Quantitative Analysis of Consistency in NoSQL Key-value Stores

Si Liu, Son Nguyen, Jatin Ganhotra, Muntasir Raihan Rahman
Indranil Gupta, and José Meseguer

February 2016

NoSQL Systems

- Growing quickly
  - $3.4B industry by 2018

- Apache Cassandra
  - Among top 10 most popular database engines in February 2016
  - Top 1 among all Key-value/NoSQL stores (by DB-Engines Ranking)

- Large scale Internet service companies rely heavily on Cassandra

  - e.g., IBM, eBay, Netflix, Facebook, Instagram, GitHub

Predicting Cassandra Performance...

- …Is Hard. Today's options:
  - **Deploy** on Real Cluster
    - Many man-hours
    - Non-repeatable experiments
  - **Prove theorems on paper**
    - Very hard to do for performance properties
  - **Simulations**
    - Large and unwieldy
    - Take time to run
    - Hard to change (original Cassandra is 345K lines of code)

Our Approach

1. Specify formal model of Cassandra (in Maude language)
2. Use statistical model-checking to measure performance of Maude model
3. Validate results with real-life deployment
4. Use model to predict performance

– First step towards a long-term goal: a library of formal executable building blocks which can be mixed and matched to build NoSQL stores with desired consistency and availability trade-offs

How we go about it

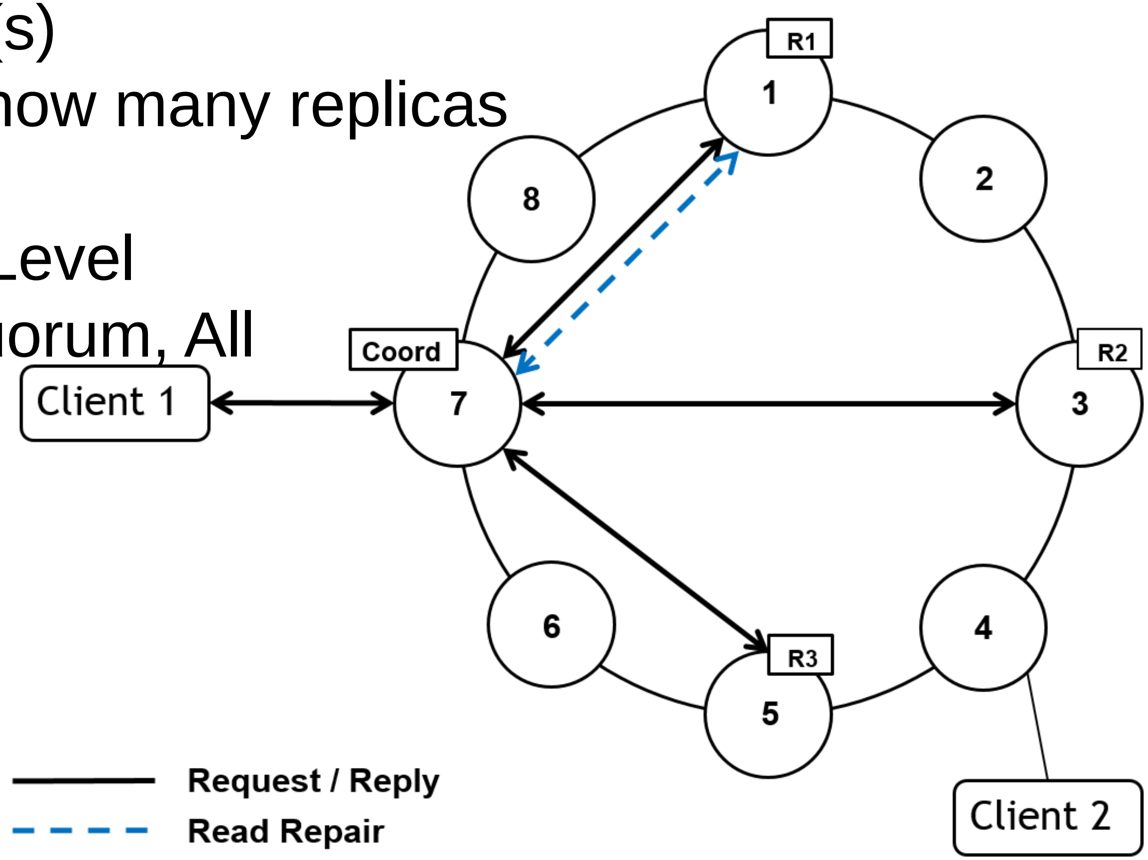
We use:

1. **Maude**
   - Modeling framework for distributed systems
   - Supports rewriting logic specification and programming
   - Efficiently executable

2. **PVeStA**
   - Statistical model checking tool
   - Runs Monte-Carlo simulations of model
   - Verifies a property up to a user-specified level of confidence

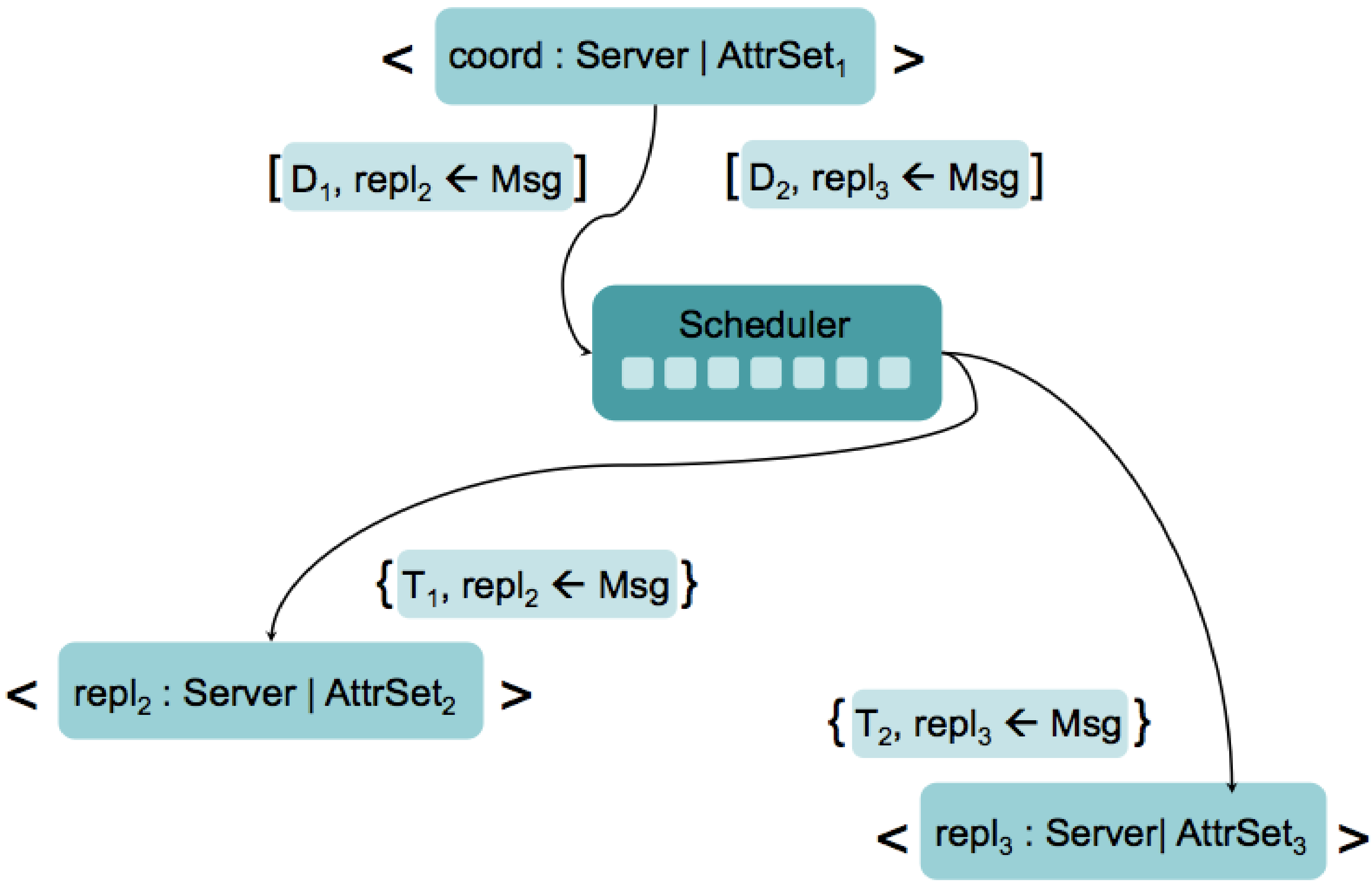
Apache Cassandra Overview

- Cassandra is deployed in data centers
- Each key-value pair replicated at multiple servers
- Clients can read/write key-value pairs
- Read/write goes from client to Coordinator, which forwards to replica(s)
- Client can specify how many replicas need to answer
  - Consistency Level
  - E.g., One, Quorum, All

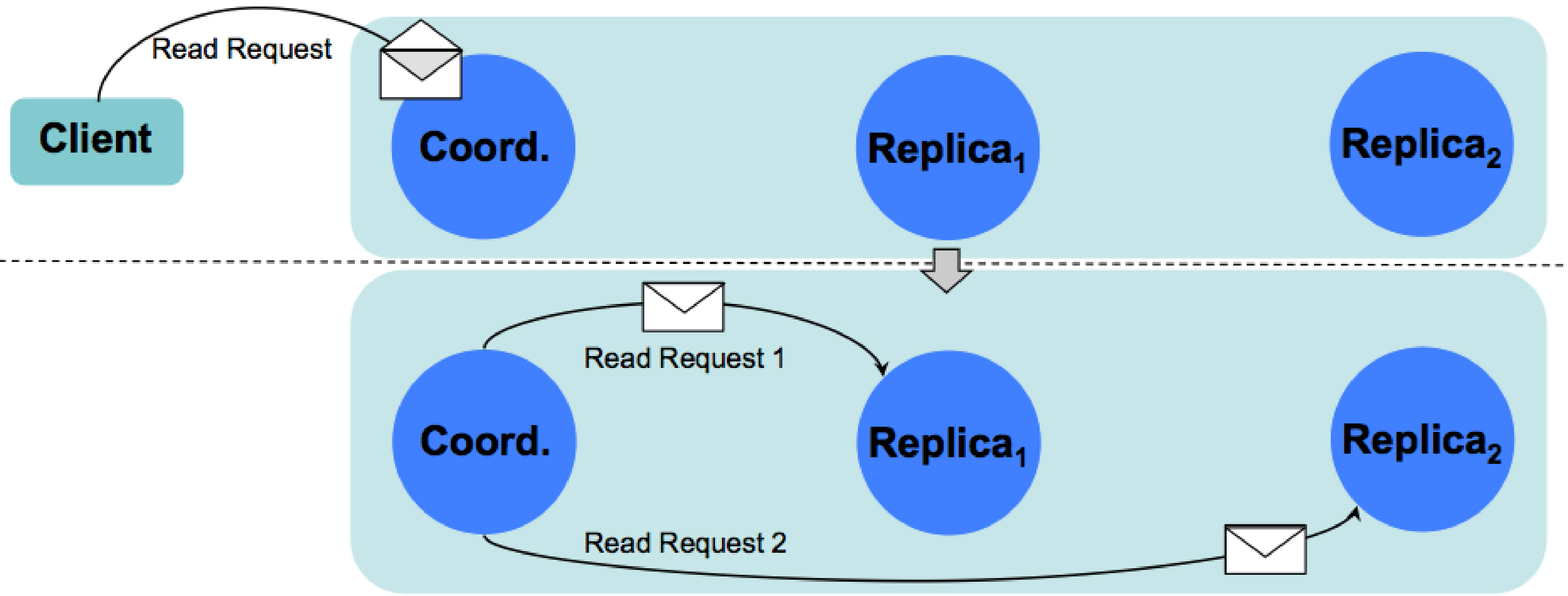


Cassandra Model in Maude: Reads & Writes

- The distributed state of Cassandra model is a “multiset” of Servers, Clients, Scheduler and Messages



Cassandra Model in Maude: Requests




```
crl [COORD-FORWARD-READ-REQUEST] :
   < S : Server | ring: R, buffer: B, ...>
   {T, S <- ReadRequestCS(ID,K,CL,A)}
 =>
   < S : Server | ring: R, buffer:
                       insert(ID,fac,CL,K,B), ... > C

 if generate(ID,K,replicas(K,R,fac),S,A) => C .
```

```
rl [GENERATE-READ-REQUEST] :
   generate(ID,K,(A',AD'),S,A)
 =>
   generate(ID,K,AD',S,A)
   [D, A' <- ReadRequestSS(ID,K,S,A)]
 with probability D := distr(...)
```

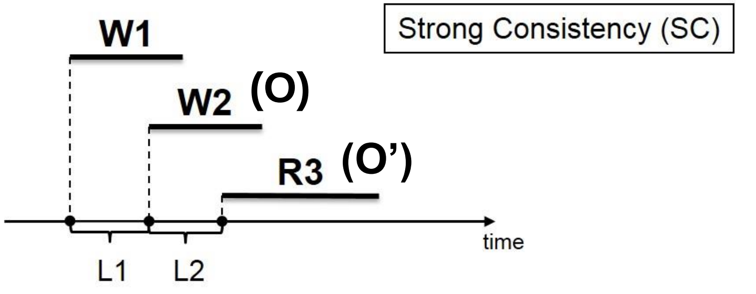
Validating Performance

- We measure Cassandra and an alternative Cassandra-like design's satisfaction of various consistency models
  - strong consistency (SC), read your writes (RYW), monotonic reads (MR), consistent prefix (CP), causal consistency (CC)
- We answer two questions:

1. How well does our Cassandra/alternative design's model satisfy those consistency models?
2. How well do these results match reality (i.e., experimental evaluations from a real Cassandra deployment on a real cluster)?

How experiments go from two sides

- Statistical Model Checking of Consistency
  - design scenario for each property (e.g., SC)
  - formally specify each property (e.g., SC)
  - run statistical model checking



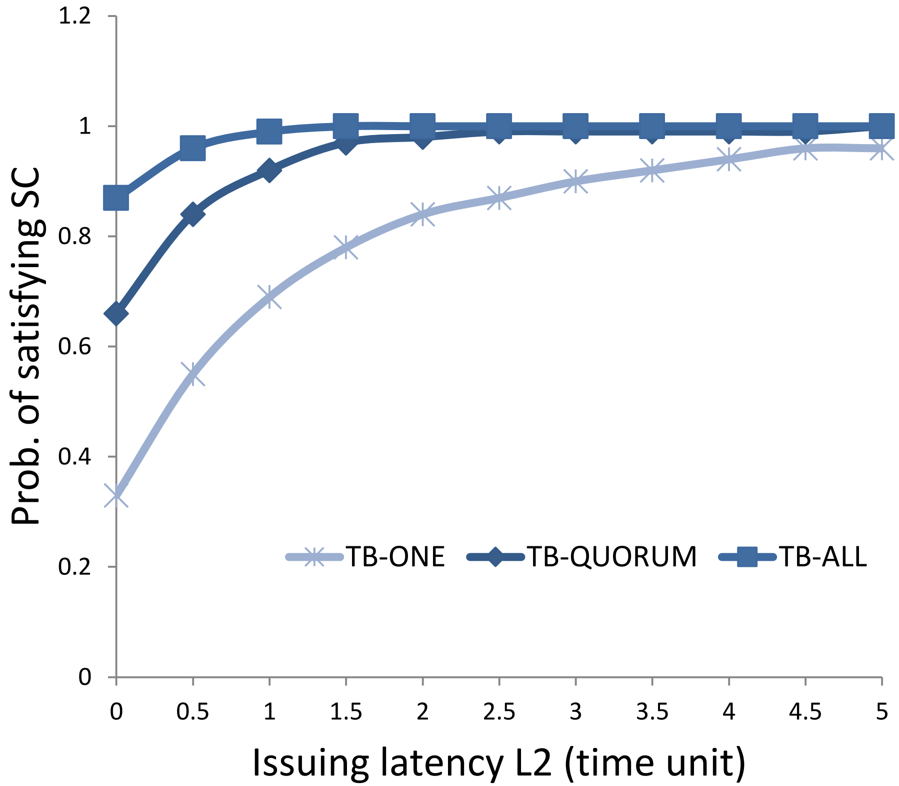
op sc? : Address Address Config → Bool .

eq sc?(A, A', < A : Client | history : ((O, K, V, T), …), … >
< A' : Client | history : ((O', K, V, T'), …), … > REST) = true .

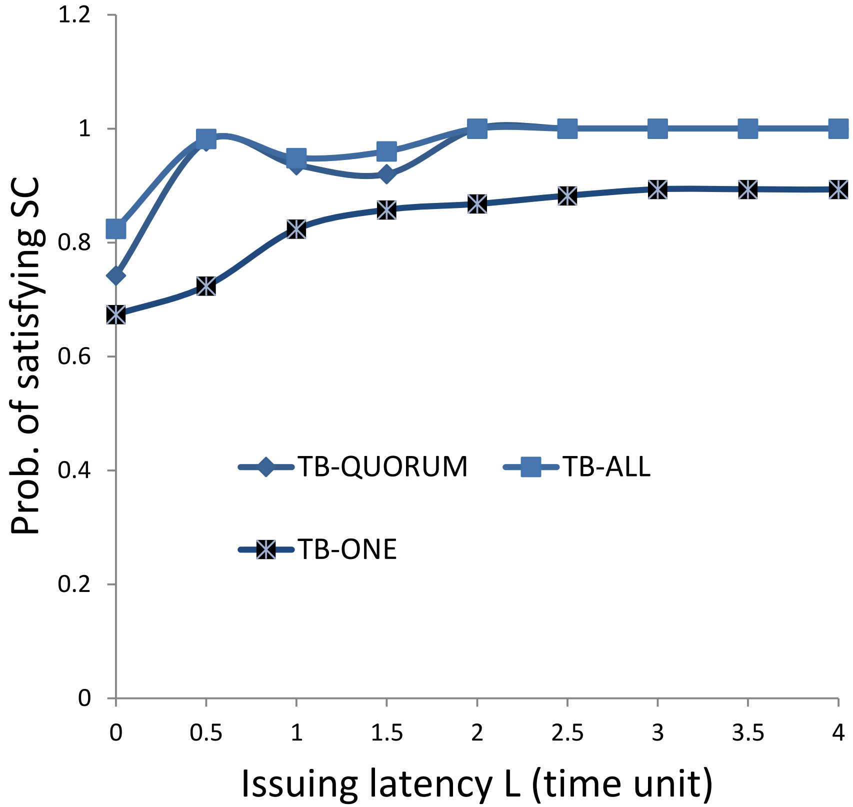
eq sc?(A, A', C) = false [owise] .

- Implementation-based Evaluation of Consistency
  - deploy Cassandra on a single Emulab server
  - use YCSB to inject read/write workloads
  - run Cassandra and YCSB clients (20,000 operations per client) and log the results
  - calculate the percentage of reads that satisfy various consistency models

Performance: Strong Consistency



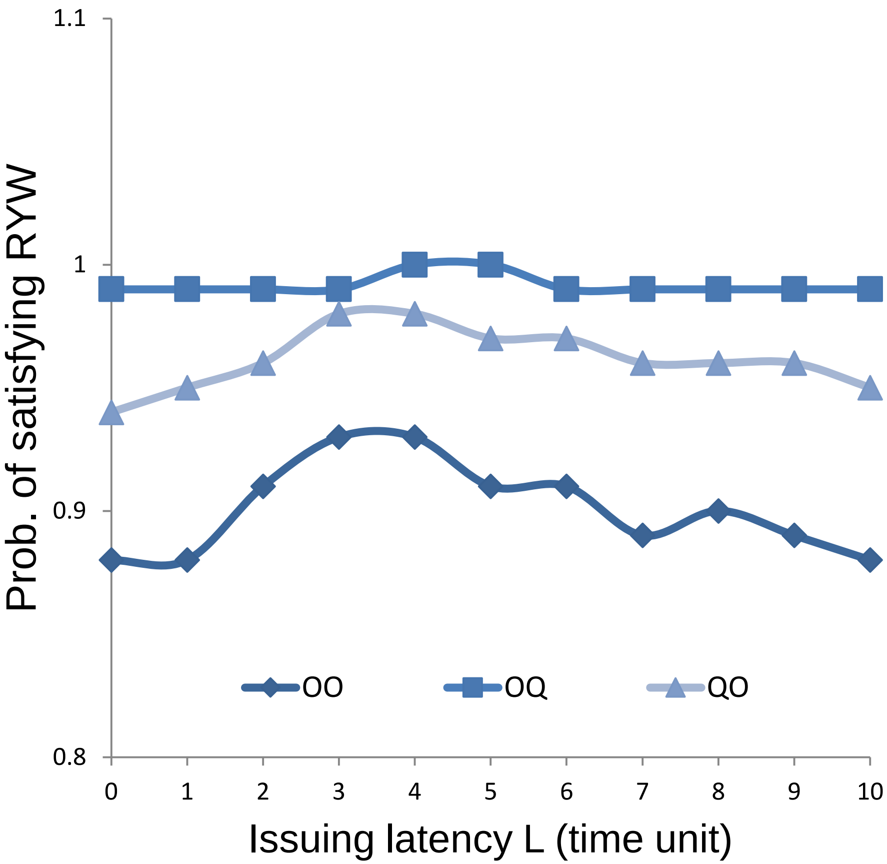
Statistical Model Checker



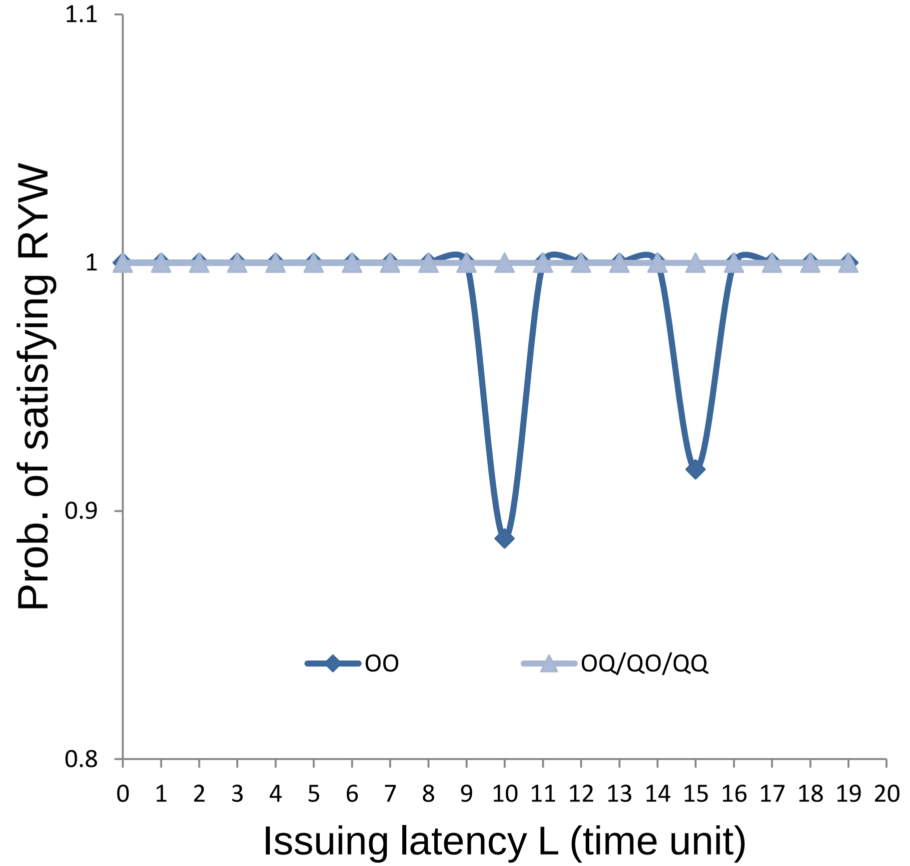
Real-deployed cluster

- (X axis =) Issuing Latency = time difference between the given read request and the latest write request
- (Y axis =) Probability of a request satisfying that model
- **Conclusion: Statistical Model Checker is reasonably accurate in predicting low and high consistency behaviors; both sides show Cassandra provides high** SC **especially with** QUORUM **and** ALL **reads.**

Performance: Read Your Writes



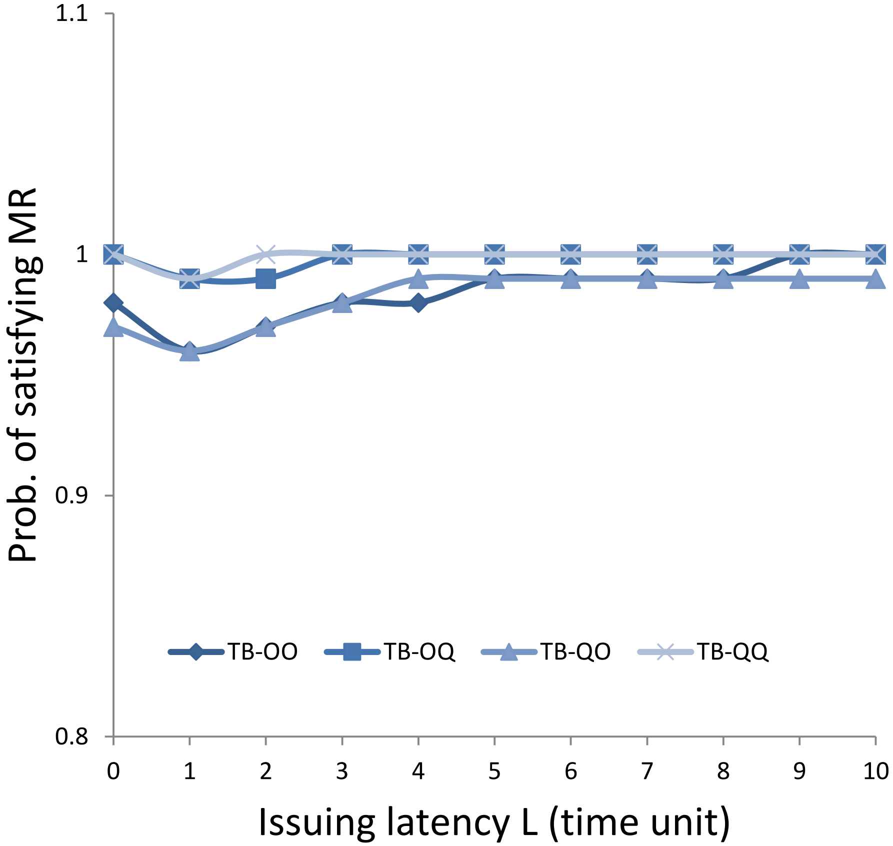
Statistical Model Checker

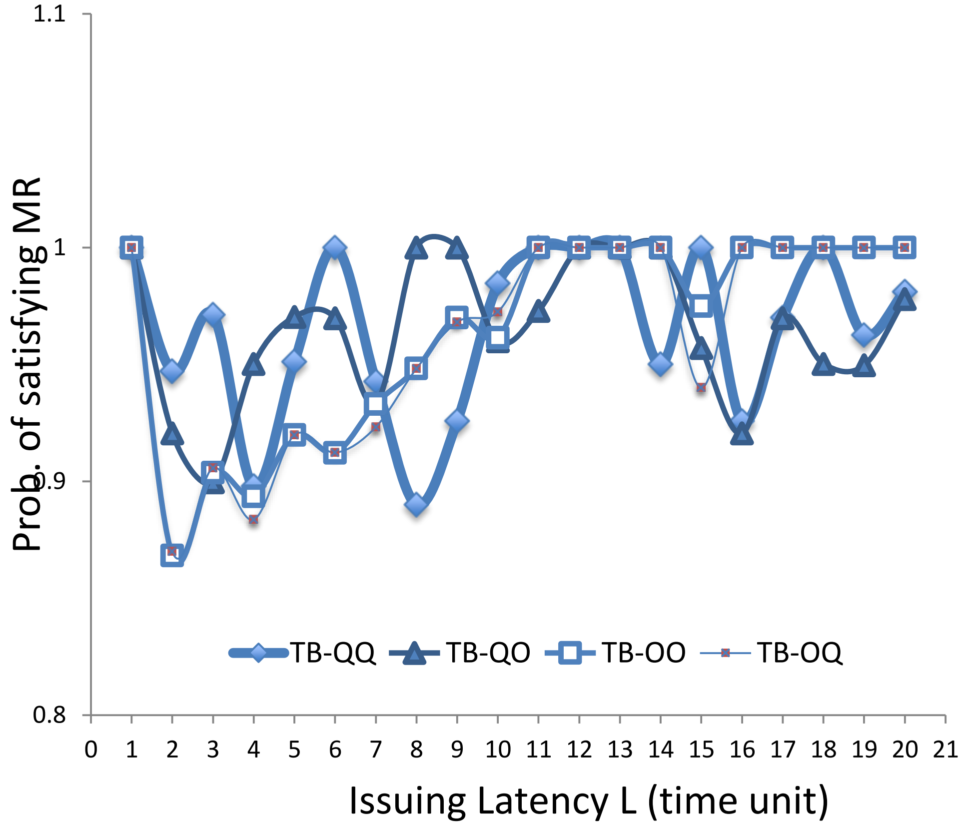


Real-deployed cluster

- **Conclusion: Statistical Model Checker is reasonably accurate (to within 10-15%) in predicting consistency behaviors; high** RYW **consistency is achieved as expected.**

Performance: Monotonic Reads

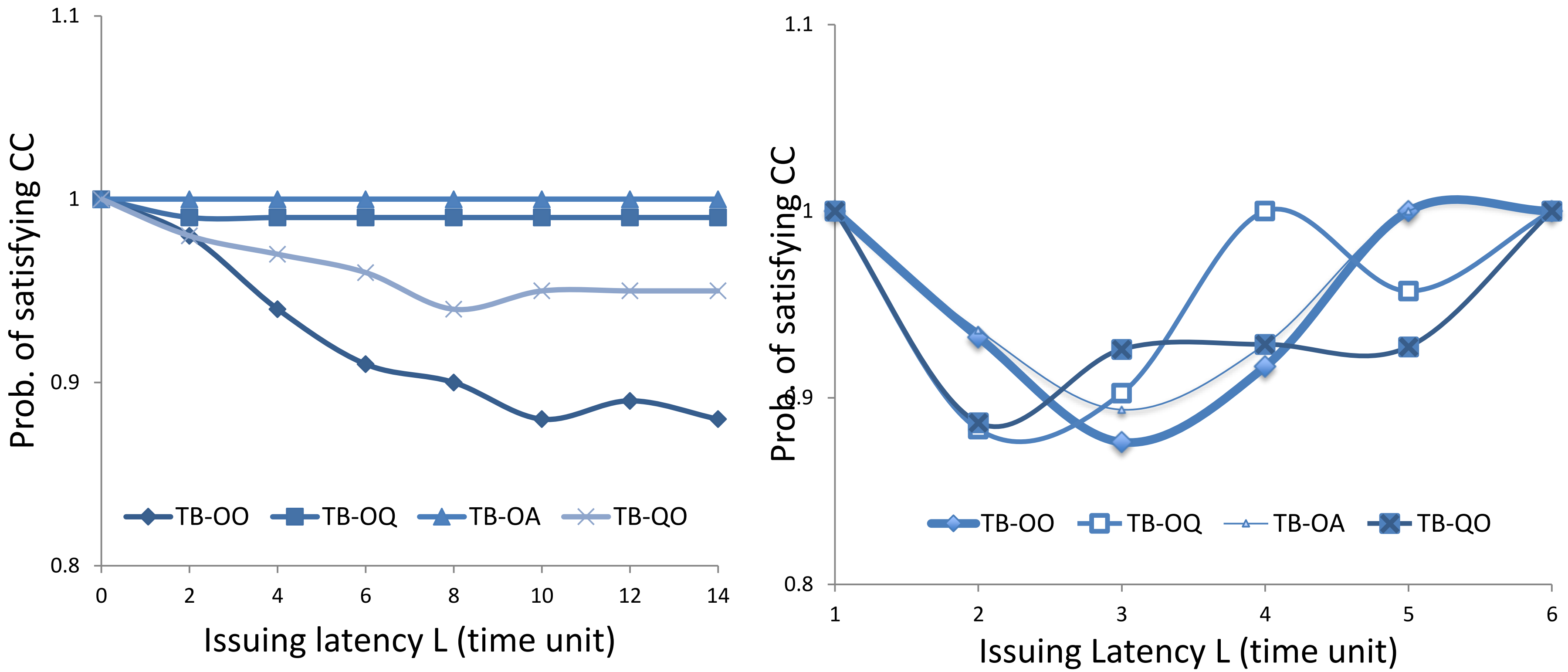




Statistical Model Checker **Real-deployed cluster**

- **Conclusion: Statistical Model Checker is reasonably accurate (to within 10-15%) in predicting consistency behaviors w.r.t. consistency level comb; high** MR **consistency is achieved as expected.**

Performance: Causality



Statistical Model Checker **Real-deployed cluster**

- **Conclusion: Statistical Model Checker is reasonably accurate (to within 10-15%) in predicting consistency behaviors w.r.t. consistency level comb; high CC is achieved as expected.**

Alternative Strategy Design & Comparison

- Timestamp-based Strategy (TB) (Cassandra's Original Strategy)

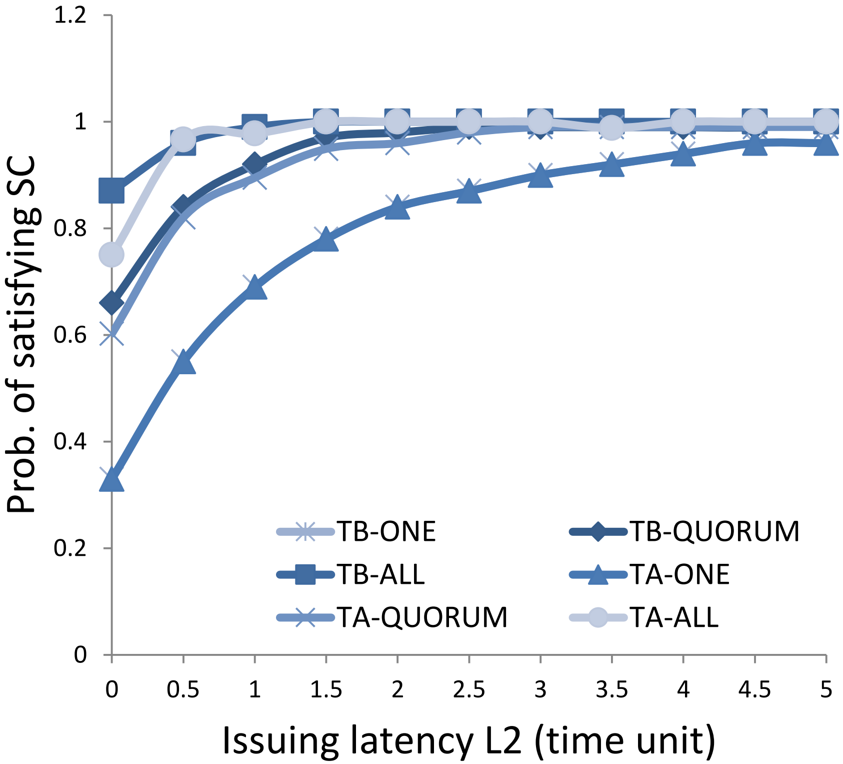
  - uses the **timestamps** to decide which replica has the latest value

