Praktikum: Cloud Databases SoSe 2022

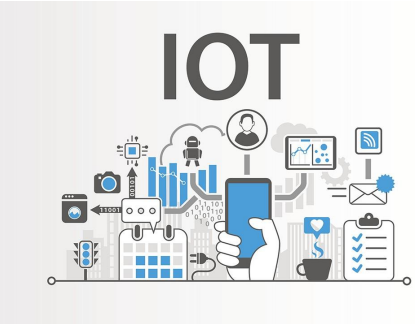
Group 5
Saqib Ali Khan, Syed Saad bin Shameem and Mahdi Mohebali

Agenda

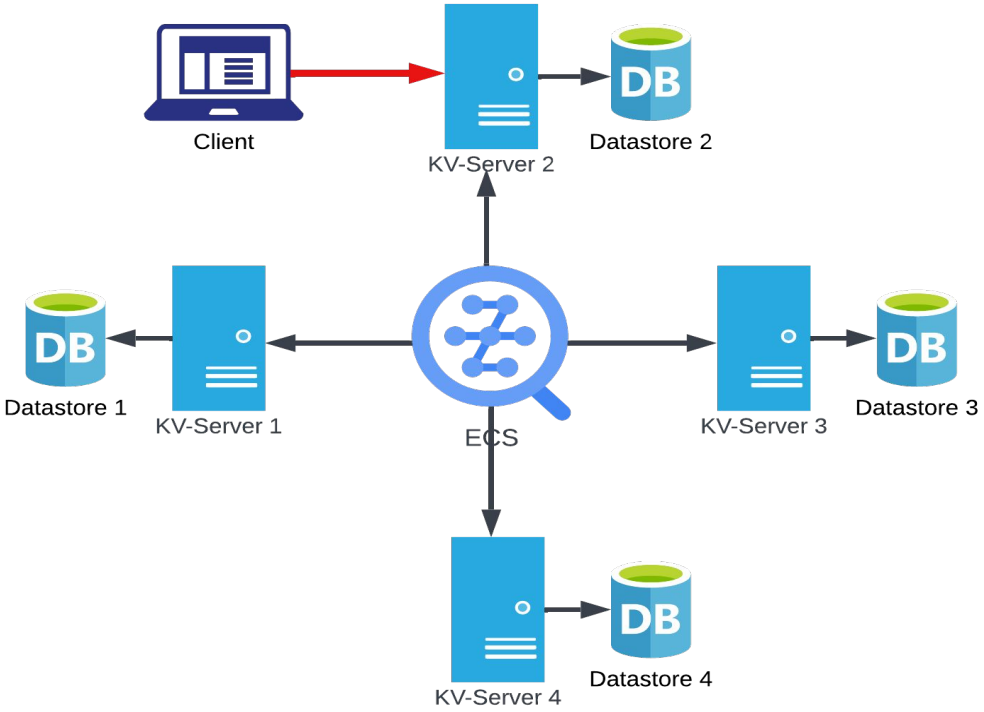
- Motivation
- Critical Components
- Design Decisions
  - Clustered B+ Trees
  - Stronger Consistency
  - Leader Election (p2p)
- Dataset
- Benchmarking

Introduction

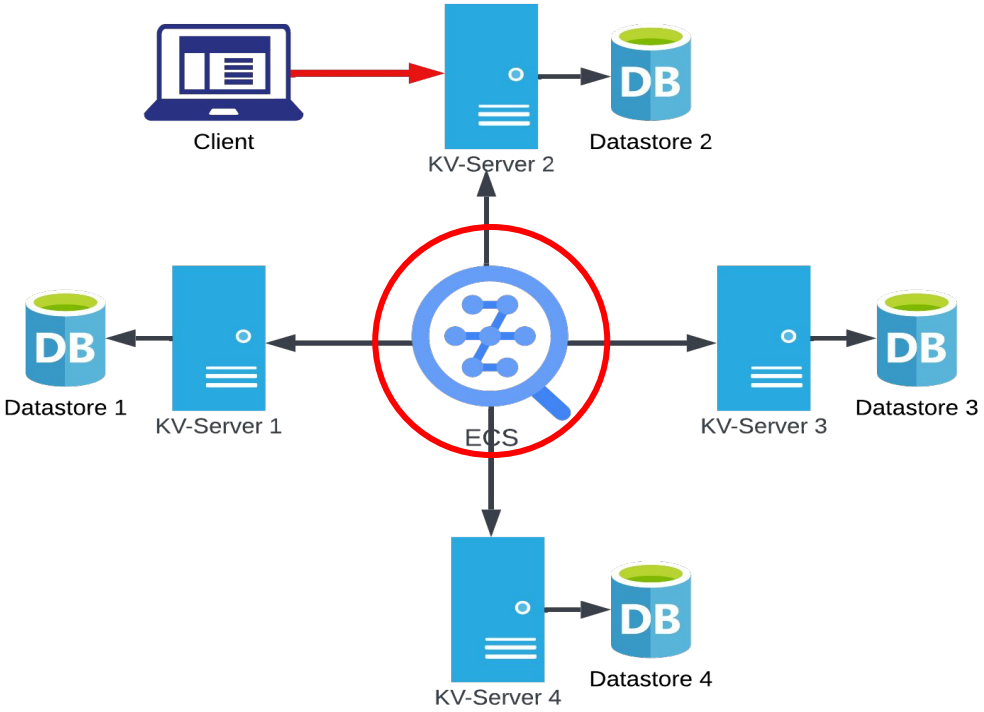
- Data storage landscape has evolved.
- Big data requires massive scale.
- Requires continuous data processing.
- Goals:
  - Scaled-out
  - Fault-tolerant
  - Reliable



