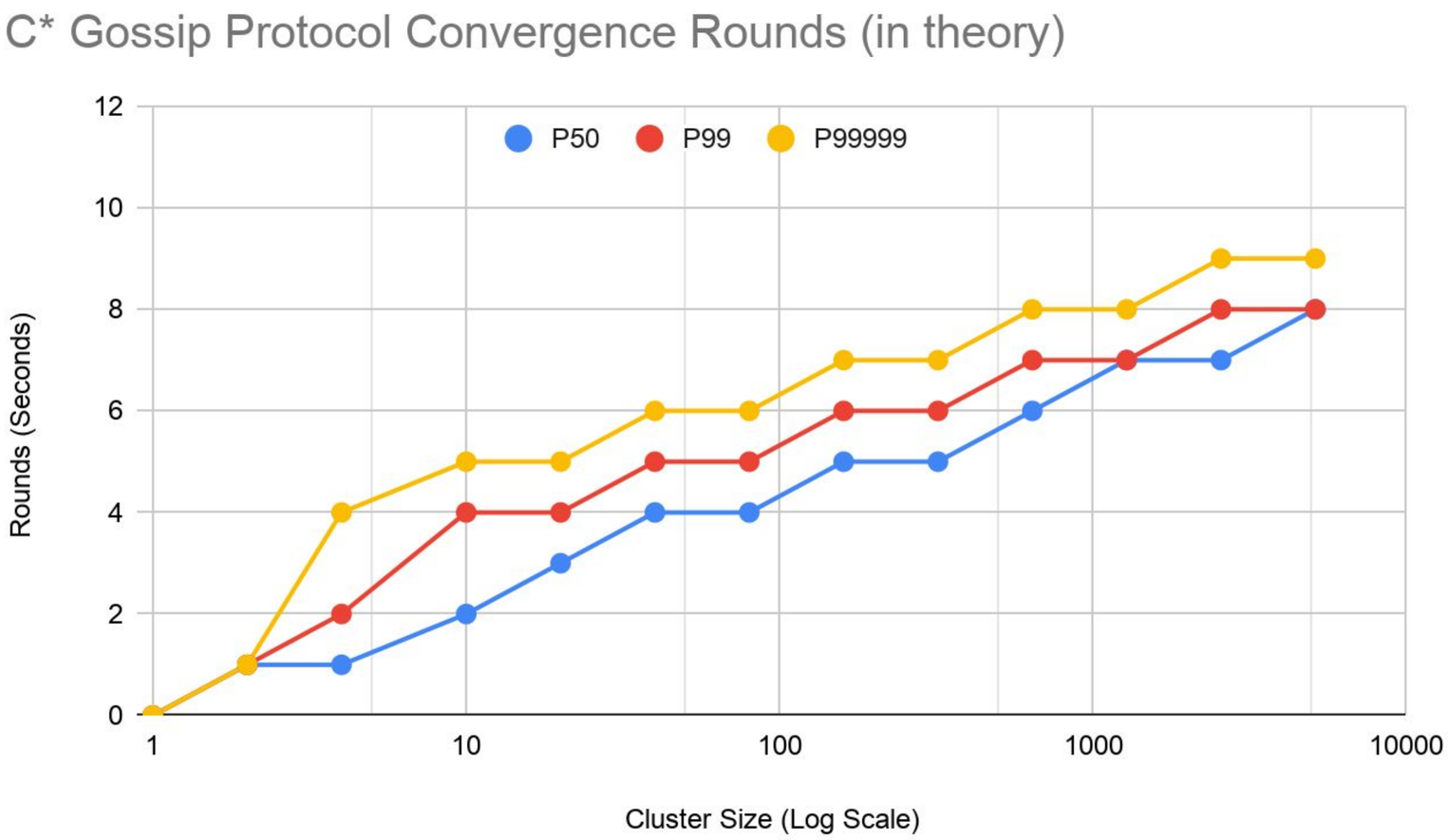


Gossip Convergence Time

- Convergence Time increases with the size of the cluster
- Worsened by:
  - JAVA GC
  - Network Delay (Multiple Continents)
  - Message Processing Time
- Cassandra ring delay: 30 seconds
  - `-Dcassandra.ring_delay_ms`

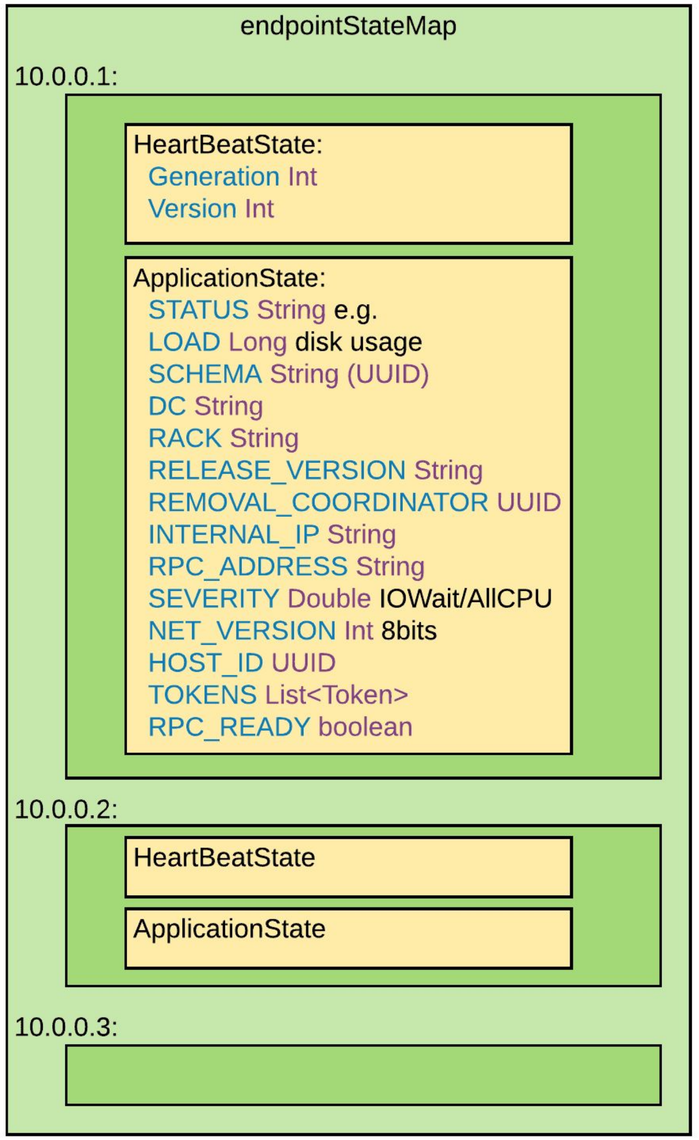


Gossip Convergence Time



Shared Gossip Message and Stage

- Gossip Message is Used by Others:
  - HeartBeat (Failure Detection)
  - Status
  - Load, Severity
  - Tokens
- Message Increase with the Size of Cluster



