GRS: Speculative Execution for Geo-replicated Datastores

Zhongmiao Li • Peter Van Roy • Paolo Romano zhongmiao.li@uclouvain.be

UCL Université catholique de Louvain · IST TÉCNICO LISBOA · inesc id lisboa

Background

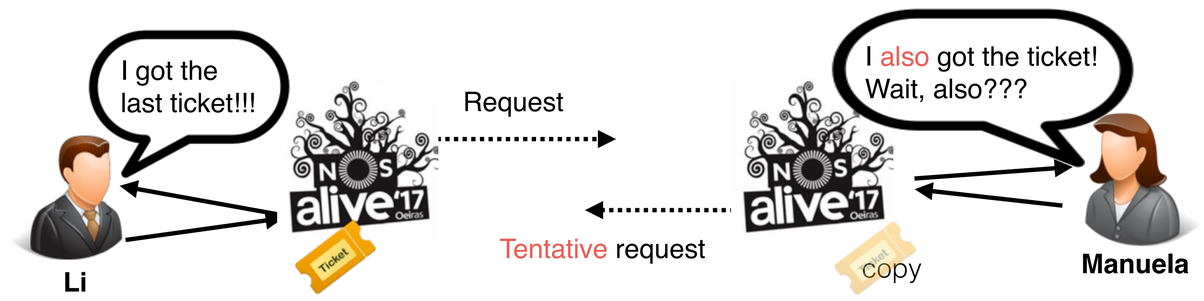
- Large-scale online services strive to serve millions of clients with high reliability and low latency.
- **Geo-replication** is a key technique to achieve these goals!
  - Google has more than 36 data centers across the globe!

- High reliability: tolerate machine failure or even whole data center outage
- Low latency: data is brought closer to clients

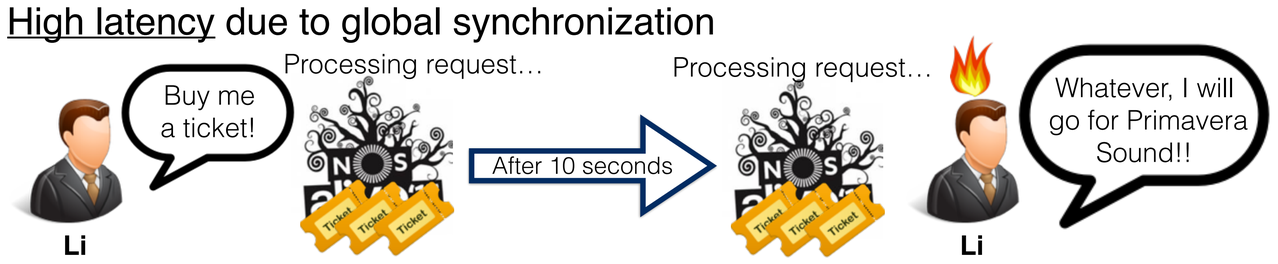
Problem

- Replicas need to **synchronize** to preserve **consistency**.

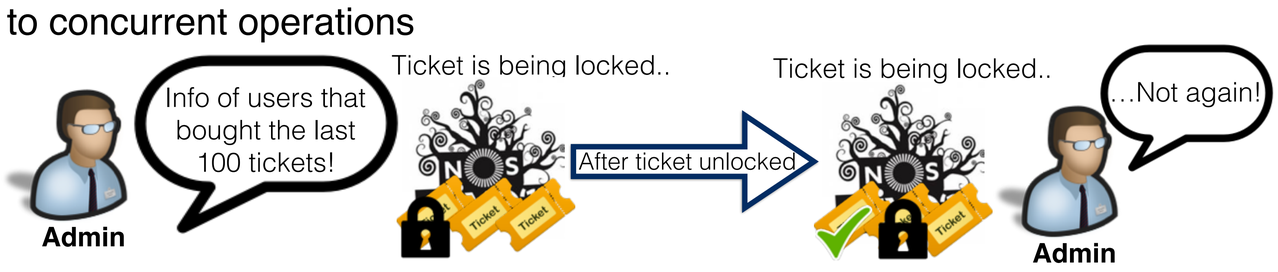
We only have one last ticket of 'NOS Alive'17'!!!