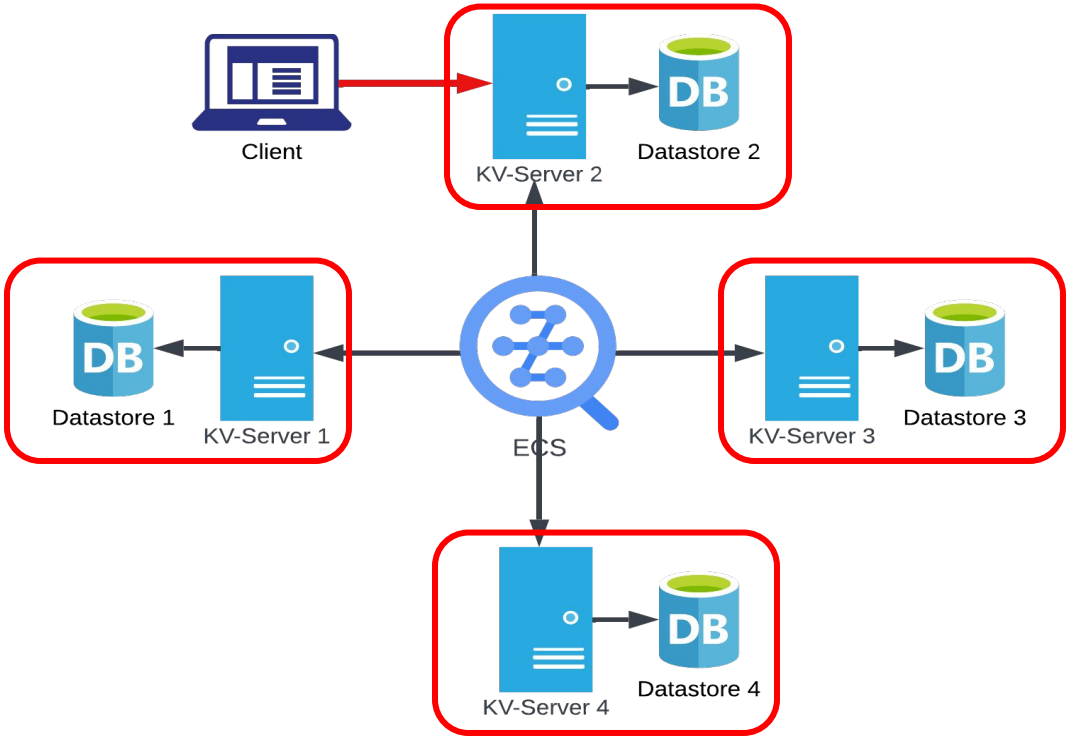
System Components



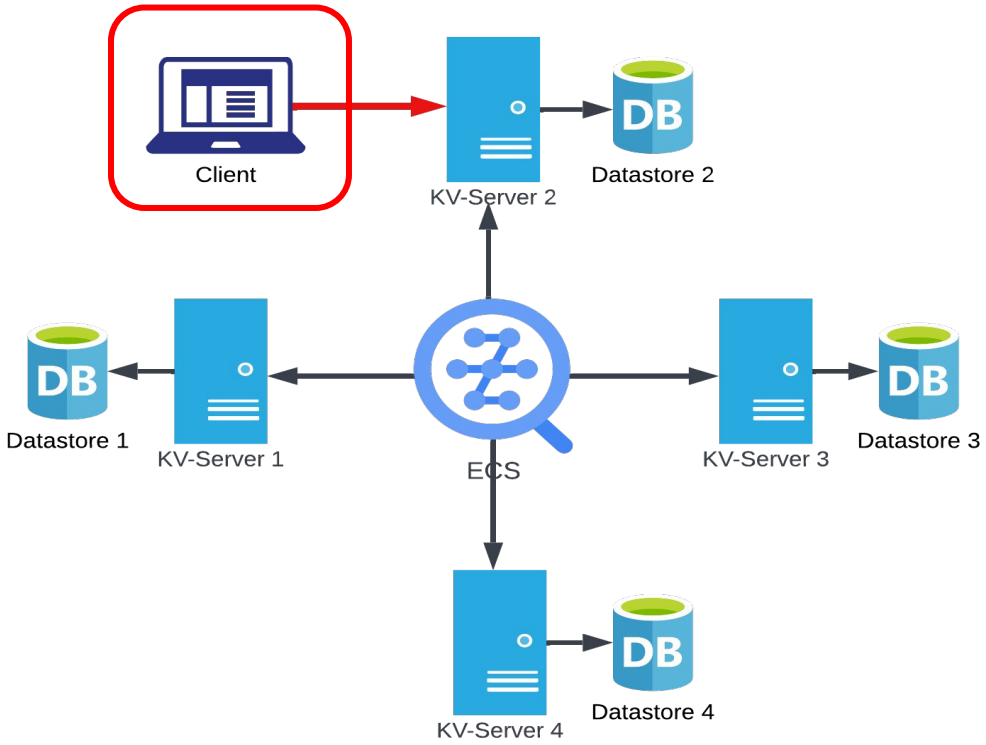
System Components



System Components

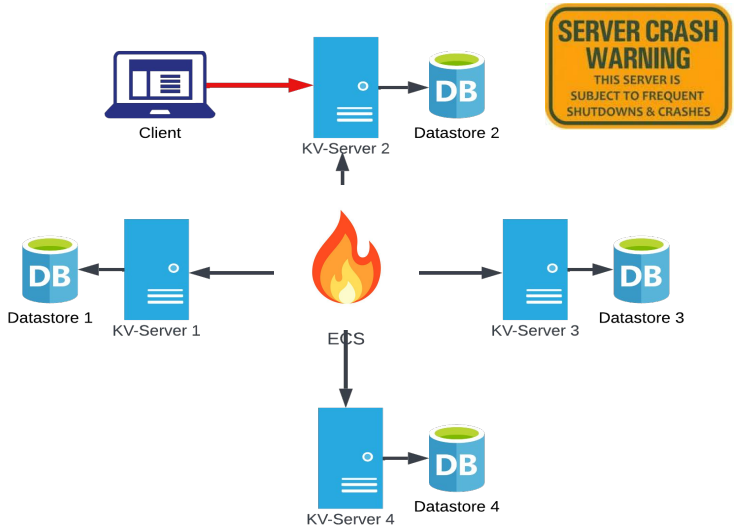


System Components



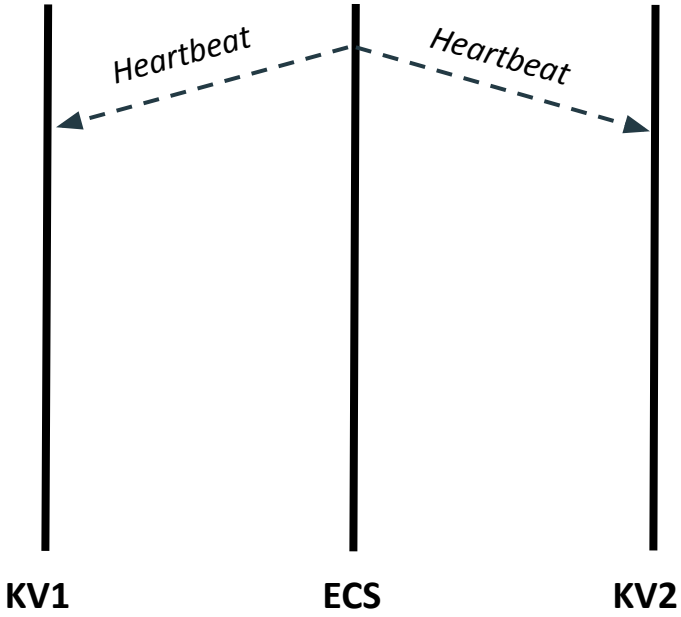
ECS - Leader Election

- ECS contains ring topology
  - Single Point of Failure
  - Byzantine Failures
- Objective
  - Fault tolerance
- Solution
  - P2P



