Eventually Consistent: Not What You Were Expecting?

María Fernanda Mora Alba
Luis Manuel Román García

“In an ideal world there would be only one consistency model: when an update is made all observers would see that update”

- Werner Vogels

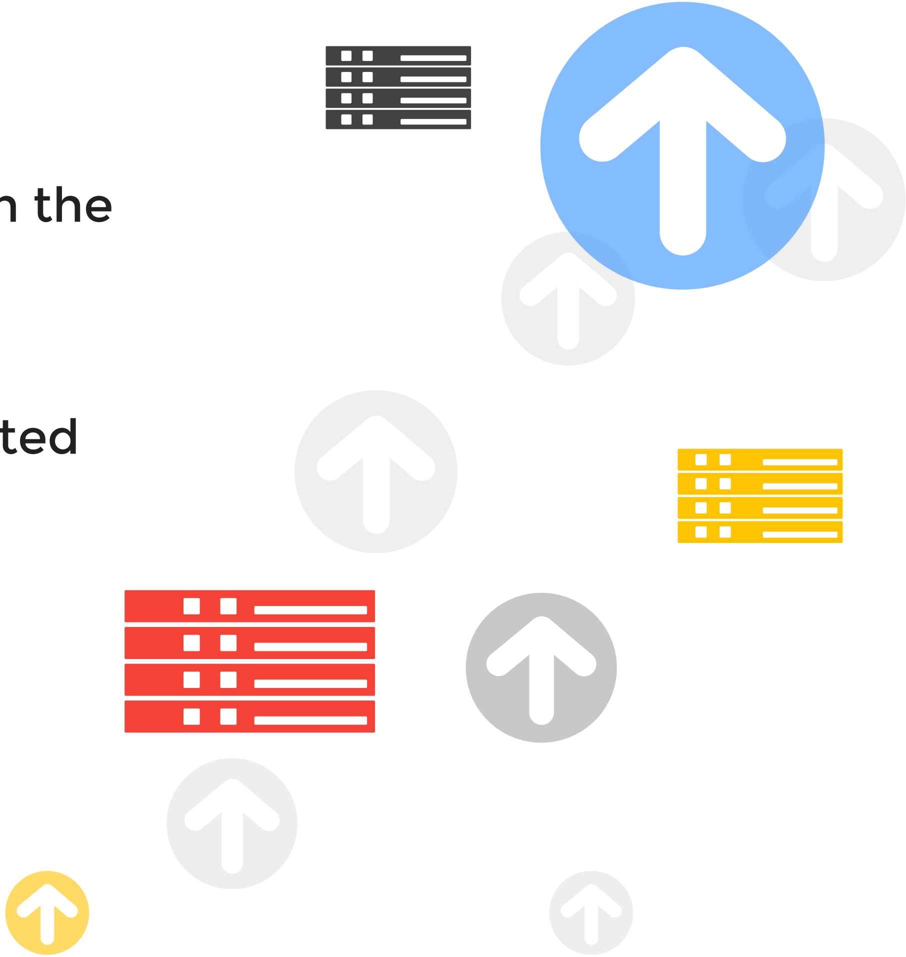
Contents

Introduction

Introduction

Distributed computing is on the rise. The reasons of their importance are several:

- Geographically distributed environment.
- Speed up.
- Resource sharing.
- Fault-tolerance.