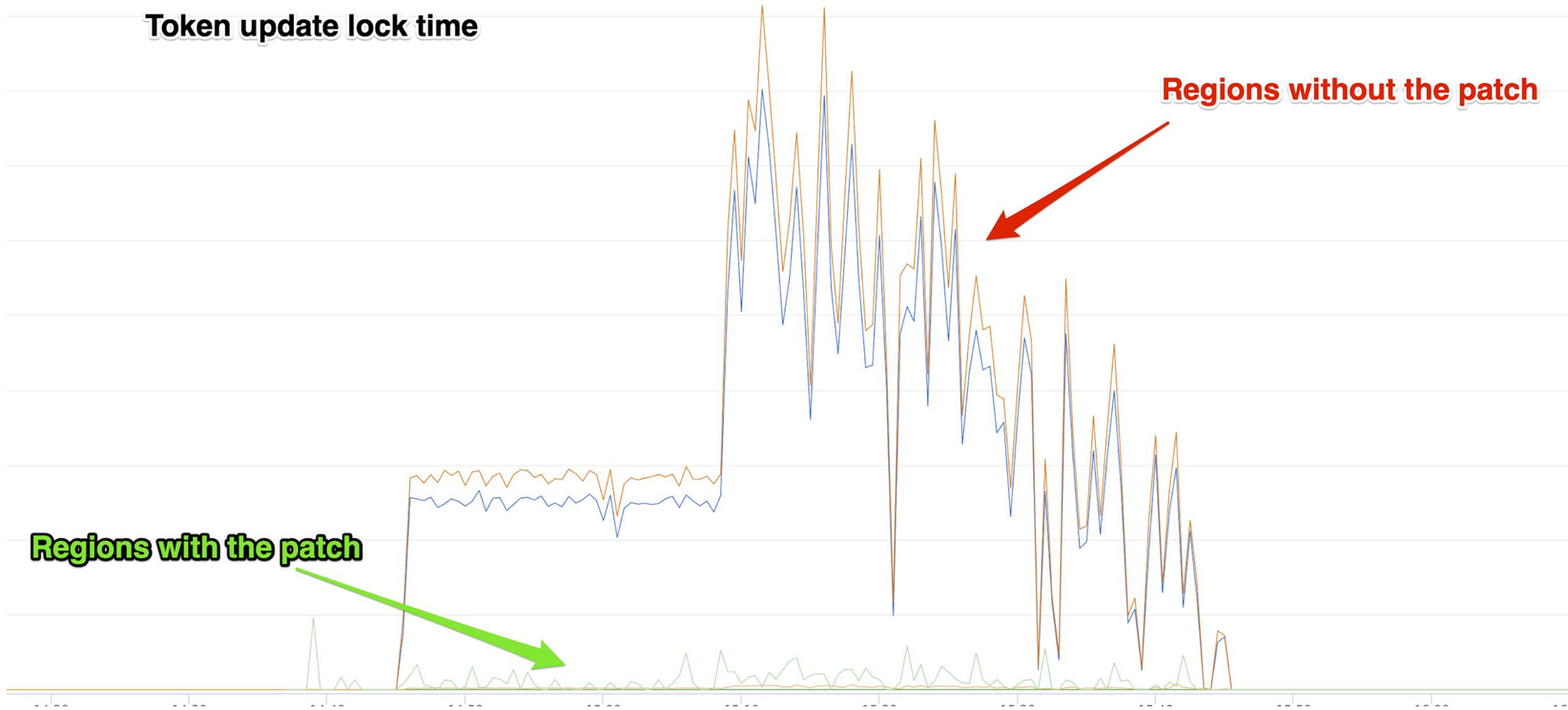
Improvements

More Efficient Token Update

- Improve Token Metadata Update Speed
  - CASSANDRA-14660: Improve Token Metadata cache populating performance for large cluster
  - CASSANDRA-15097: Avoid updating unchanged gossip state
  - CASSANDRA-15098: Avoid token metadata copy during update
  - CASSANDRA-15133: Node restart causes unnecessary token metadata update
  - CASSANDRA-15290: Avoid token cache invalidation for removing proxy node
  - CASSANDRA-15291: Batch the token metadata update to improve the speed
- Improve Token Ownership Calculation
  - CASSANDRA-15141: Faster token ownership calculation for NetworkTopologyStrategy

Token Update Lock

- Token Metadata Update is Faster
- No Unnecessary Token Update
- 10x Cassandra start up time for Large cluster