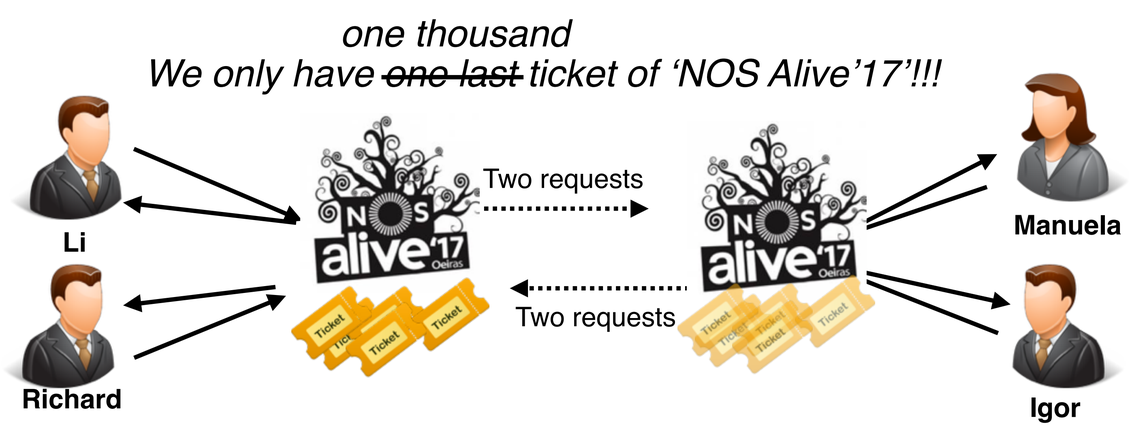
- Coordination among replicas is required before confirming any request. This causes large performance overhead:
- High latency due to global synchronization



- Low throughput: during synchronization, objects are unaccessible (locked) to concurrent operations



Motivation & Solution



Everyone is happy: we got the ticket and latency was low!

Observation & Intuition

Risk of over-selling is only high when there is few tickets left.
Let's be optimistic and speculate on the result of synchronization!

Opportunities

Avoid performance penalty due to replica synchronization:

- Internal speculation: show tentative results to internal operations, but not to end users
  - ✓ Enhance throughput by avoid blocking
  - ✗ Does not reduce perceived latency
- External speculation: expose tentative results to end users
  - ✓ Reduce perceived latency
  - ✗ May require compensation

Challenges & Proposed solutions

Integrate speculative techniques in existing data stores and programming models?

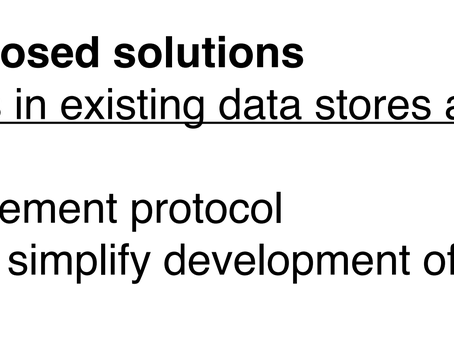
1. New speculative data management protocol
2. New programming models to simplify development of speculative applications

Speculative ticket booking

```
Txn = beginSpeculativeTxn(
        //Commit when commit prob > 0.9
        //Send confirmation when finally commits
        //Send coupon when finally aborts
        )
Ticket = Txn.read(TicketKey)
Ticket.Quantity -= 1
Txn.tryCommit()
```

Predict in which scenarios speculation is likely to succeed?
Use Artificial Intelligence techniques to automate this decision

The tuning technique is able to achieve optimal throughput in various workload scenarios



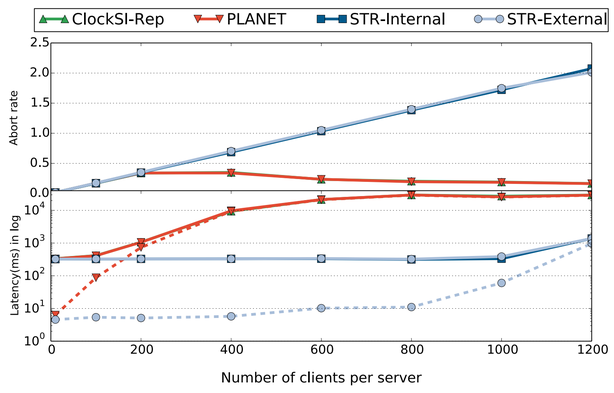
Results & More Information

Platform

- Amazon EC2 cloud service
- 9 data centers scattered over 4 continents

Applications

- Synthetic benchmarks
- Standard benchmarks simulating wholesale supplier (TPC-C and RUBiS)



* Up to 100x reduction of user perceived latency
* Throughput enhanced by up to 6x

For more information: