ROTATIONAL

Smart Global Replication
Using Reinforcement Learning

Benjamin Bengfort

KubeCon + CloudNativeCon NA 2023

Motivating Use Case

Blockchain Virtual Assets & Cryptocurrency are being regulated by the **Travel Rule**.

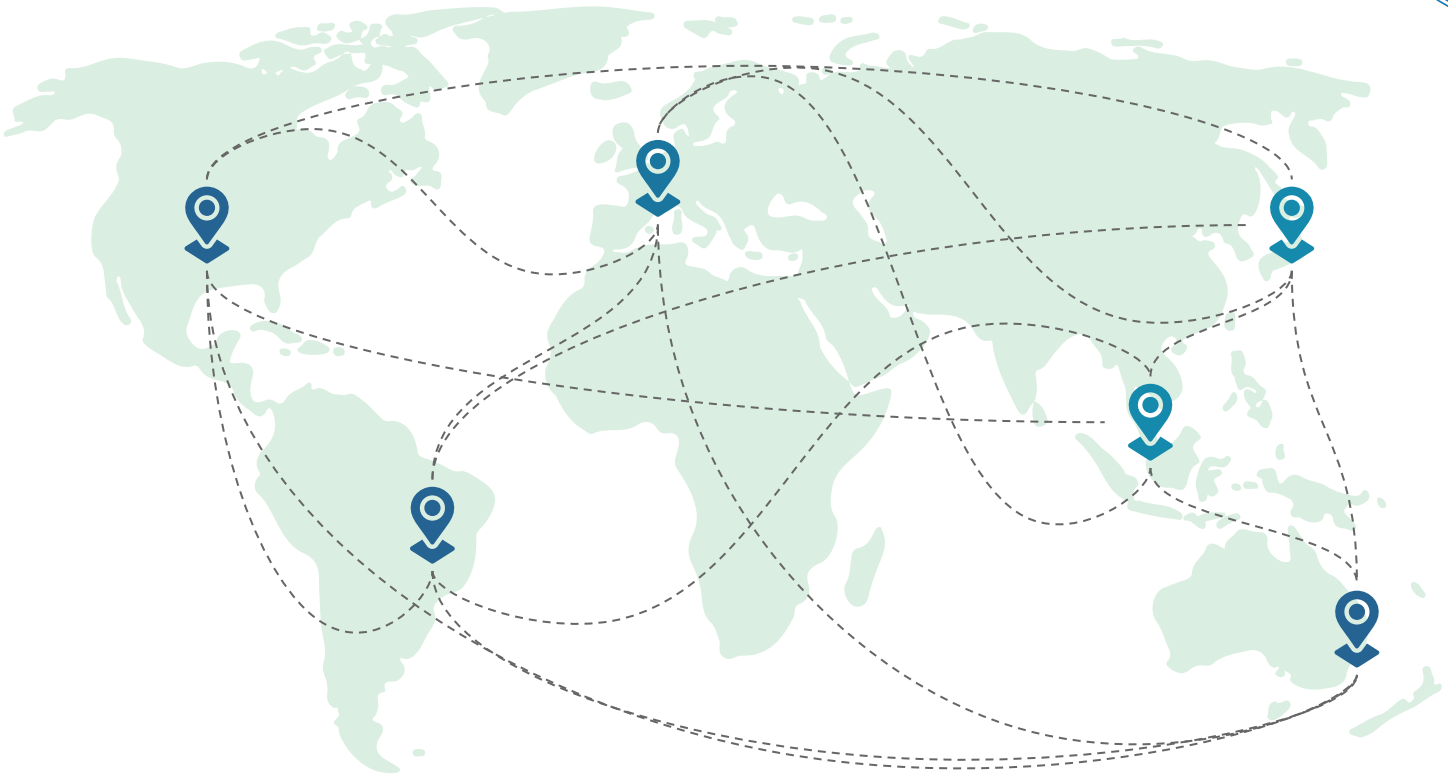
VASPs must exchange PII and perform sanctions checks before **cross-border** transactions for AML and anti-terrorism.

Motivating Use Case

TRISA: Travel Rule Information Sharing Architecture: uses a certificate authority and mTLS to create a peer-to-peer network for information sharing.

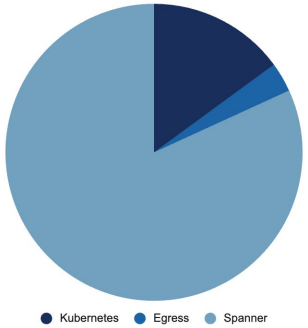
TRISA is an open source, **non-profit** professional working group.

ROTATIONAL



Global Directory Service: Public Key Infrastructure & Matchmaking

ROTATIONAL



USD $8,570

Estimated Monthly Costs with Spanner

What are our actual data storage requirements?

Should you build your own distributed database?

(talk to me after ...)

Do we really need a strong consistency model with distributed SQL? Not really.

Heavy read workload: ~30 reads per second, 24 hours a day.

Write workload: 5-250 writes to 5-11 objects over the course of 14-20 days in **2 regions** (admin and client region).

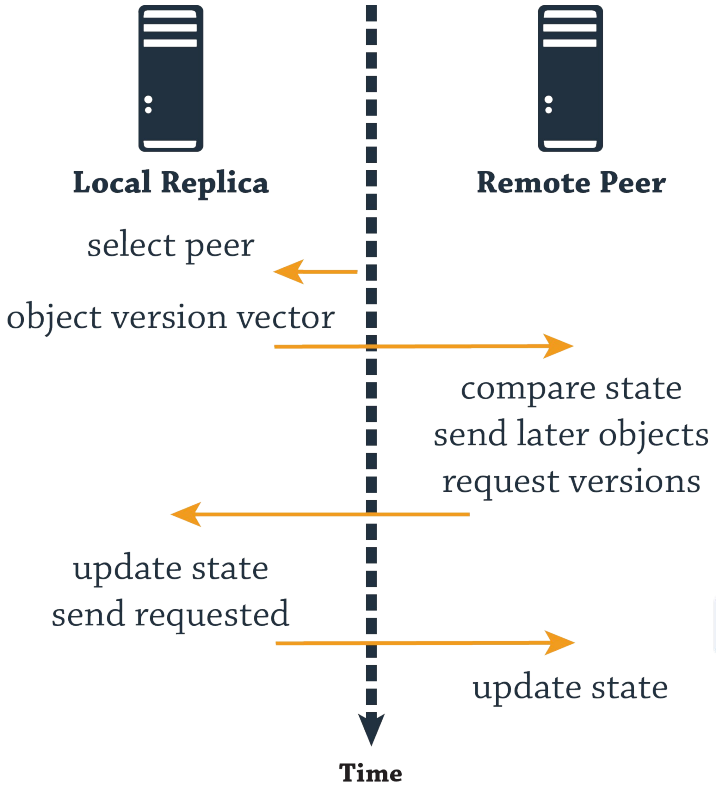
Write burst to same objects every 9 months for reissuance.

Bilateral Anti-Entropy

Nodes **gossip** about latest versions of objects that they've seen for replication and exchange data if they're out of sync.

Eventual consistency: The latest version of the object will eventually be propagated to all nodes in the system.

Latest-writer wins: use of conflict-free distributed version numbers (Lamport scalars)



Consistency

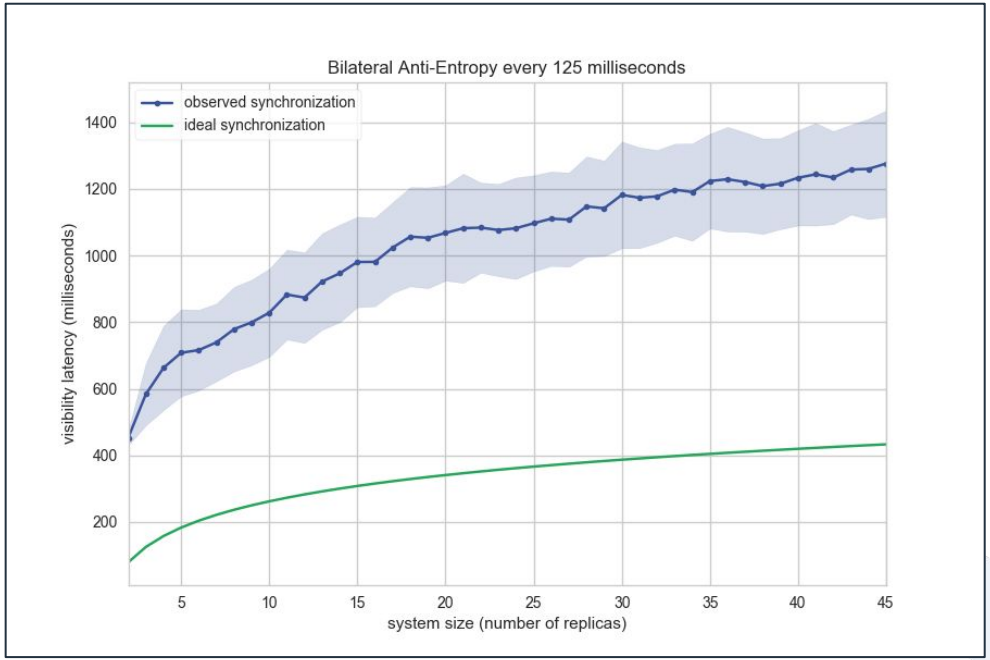
Uniform Random Selection of Peers:

- Will eventually converge
- Minimizes broadcast
- Fault tolerant
- Dynamic
- Adds noise to selection

Inconsistencies:

- Stale reads
- Forked writes

Bound by **Visibility Latency**
(Probabilistically Bounded Staleness)



$$\lambda_v = t \log_3 n + \epsilon$$

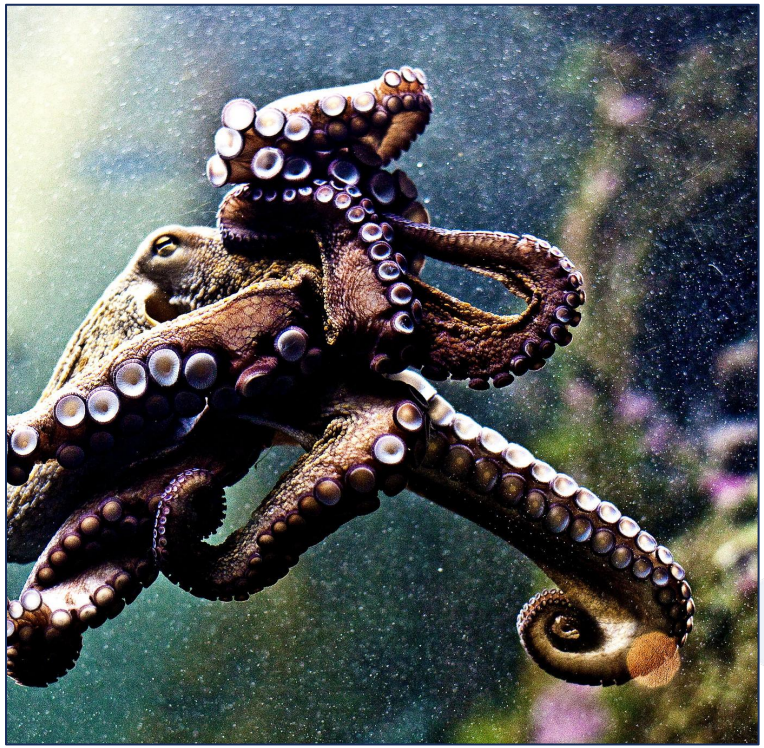
Reinforcement Learning

Intelligent agents make choices and observe outcomes in a specific environment in order to maximize a cumulative reward function.

Multi-Armed Bandits

Each slot machine (a one-armed bandit) has a different payout rate and odds probability.

Goal: maximize winnings by updating probabilities with observed outcomes of choices.



Exploration vs. Exploitation

When you find a slot machine that's paying out, how do you determine if you should stick with it, or keep trying other machines in case of a higher payout?

Choose random with probability ϵ, choose best with probability 1-ϵ.

- Epsilon Greedy
- Annealing Epsilon Greedy
- Bayesian Sampling
- Upper Confidence Bound

Reward f(x)

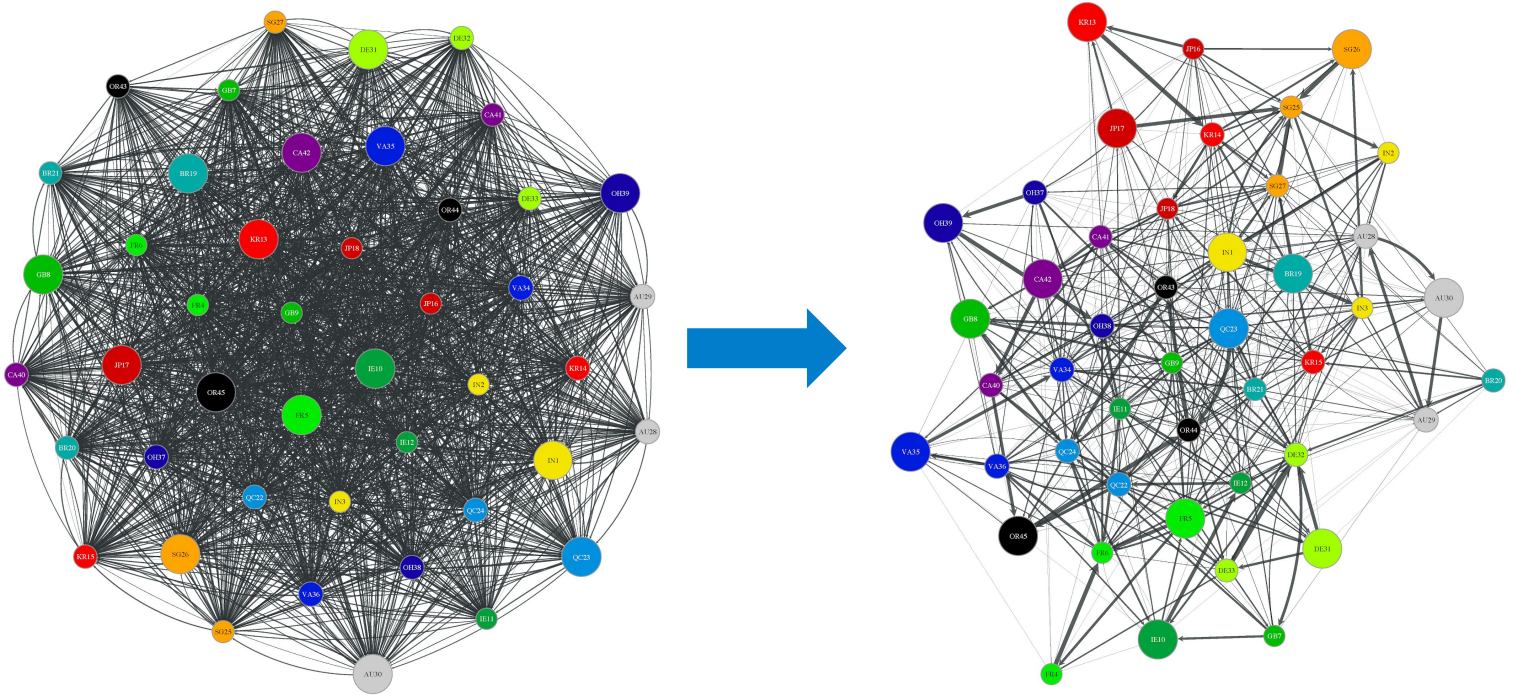
Goal: **learn the optimal topology** based on accesses (particularly writes) such that each peer prioritizes peer selection that leads to a globally emergent decrease in visibility latency.

During every peer selection, the probability of selecting that peer is updated based on the outcome of the synchronization according to a **reward function**.

	Push	Pull	Total
Synchronize at least 1 object	0.25	0.25	0.50
Additional for multiple objects	0.05	0.05	0.10
Latency ≤ 5ms (local)	0.10	0.10	0.20
Latency ≤ 100ms (regional)	0.10	0.10	0.20
Total	0.50	0.50	1.00

Global Emergence

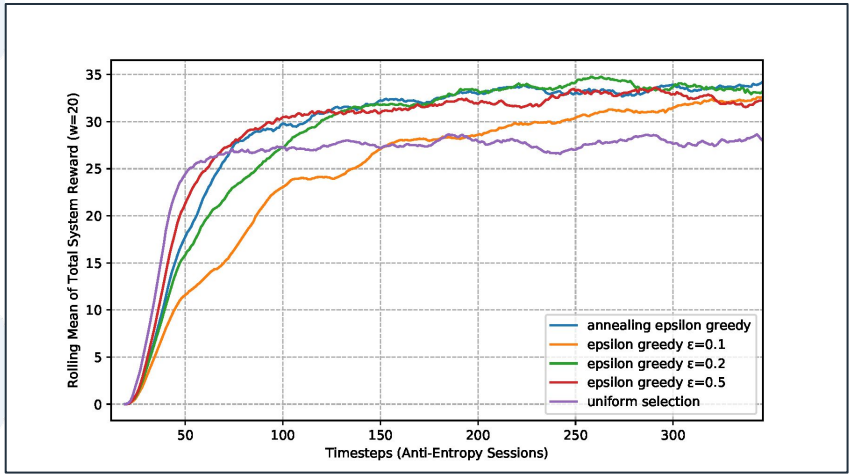
Global patterns emerge from the interactions of local, simple agents that do not exhibit the same patterns independently.



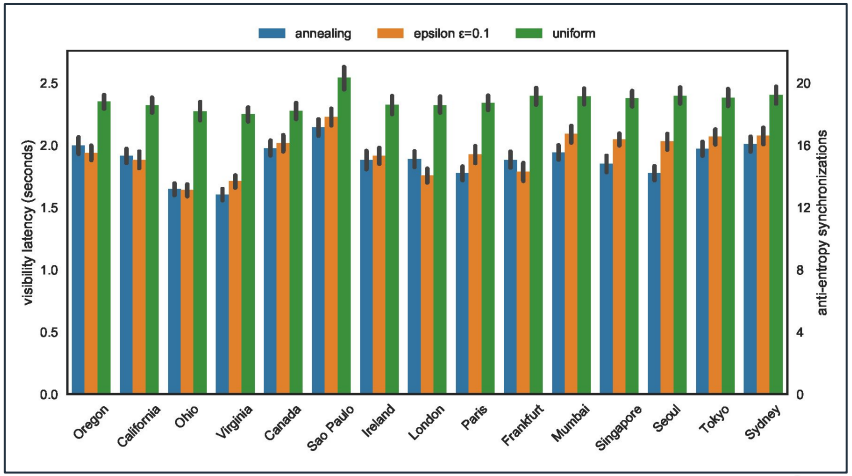
Access Based Topologies Emerge

ROTATIONAL

Real World Results



Bandit strategies eventually improve global rewards 125% over uniform random selection.

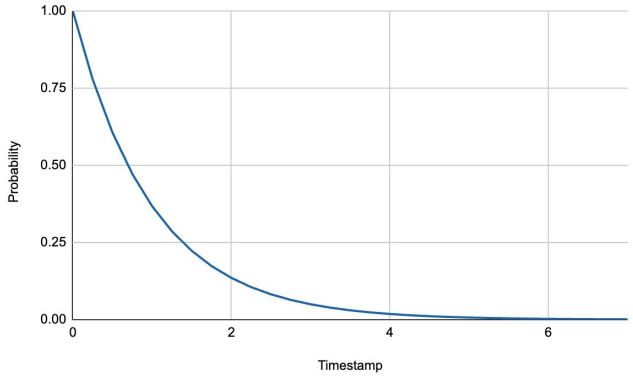


Visibility latency decreased on average by 490ms e.g. by 4 synchronization sessions.

Object Sampling

Instead of always sending *all* version vectors, sample the vectors to send.

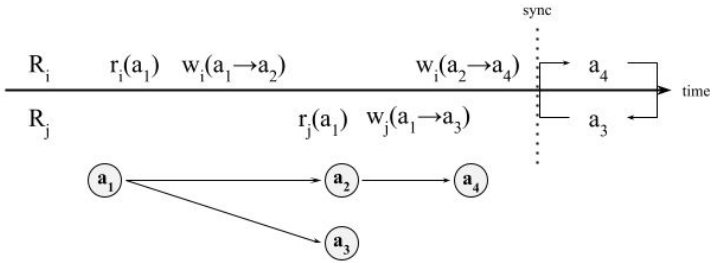
The more time that passes from the last write, the less likely the object is included in the synchronization.



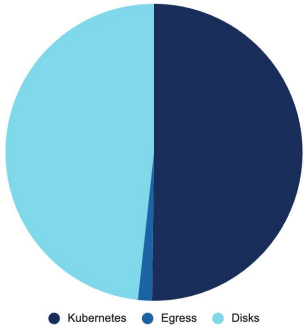
Merge Forked Versions

Rather than simply *stomping* conflicts, detect them and attempt to sequentially apply changes to merge versions.

If versions cannot be automatically merged, present to admins to handle conflicts.



ROTATIONAL



USD $2,550

Final Monthly Costs: **29.8%** of the estimated Spanner Costs

Egress decreased to **14.1%** of the estimated Uniform Random Costs

Takeaways

The data systems we choose to use should not be based on familiarity or marketing but rather their well-suitedness to the particular workload of our application.

Takeaways

The data systems we choose to use should not be based on familiarity or marketing but rather their well-suitedness to the particular workload of our application.

Machine learning techniques can and should be used to optimize distributed systems!

Takeaways

The data systems we choose to use should not be based on familiarity or marketing but rather their well-suitedness to the particular workload of our application.

Machine learning techniques can and should be used to optimize distributed systems!

Design systems with emergence in mind.

Takeaways

The data systems we choose to use should not be based on familiarity or marketing but rather their well-suitedness to the particular workload of our application.

Machine learning techniques can and should be used to optimize distributed systems!

Design systems with emergence in mind.

When you see probability in systems - think reinforcement learning!

ROTATIONAL

Thank You!

⭐ rotationalio/honu

Connect with me at the conference!

in bbengfort

GDS Workload Detail

Locating and describing counterparties is the vast majority. Involves several different kinds of reads.

- searches
- index-based lookups
- full scans

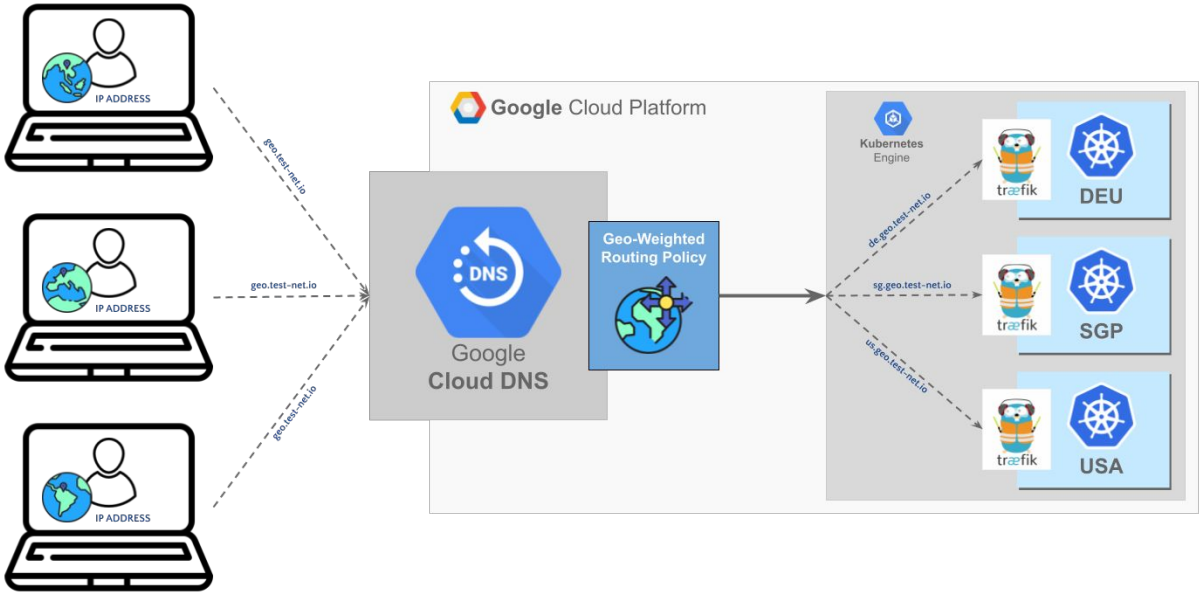
GDS is a read-heavy workload that can tolerate stale reads but must repair conflicting writes as soon as possible.

A stale read might lead to a retry of a travel rule information exchange (routine)

An invalid write could lead to a compliance failure (harmful)

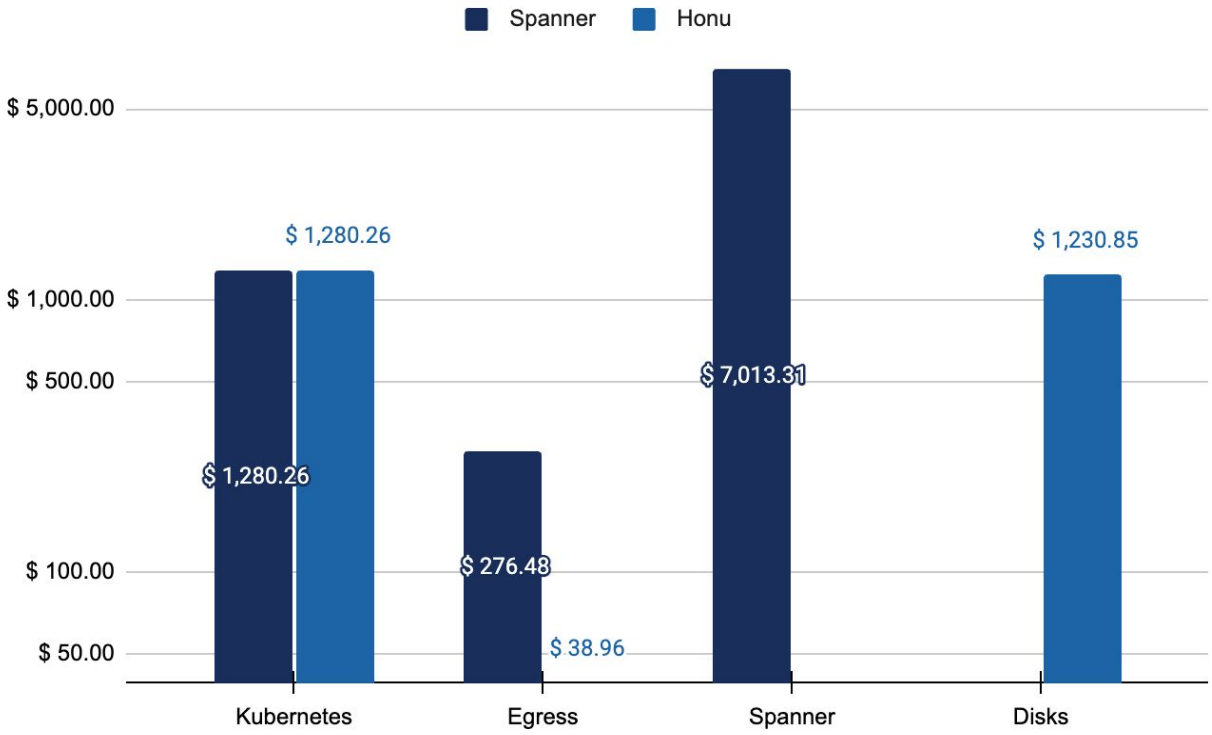
Writes (Rare)	Reads (Near Constant)
There are 3 write loads: 1. Registration, at a pace of: a. 3-4 per month for SGP b. 1-3 per month for USA c. 0-2 per month for DEU Each generates 50-250 writes to 5-11 objects over the course of 5-14 days. 2. Reissuance is routine; Every 6 months, each Virtual Asset Service Provider has 14 writes to 3-6 objects 3. Updates are random; There are 100-1000 per month; usually 1-4 writes to 1-2 objects.	Reads are frequent and near constant. They consist of: - Polling - Automated checks - Searches - Transfer-specific queries This results in 5-30 reads per second, around the clock. - 80% are gets of a single object - 15% are range queries - 5% are over the entire keyspace

Geo-Weighted DNS Routing



We set up our own ingress, but then we needed geo-weighted DNS routing (using Cloud DNS and Route53). Some silver linings:

- Simplified our GDPR compliance, i18n, and other app-specific details
- Enabled multi-cluster deployments
- We are cloud native, but still in the driver's seat



Google Pricing Calculator Detail: Spreadsheets on Request

Abstract

There are many great reasons to replicate data across Kubernetes clusters in different geographic regions: e.g. for disaster recovery and to ensure the best possible user experiences. Unfortunately, global replication is not easy; not just because of the difficulty in consistency reasoning that it introduces, but also due to the increased cost of provisioning multiple volumes that exponentially duplicate ingress and egress. Wouldn't it be great if our systems could learn the optimal placement of storage blocks so that total replication was not necessary? Wouldn't it be even better if our replication messaging was reduced ensuring communication only between the minimally necessary set of storage nodes? We show a system that uses multi-armed bandits to perform such an optimization; dynamically adjusting how data is replicated based on usage. We demonstrate the savings achieved and system performance using a real world system: the TRISA Global Travel Rule Compliance Directory.