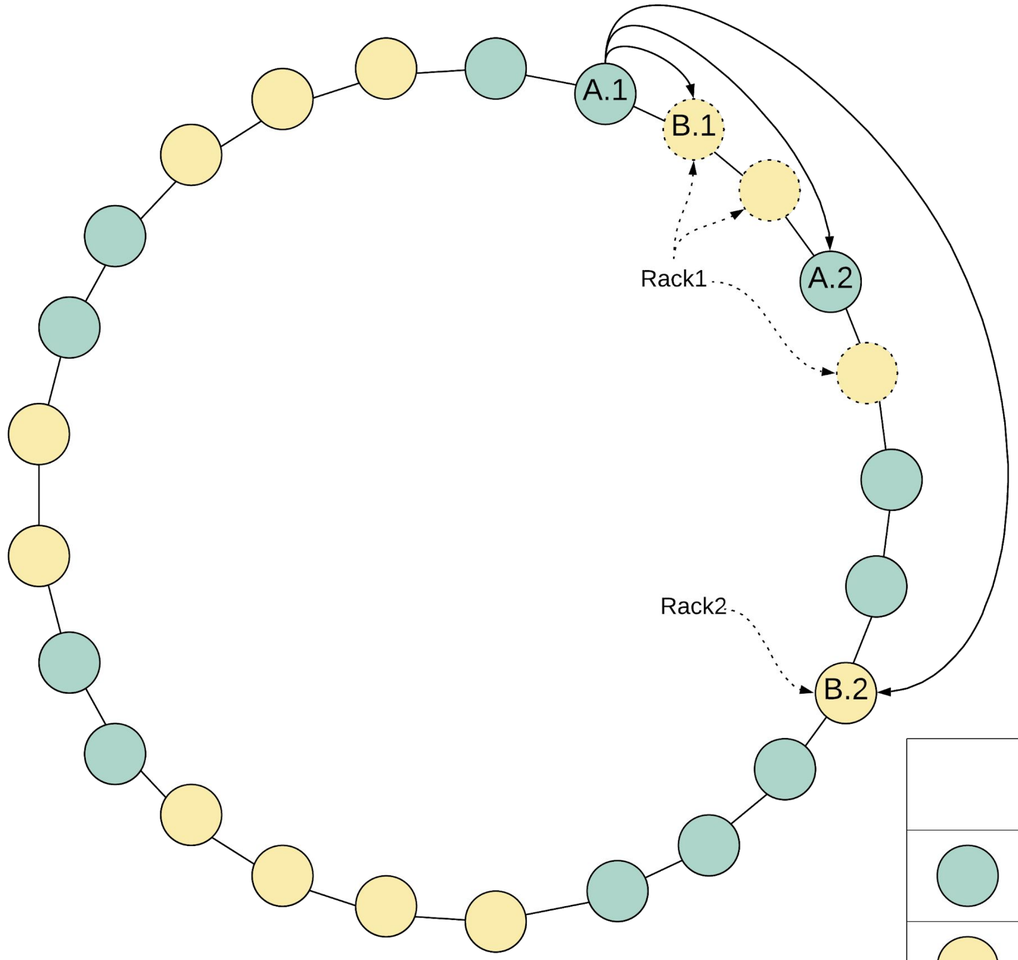
Faster Token Replication Computation

CASSANDRA-15141

- Token Replication Computation:

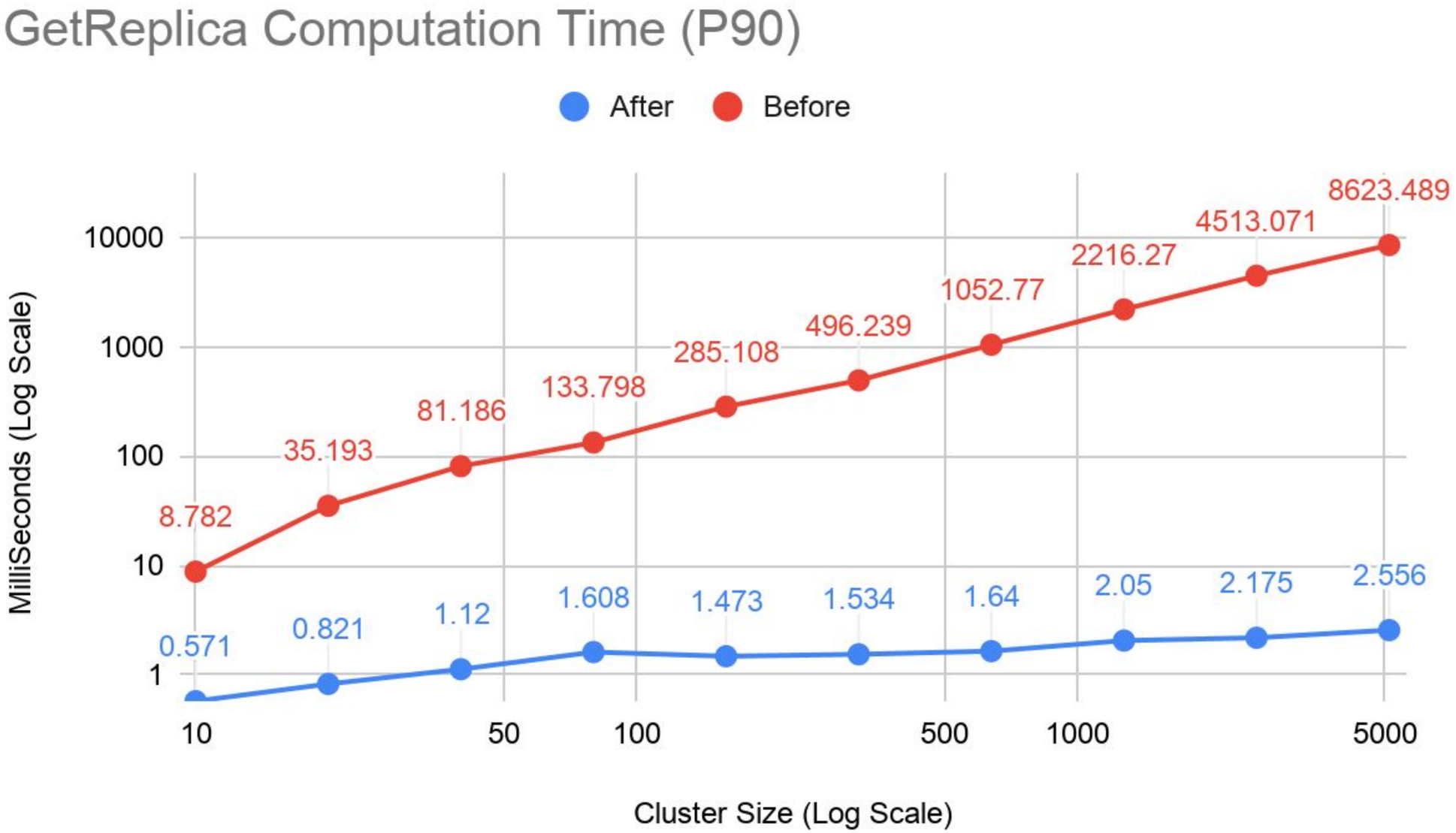
  Get Replicas on One Node
- Needed to Recalculate for Token

