Samya: Geo-Distributed Data System for High Contention Data Aggregates

Sujaya Maiyya, Ishtiyaque Ahmad, Divyakant Agrawal, Amr El Abbadi
UC Santa Barbara
Today, we are in a world of geo-distributed databases
Today, we are in a world of geo-distributed databases

While a great choice for fault-tolerance and high availability...

Latency can be high, esp for update heavy workloads

Google Spanner commits a txn with avg 17ms and tail 75ms[1]

Today, we are in a world of geo-distributed databases

While a great choice for fault-tolerance and high availability...

Latency can be high, esp for update heavy workloads

i.e., Spanner can commit avg 60tps and tail 13tps

Consider an example: Resource management within a cloud provider

ultraCloud

A max quota limit is set for each resource

Individual teams acquire or release resources via read-write txns
Consider an example: Resource management within a cloud provider

Root node becomes a **hotspot**

60tps becomes a **bottleneck** for large enterprises
Issues with Spanner-like db design

1. Sequential execution
2. Centralized, constant synchronization
3. Underutilized replicas

E.g. tokens of vms available
Issues with Spanner-like db design

1. Sequential execution
2. Centralized, constant synchronization
3. Underutilized replicas

- Manage aggregate data
- Update heavy workload

But low performance due to centralized, sequential execution
Issues with Spanner-like db design

1. Sequential execution
2. Centralized, constant synchronization
3. Underutilized replicas

Our research question:
Design an alternate system to manage simple data types and provides high throughput for update heavy workloads?
Looking back in the literature, we stumble upon many seminal works that answer our question.

O’Neil’s Escrow transactions [1]
Kumar and Stonebreaker [2]
Barbara and García-Molina’s Demarcation protocol [3]
Gustavo and El Abbadi [4]

Looking back in the literature, we stumble upon many seminal works that answer our question.

O’Neil’s Escrow transactions

Kumar and Stonebreaker

Barbara and Garcia-Molina’s Demarcation protocol

Gustavo and El Abbadi

Partition the aggregate data and allow transactions to concurrently update different partitions

Looking back in the literature, we stumble upon many seminal works that answer our question.

But proposed for radically different environments:

- sites are not geo-distributed
- networks are assumed reliable
- results are only simulations

Samya brings the basic idea
– *dis-aggregate the aggregate data to increase concurrency* –
to the modern context of cloud and geo-distributed dbs
Clients communicate with closest sites by sending *acquire* or *release* tokens request.
Sites serve requests locally and update tokens left.
But what if I want more than 200 tokens??
But what if I want more than 200 tokens?

Each site stores disaggregated data, e.g., tokens of vms available locally.

Clients communicate with closest sites.

Sites serve requests locally and update tokens left by sending acquire or release tokens request.

Avantan: a consensus protocol to agree on the global token availability and to **redistribute** tokens.
1. Avantan reaches agreement on *available tokens* – not on a client provided value
2. Avantan does *not* require a majority for consensus
1. Avantan reaches agreement on available tokens – not on a client provided value
2. Avantan does not require a majority for consensus
1. Avantan reaches agreement on *available tokens* – not on a client provided value
2. Avantan does *not* require a majority for consensus
But redistributing *after* a client sends request can cause lot of delay.

**Demand predictions using machine learning and deep learning to the rescue!!**

- Use analytical past resource demand data to predict future demands
- When predicted demand increases, trigger *proactive* redistributions
- Execute Avantam and borrow tokens from sites with decreasing demand
Evaluation setup

• **Servers/Clients**: GCP n1-standard VMs

• **Baselines**: Demarcation/Escrow, CockroachDB (Spanner-like db)

• **Dataset**: VM workload dataset by Microsoft Azure [1], inherently predictable workload

• **Prediction method**: Neural Networks (LSTMs)
Performance analysis of Samya

Samya commits \textbf{16x} to \textbf{18x} more transactions than CockroachDB

Although redistributions are expensive, redistributions increases Samya’s throughput by \textbf{14%}

Samya performs about \textbf{1.4x} better with predictions

If app. workload has \textbf{less} than 35% writes, Spanner-like DB performs better than Samya
Summary

• Samya: a data system for high-contention aggregate data

• Avantan is a novel consensus protocol used for token redistribution that does not require a majority

• Dis-aggregation and executing Avantan allows Samya to commit 16x to 18x more transactions than a Spanner-like database

• Redistributions and demand predictions significantly increases Samya’s performance