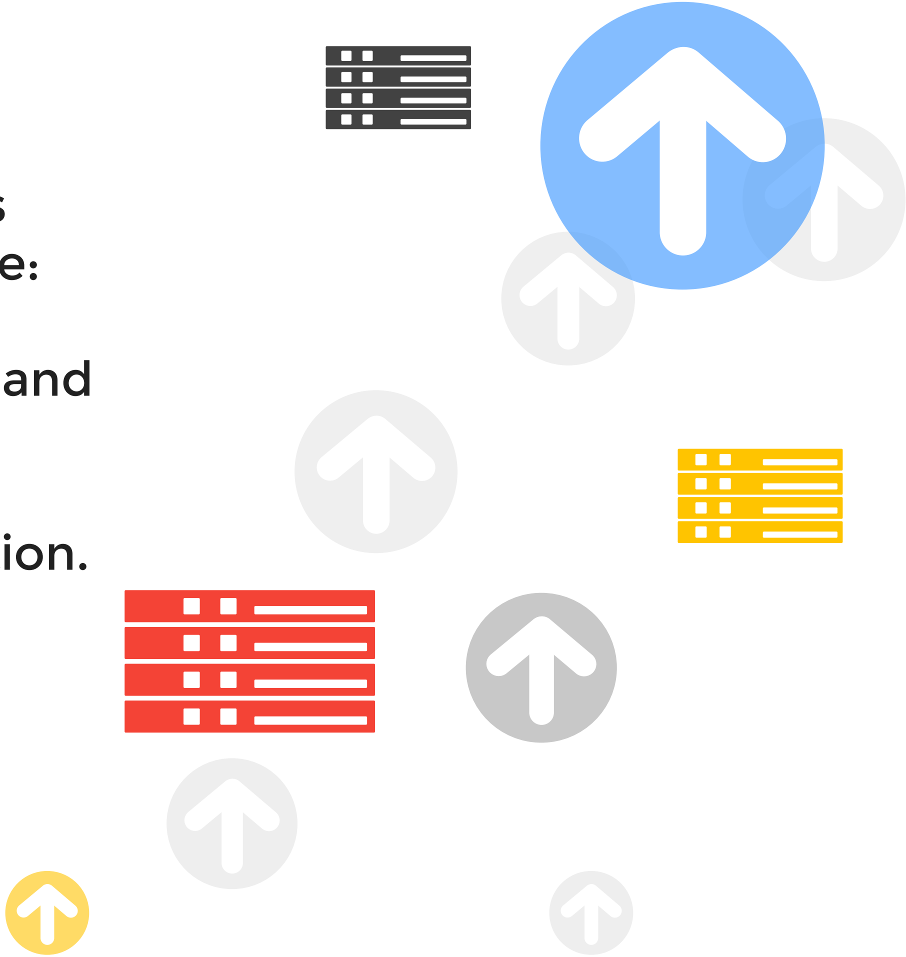


Taylor & Francis Group, LLC, 2007.

Introduction

Some of the main concerns regarding these systems are:

- Process independence and anonymity.
- Network topology.
- Degree of synchronization.
- Failures.

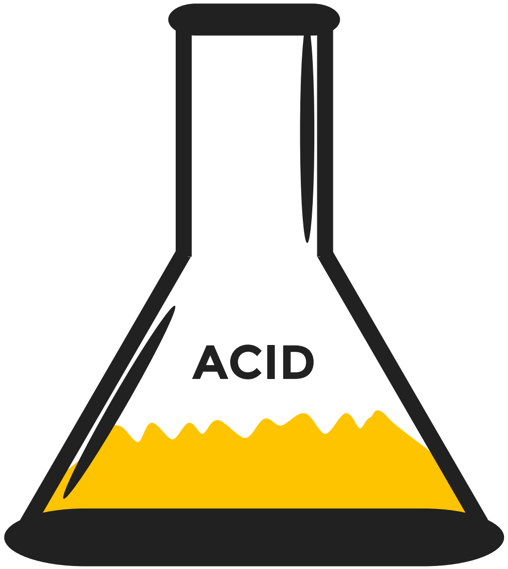


Taylor & Francis Group, LLC, 2007.

Introduction

In most distributed database systems the concept of transaction is paramount. A transaction is a unit of consistent and reliable computation. Therefore, it must be:

- Atomic
- Consistent
- Isolated
- Durable

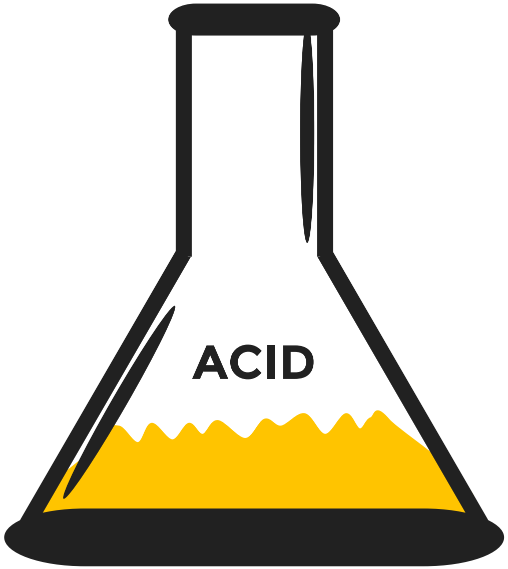


communication of the ACM, january 2009, vol 52, no. 1

Introduction

In the early days of distributed computing, atomicity was the main concern of the database designer.

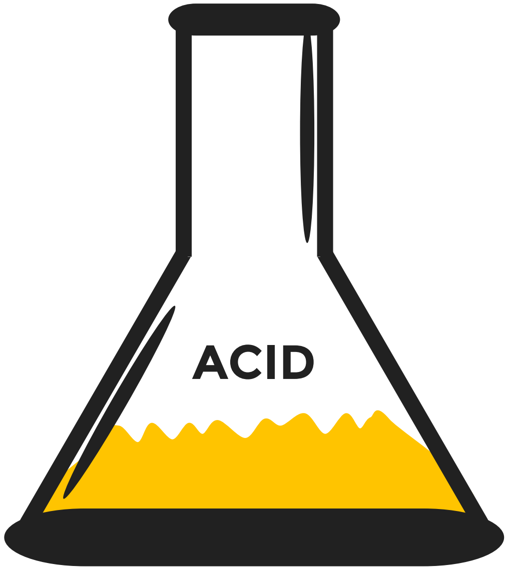
This was mainly enforced by the principle posed by Bruce Lindsay in its Notes on Distributed Databases that required that the distribution of database systems should be **transparent**.



Introduction

In the 1990's, with the grown of internet systems, this principles were revisited and availability became a major concern.

Then suddenly in a conference named Principles of Distributed Computed in 2000, Eric Brewer came out with a brilliant discovery...



communication of the ACM, january 2009, vol 52, no. 1

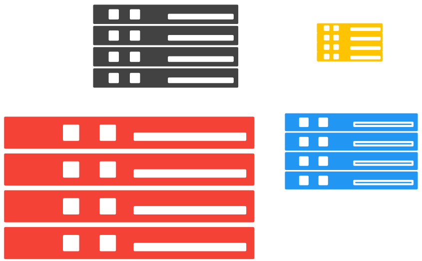
Introduction

The curse of the CAP theorem

Consistency

Availability

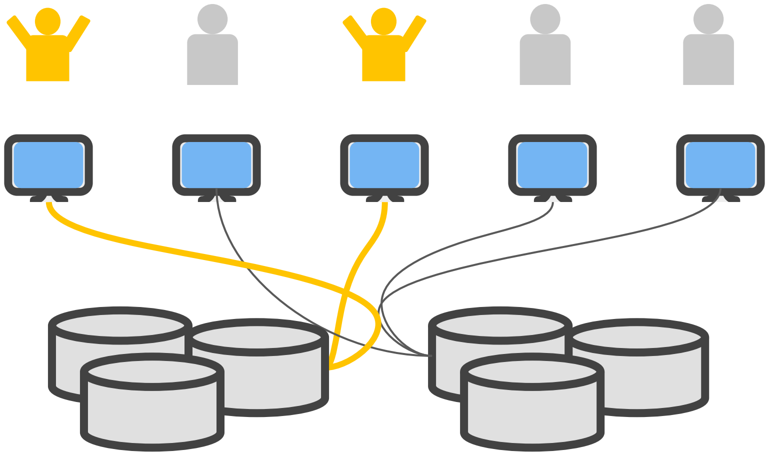
Partition Resistance



Defining Eventual Consistency

Defining Eventual Consistency

It can be defined either as a property of the underlying storage system or as a behavior observed by a client application.

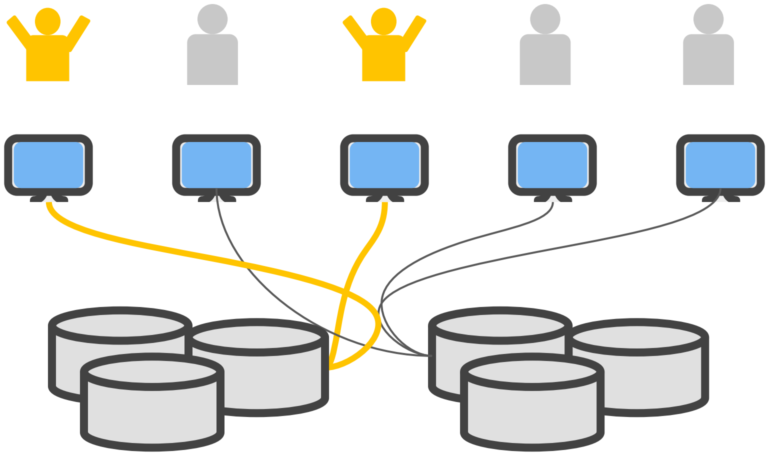


Defining Eventual Consistency

It can be defined either as a property of the underlying storage system or as a behavior observed by a client application.

"All replicas eventually receive all writes, and any two replicas that received the same set of writes have identical databases."
- Doug Terry

Storage system perspective

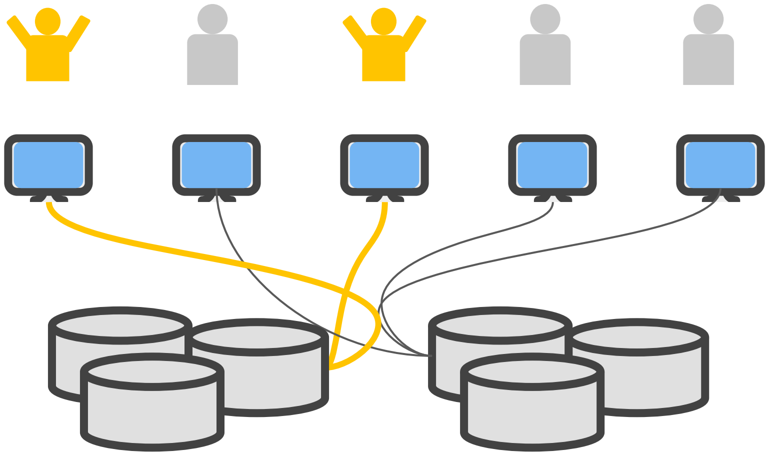


Defining Eventual Consistency

It can be defined either as a property of the underlying storage system or as a behavior observed by a client application.

"The storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value."
- Werner Vogels

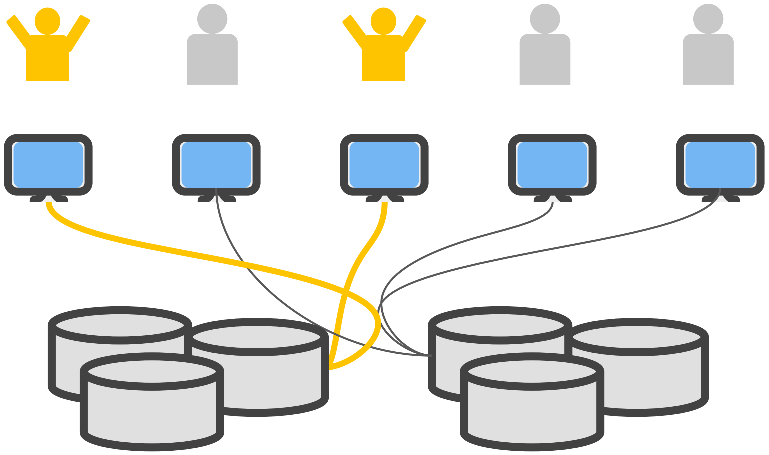
Client application perspective



Defining Eventual Consistency

Abstract definitions of eventual consistency leave open a number of questions regarding behaviour under concurrent accesses & failure prone environments.

It also leaves unresolved the issues regarding propagation speed and consistency among replicas.

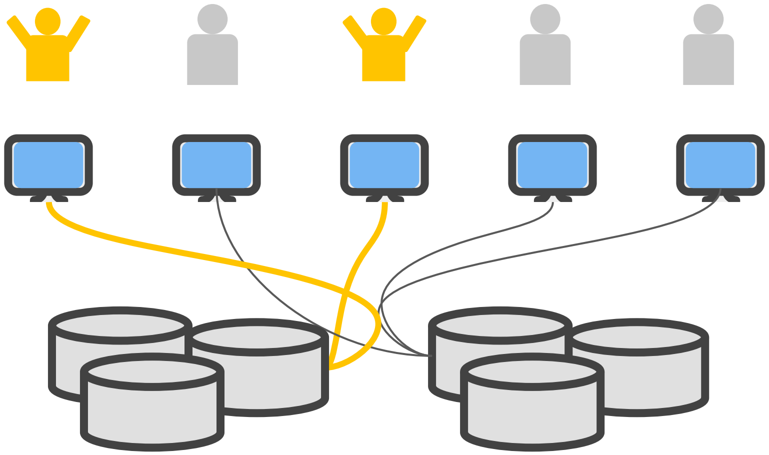


Defining Eventual Consistency

Due to the fact that eventually consistent systems are difficult to grasp, some other properties have been proposed, such as:

- Monotonic reads.
- Read my writes.
- Causal consistency.

In order to improve our understanding of such systems. In the following we are going to dig deeper in the several properties of an eventually consistent systems.



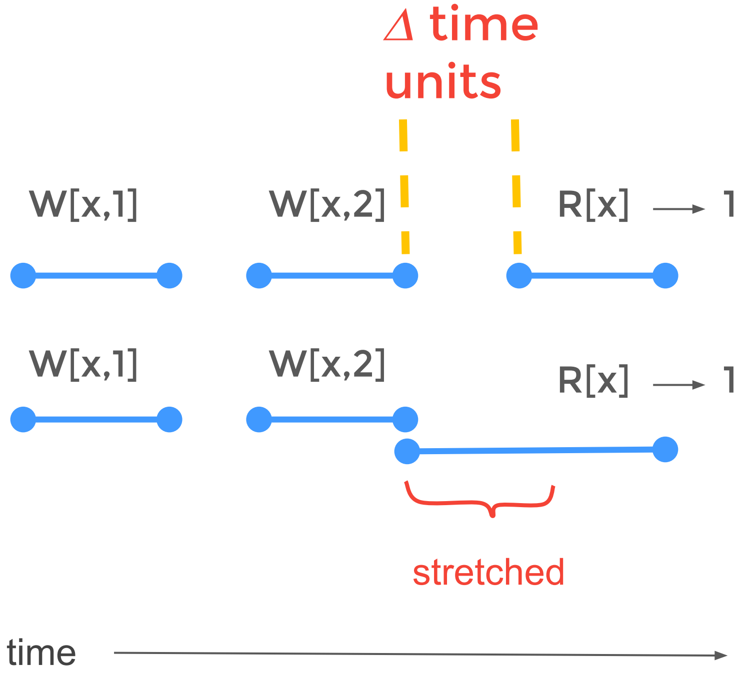
Relaxed Consistency Properties

Relaxed Consistency Properties

Comparing eventually consistent with fully sequential. **Linearizability**

How old is a value:

- Version-based staleness
  k-atomicity
- Time-based staleness
  Δ-atomicity



Prediction

Prediction

Probabilistically Bounded Staleness (PBS) framework by Peter Bailis *et al.* to predict observed data staleness.

Estimates P(<k,t>-staleness).

When k=1, it estimates the probability of non-staleness.

Assumptions:

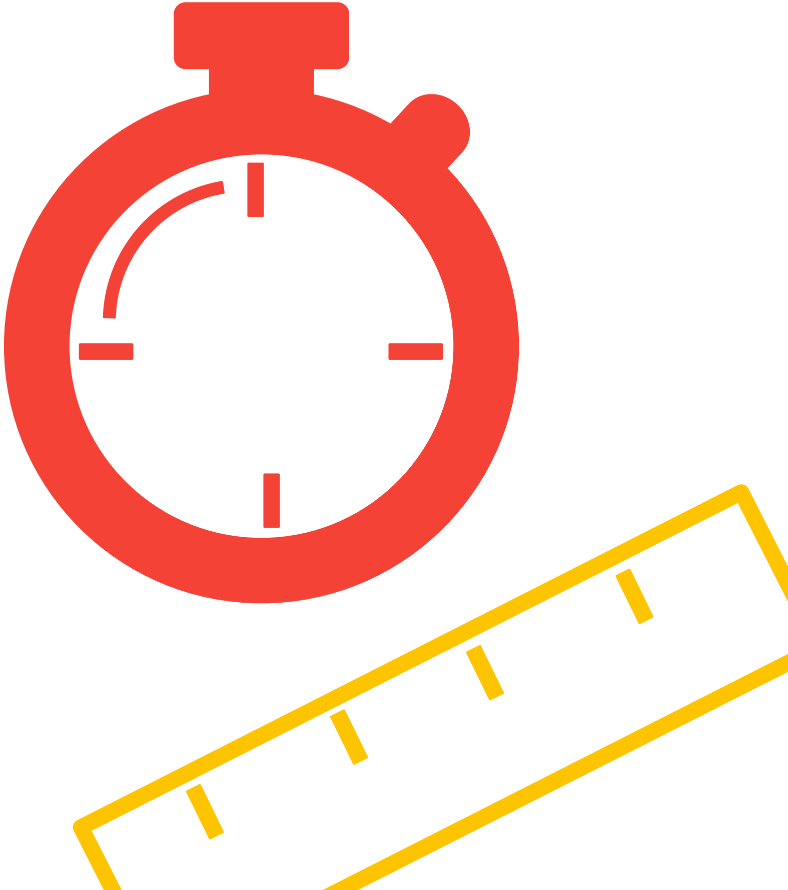
- Does not consider overlapping of reads and writes
- Does not model failures

Empirical Measurement

Empirical Measurement

Difficult task: concurrent operations difficult order identification, clients may always observe a different last updated value due to network partition

2 methodologies: active measurement and passive analysis.



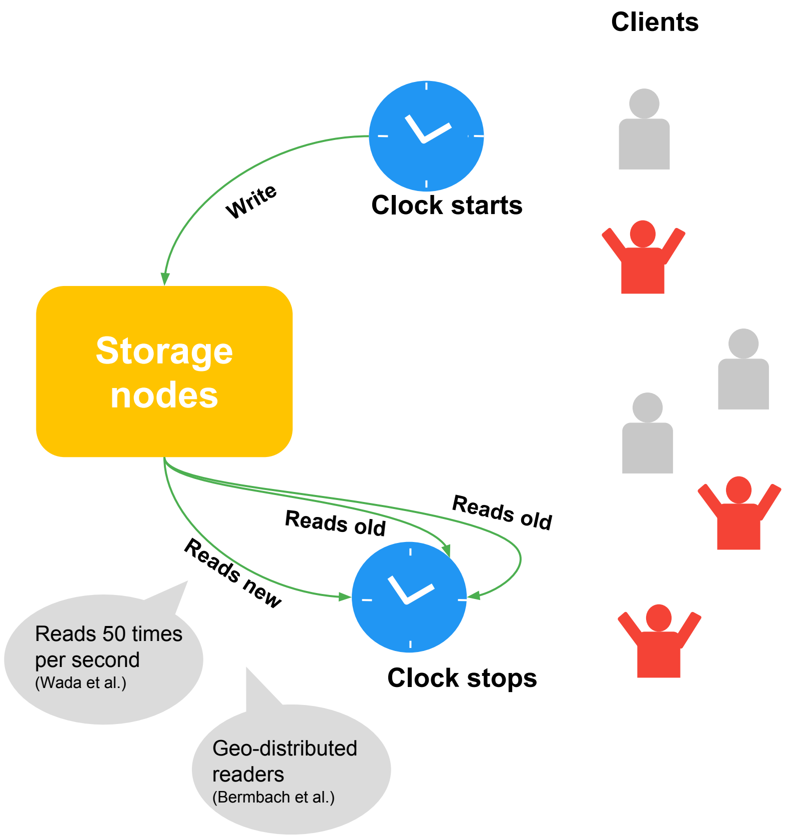
Active measurement

How eventual?: Measures time from the write to the last read that returns the old value.

Estimates the convergence time of the replication protocol (time needed to propagate new values to every replica).

Challenges: determine difference in time between writes and reads at different client nodes.

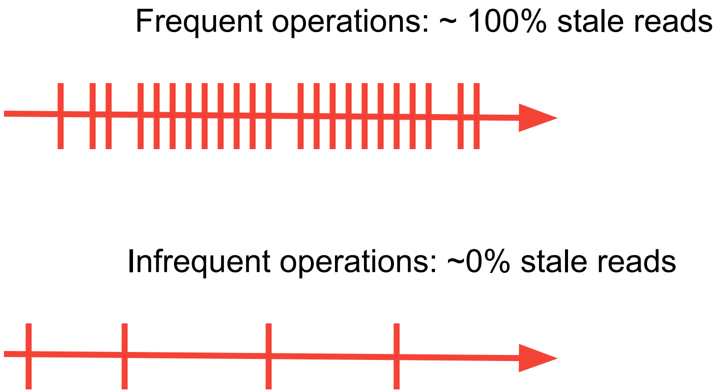
YCSB++ (Yahoo! Cloud-serving benchmark) uses this approach.



Active measurement

Can discover the range of update propagation times in an eventually consistent system: in Amazon SimpleDB, convergence of 90% of runs ocurred in <=1 sec and <1% of runs in >4 secs.

But doesn’t indicate the % of the stale reads.



Passive analysis

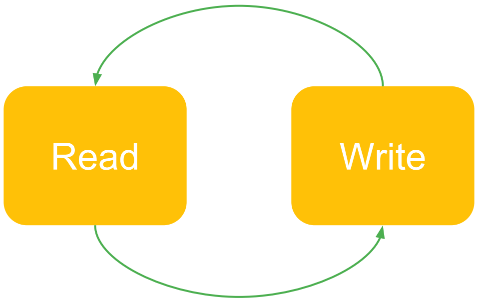
Prior consistency metrics:

1. Linearizability (binary property)
2. Delta-atomicity (bounds staleness of reads): which delta describes best the system? -> we examine operations in trace

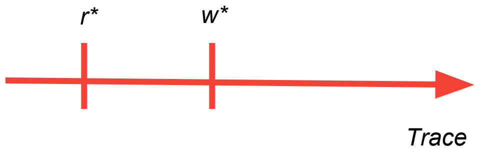
Challenges: collect the trace i.e. for each operation *O* applied to *X* starting at *t1* and finishing in *t2* return the value of *y*.

Problems: dangling reads and reversed operations

Dangling reads



Reversed operations



Comparison

Comparison

Address

Active measurement and PBS are system-centric: controlled workflow allows comparison between systems.

Passive analysis is client-centric: client interaction allows workload comparison.

Model

Active measurement and PBS consider a simplified model (writes before reads).

Passive analysis considers a general model (allows overlapping).

Cost vs taste

Passive analysis is more expensive but reflects better the observed consistency (stale reads).

Future Work & Conclusion

Future Work & Conclusion

Future work

Latency impacts on user experience and revenue.

Weak consistency is used to reduce latency: how much is it reduced?

Consistency & guarantees awareness for differentiated pricing schemes: Amazon's DynamoDB, systems with user wishlist.

Current challenges: consistency verification (SLA, QoS): computation complexity of k-atomicity (only for k<3).

Conclusion

Eventual consistency is not a binary property.

Characterizing, measuring and verifying it will enrich services and improve user's experience.

Thanks