  Change:
  - Based on Keyspace Replication Factor
  - Region aware
  - Rack aware



	DataCenter	Replication
	DC:A	2
	DC:B	2

Replication Computation Time



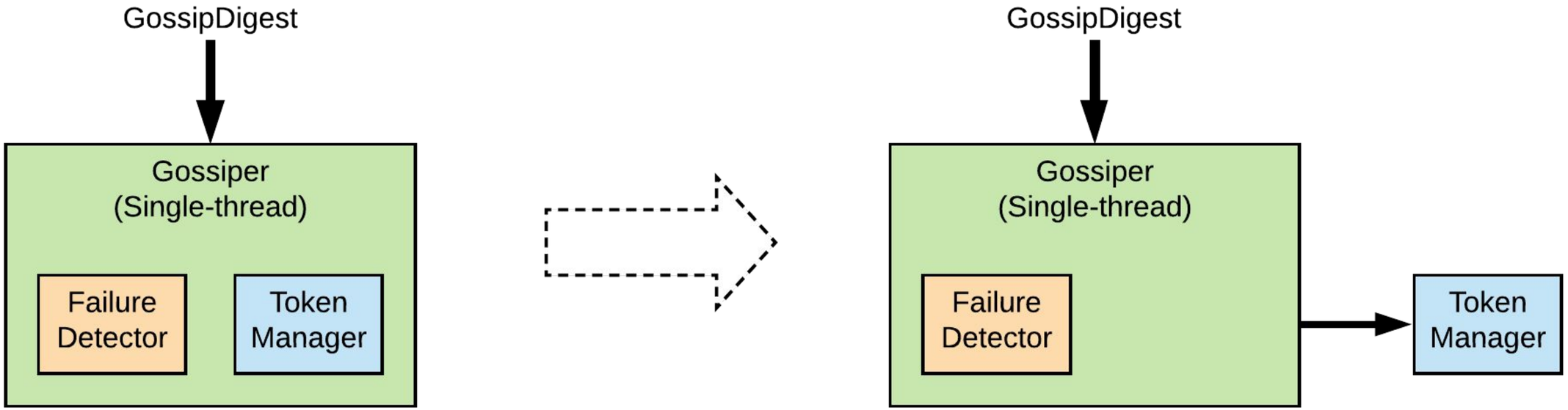
Future Work

Future Work

- Separate Token Manager and Failure Detector
- Consistent Token Management
- Pre-Allocate Tokens
- Pluggable Token Management

