C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection

Lalith Suresh, et al, NSDI 2015

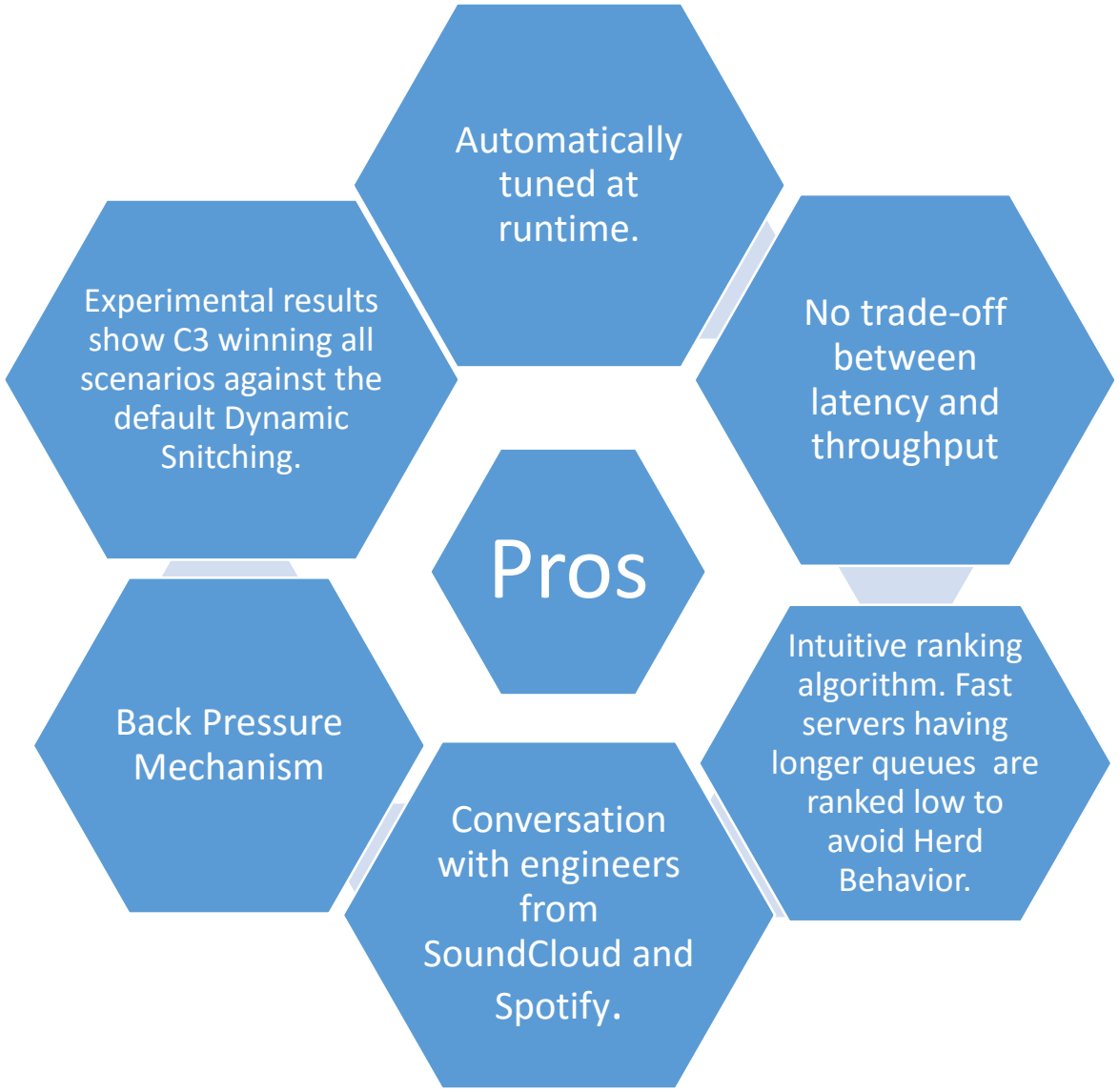
Scribe by

- Piyush Shrivastava

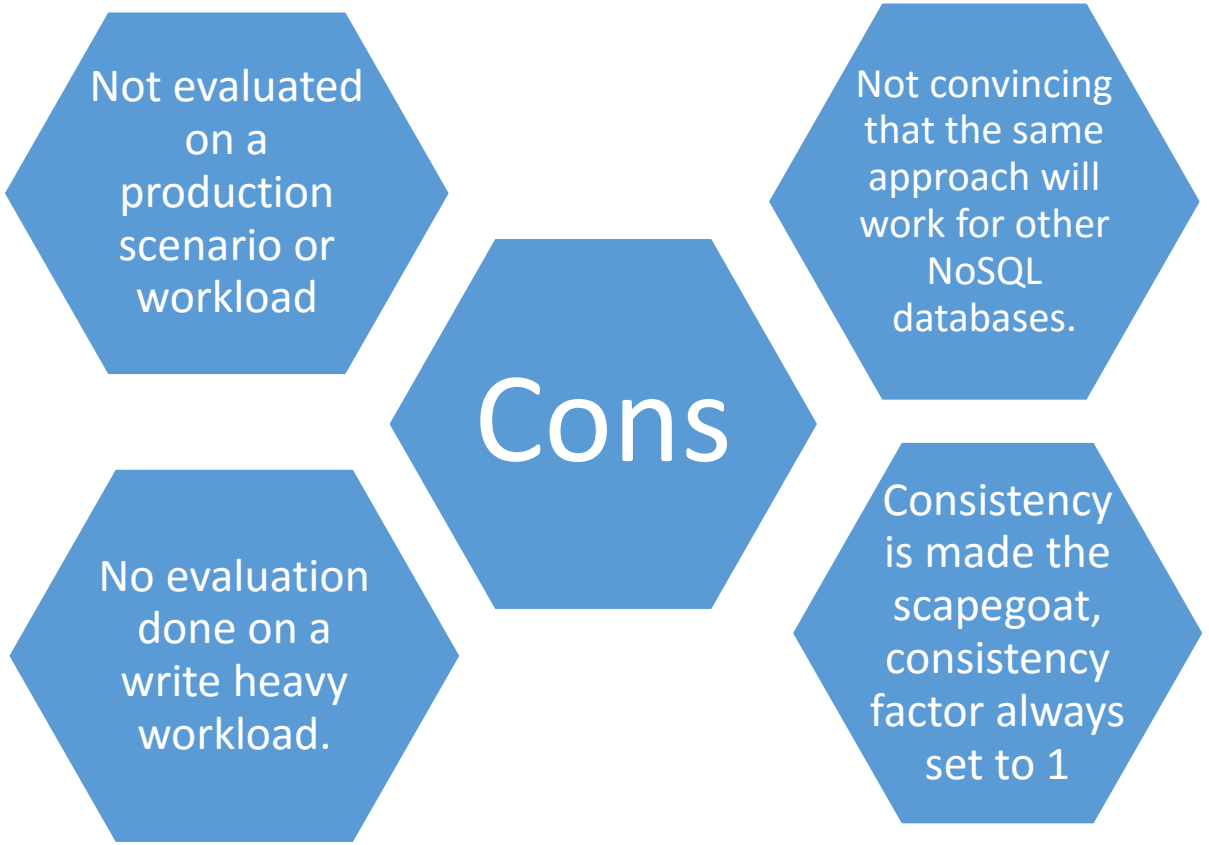
Summary

- Motivation: To minimize tail latency in a distributed data store.
- Authors introduce C3 - a system for cutting tail latency in cloud data stores by adaptive replica selection
- Implemented on top of Cassandra.
- Two concepts introduced:
  - Replica Ranking algorithm
  - Distributed Rate Control mechanism

Pros


Pros
Automatically tuned at runtime.
No trade-off between latency and throughput
Intuitive ranking algorithm. Fast servers having longer queues are ranked low to avoid Herd Behavior.
Conversation with engineers from SoundCloud and Spotify.
Back Pressure Mechanism
Experimental results show C3 winning all scenarios against the default Dynamic Snitching.

Cons


Not evaluated on a production scenario or workload
Not convincing that the same approach will work for other NoSQL databases.
Cons
No evaluation done on a write heavy workload.
Consistency is made the scapegoat, consistency factor always set to 1

Discussion/Questions

- The system model has consistency level set to 1 for all scenarios. Is that a fair assumption to make?
- Can C3 be easily extended to other NoSQL systems?
- Experimental evaluation does not include a write/update heavy workload. Why?

Discussion/Questions (Most probable answers)

- The system model has consistency level set to 1 for all scenarios. Is that a fair assumption to make?

  ** Okay for read heavy workload. But for a other work loads, such an assumption can surely invite the problem of stale reads.

- Can C3 be easily extended to other NoSQL systems?

  ** Authors have mentioned this in the future work section. But since there are major design differences between Cassandra and MongoDB, the 'porting task' (as mentioned by the authors) would not be easy.

- Experimental evaluation does not include a write/update heavy workload. Why?

  ** Difficult to answer this. Most likely explanation - Since the authors were in conversation with engineers from SoundCloud and Spotify (which have a read heavy workload), they ignored the write heavy workload in the evaluation.

Thank You!