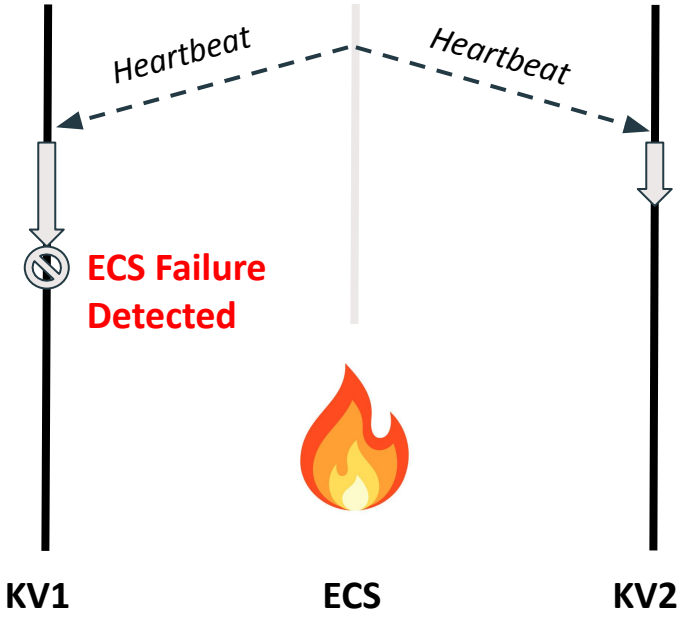
Our Leader Election Protocol

- In normal running mode, ECS sends metadata update to every node in the system every 1 sec.



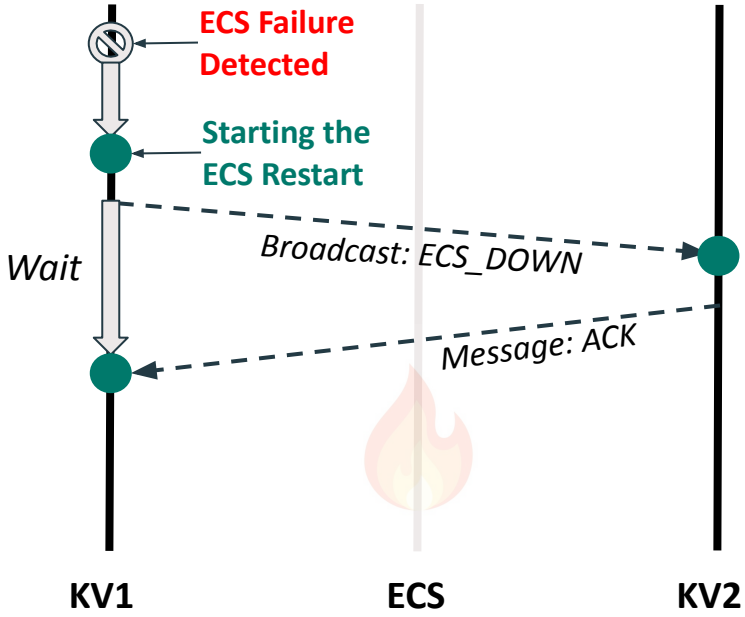
Our Leader Election Protocol

If the ECS fails, the KVs will not receive metadata update (heartbeat) and hence will detect the failure.



Leader Election Algorithm

After the ECS failure is detected by any node in the cluster, they can initiate the protocol.



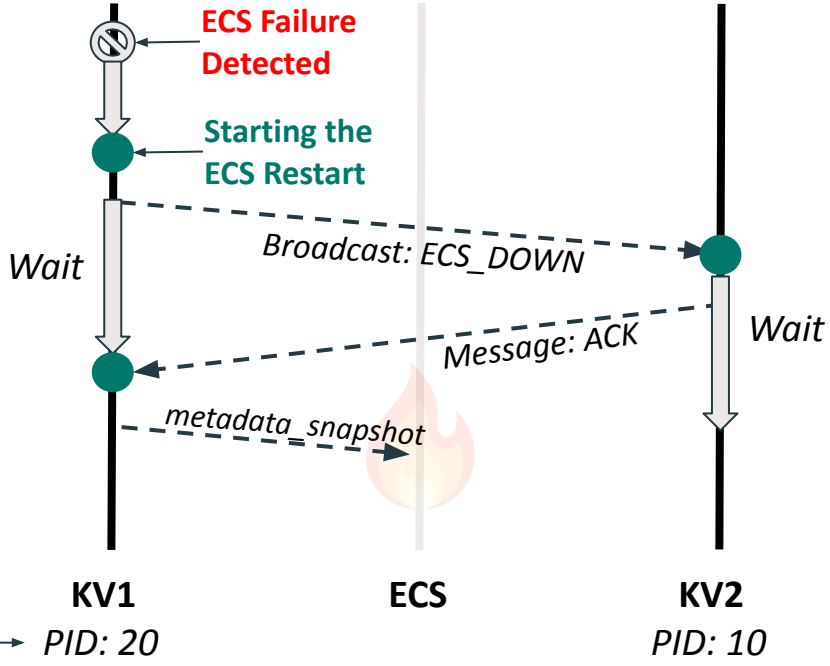
Leader Election Algorithm

If current_node.PID > ALL:

Restart the ECS locally

Else:

Ignore and wait for the ECS startup by the node with highest PID using the local available metadata snapshot



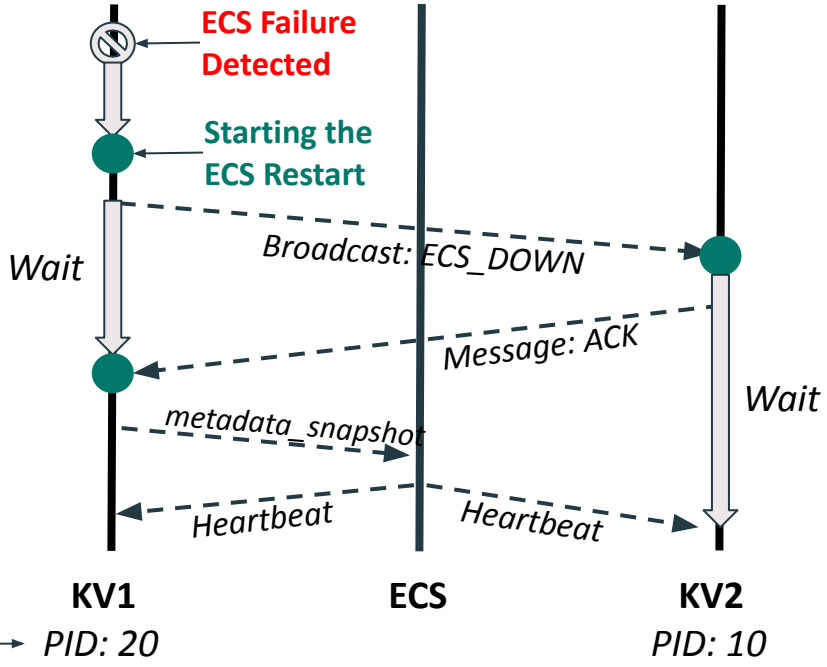
Leader Election Algorithm

If current_node.PID > ALL:

Restart the ECS locally

Else:

Ignore and wait for the ECS startup by the node with highest PID using the local available metadata snapshot



Special Thanks to….

Chang and Roberts

https://en.wikipedia.org/wiki/Chang_and_Roberts_algorithm

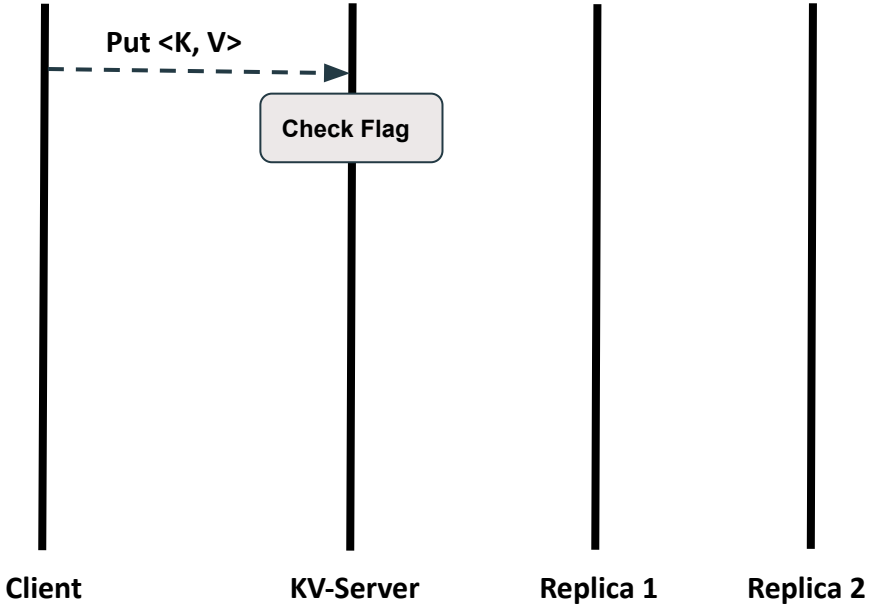
Stronger Data Consistency

- FIFO Consistency is stronger than eventual consistency.
- **Strategy**:
  - Allows all processes to view the operations of a single process in the same order that they were issued by that process.
- Makes it less likely for clients to infer a contradiction directly or indirectly by interacting with the datastore.

Stronger Consistency

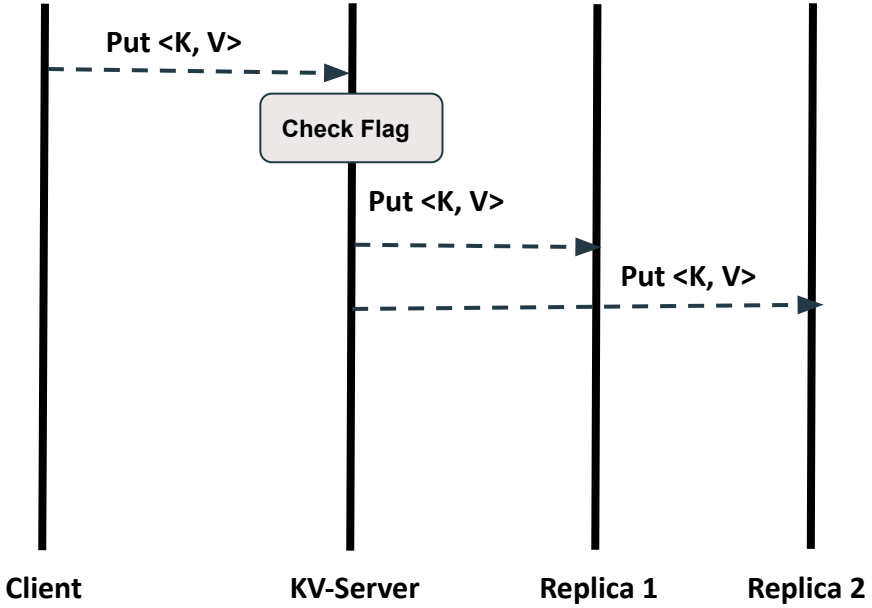
Client sends put request to main kv-server.

Server checks if the flag for stronger consistency is set.



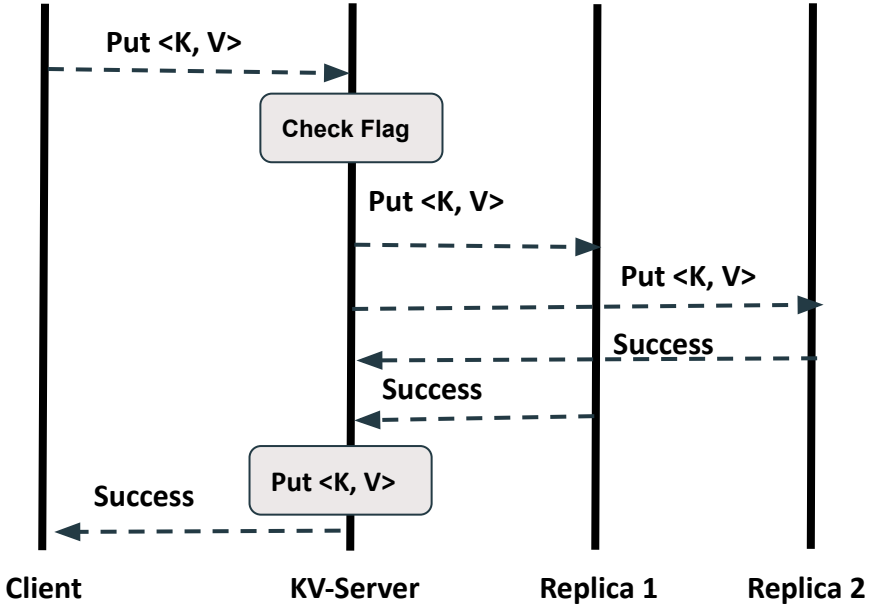
Stronger Consistency

If the flag is set, the server sends the request to it's replicas through WAL, then waits for WAL, which acts as a FIFO list, to sync, so that the requests get processed at the replicas.

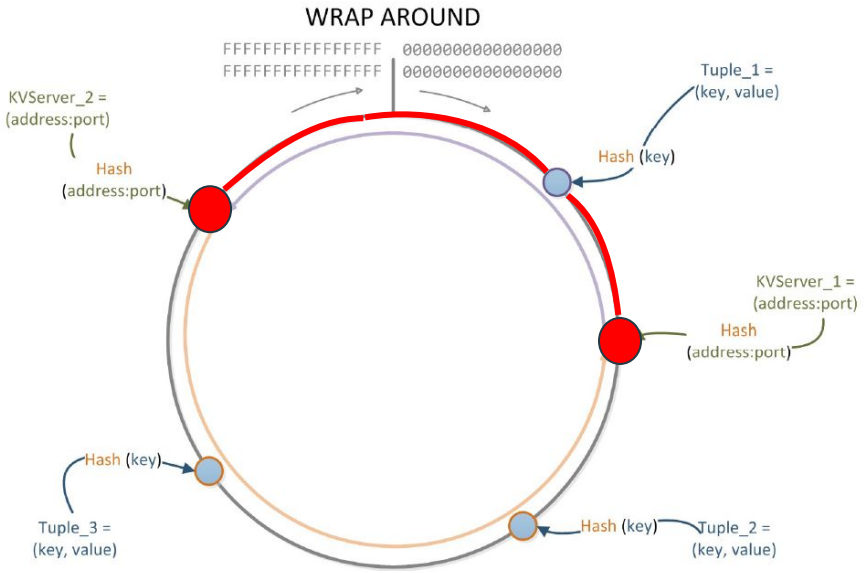


Stronger Consistency

After the put request is processed at the replicas, the servers runs its own put function and then returns a message to the client.



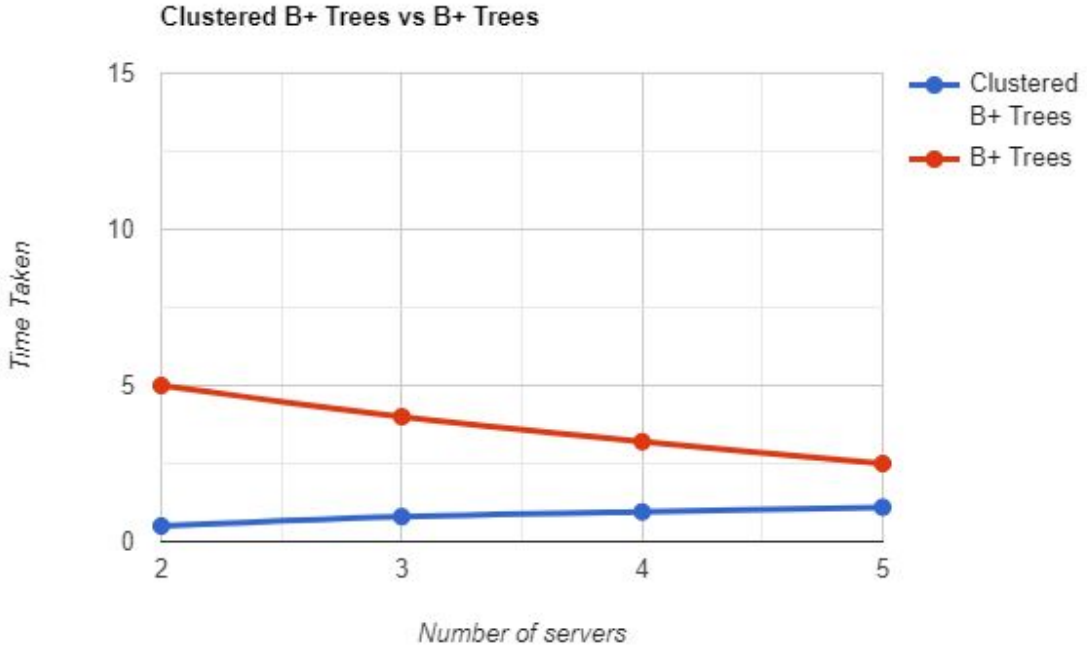
Optimizing Repartitioning/Re-replication



Optimizing for short-range lookups

- For the underlying datastore, we use clustered B+ Trees.
- Key-Value pairs are sorted on disk
- Fast retrieval for keyranges, useful when replicating or repartitioning.
- The leaf nodes of B+ trees are linked, so doing a full scan of all objects in a tree requires just one linear pass through all the leaf nodes

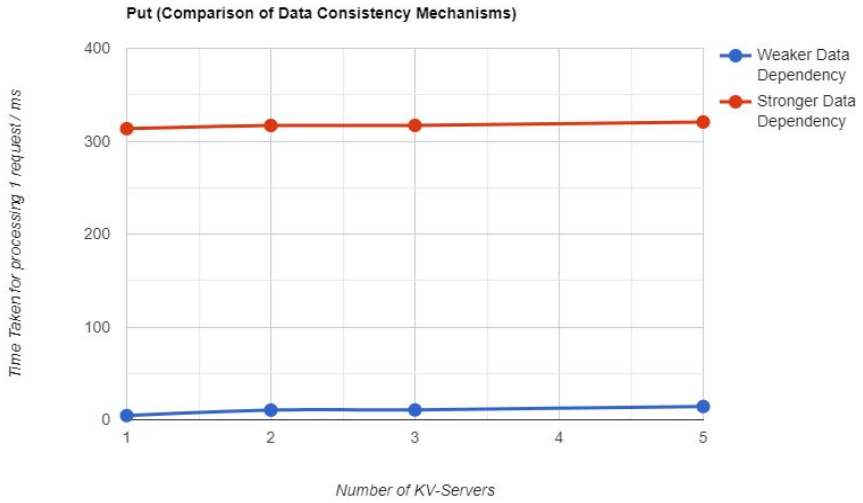
Effect of Clustered B+ Trees

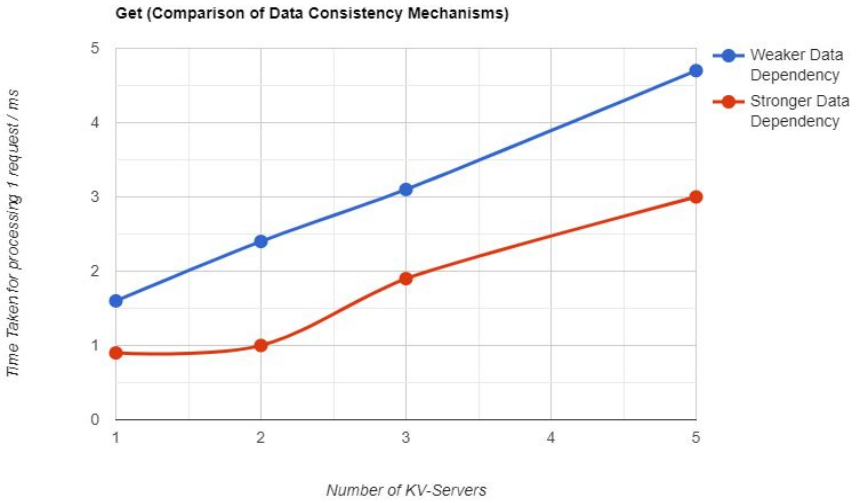


Benchmarking

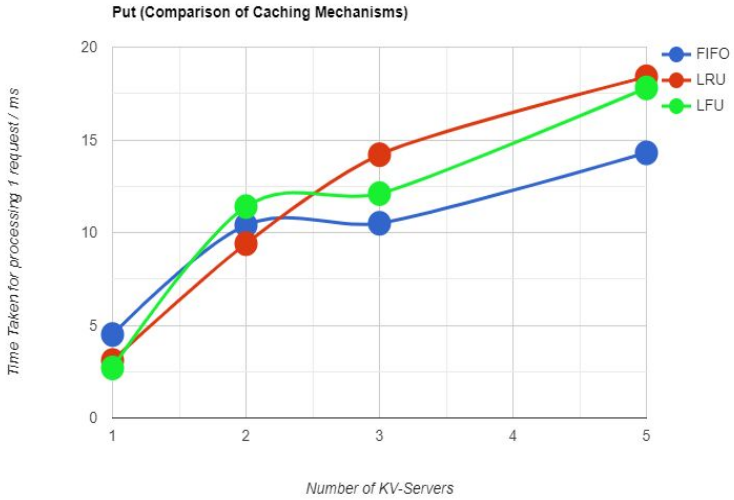
- Enron Email dataset (https://www.cs.cmu.edu/~enron/)
- i5-7200 ~ 2.4 GHz, 8GB RAM, HDD
- Keys are the filenames, and values are a part of the email file data.
- Number of KV-Servers: **1, 2, 3, 4, 5**
- Caching Strategies: **FIFO, LFU, LRU**
- Consistency Models: **Eventual Consistency**, **FIFO Data Consistency**

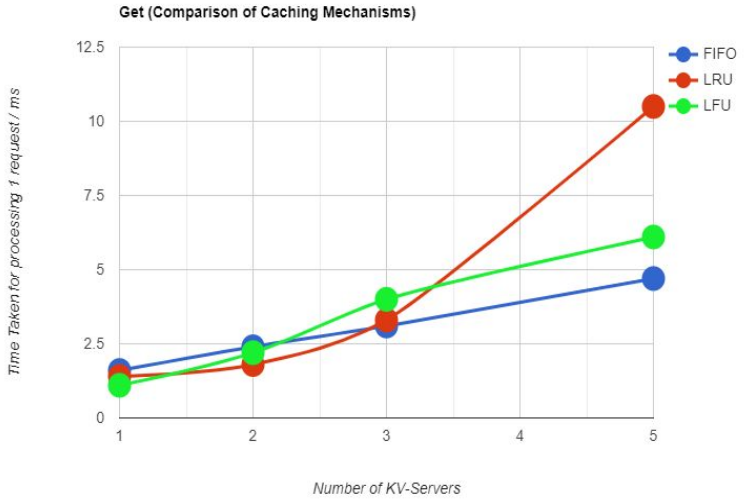
Evaluation: Effect of Consistency on PUT and GET





Evaluation: Effect of Caching strategies on PUT and GET





Summary

Our System Offers:

- Stronger consistency
- No Single Point of Failure (p2p leader election)
- Efficient data storage structure.

Demo

Thank You !