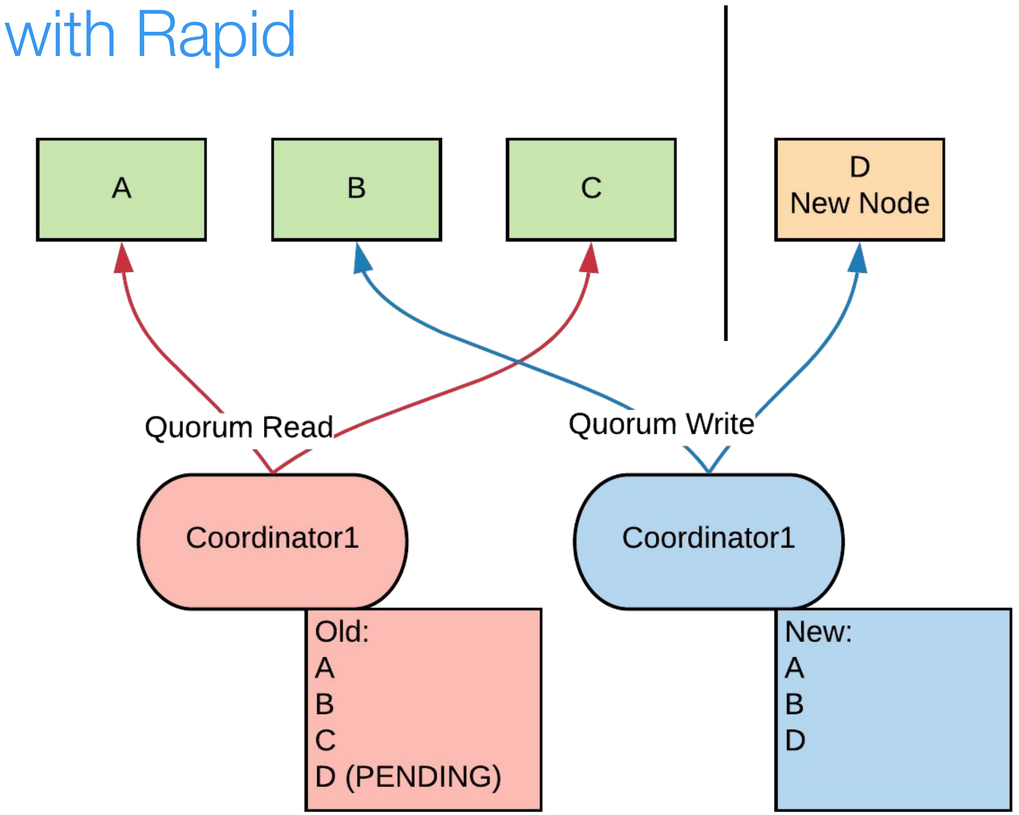
Separate Token Manager and Failure Detector

- Separate 2 Critical System
- Async Update Token
- Rewrite Token Metadata: CASSANDRA-6061



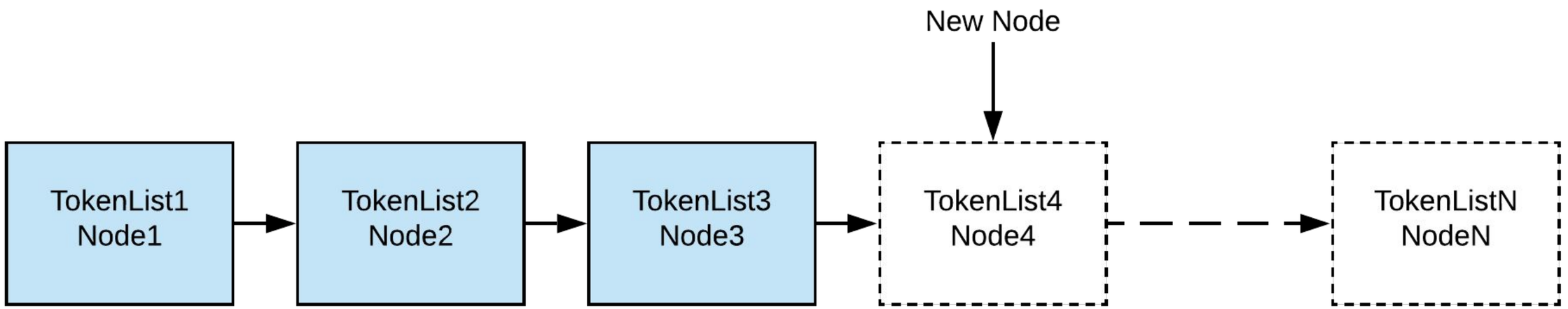
Consistent Token Management

- CASSANDRA-9667: Strongly Consistent Membership and Ownership
  - Token change can only be committed if majority of Nodes agrees (with a consensus transaction like PAXOS)
  - Stable and Consistent Membership at Scale with Rapid
- Data ownership verify on local read/write path



Pre-Allocate Tokens

- Generate Fixed Tokens for N Number Nodes (Even Before Cluster Creation)
- Even better for replication aware token allocation, as token imbalance issue if randomly decommission a node
- Other Token Computation Could also be Pre-Computed



Pluggable Token Management

- Integrate with Other Consistent Shard Management System:
  - ZooKeeper
  - ShardManager

Summary

1 Challenges

2 Improvements

3 Future Work