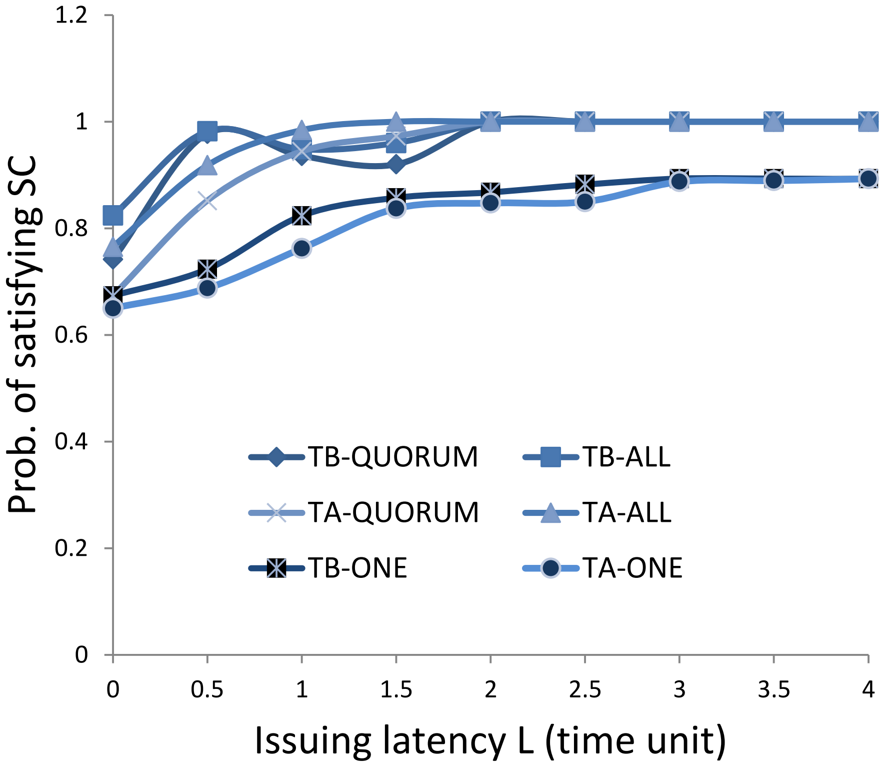
- Timestamp-agnostic Strategy (TA) (Alternative Strategy)

  - uses the **values** themselves to decide which replica has the latest value

- Compare TA and TB in terms of various consistency models both by **statistical model checking** and **implementation-based evaluation**

Comparison: Strong Consistency

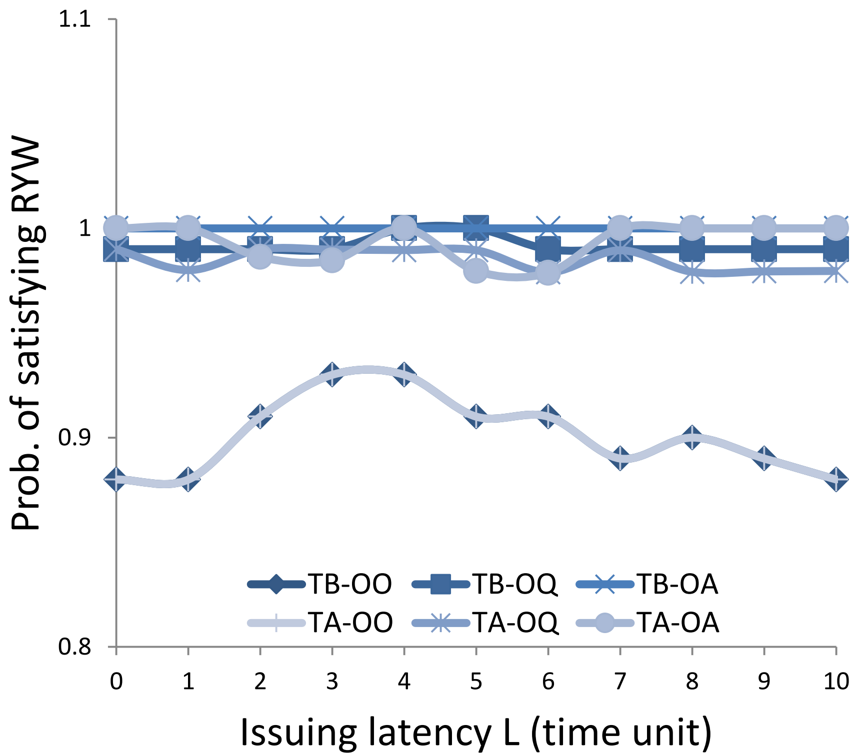


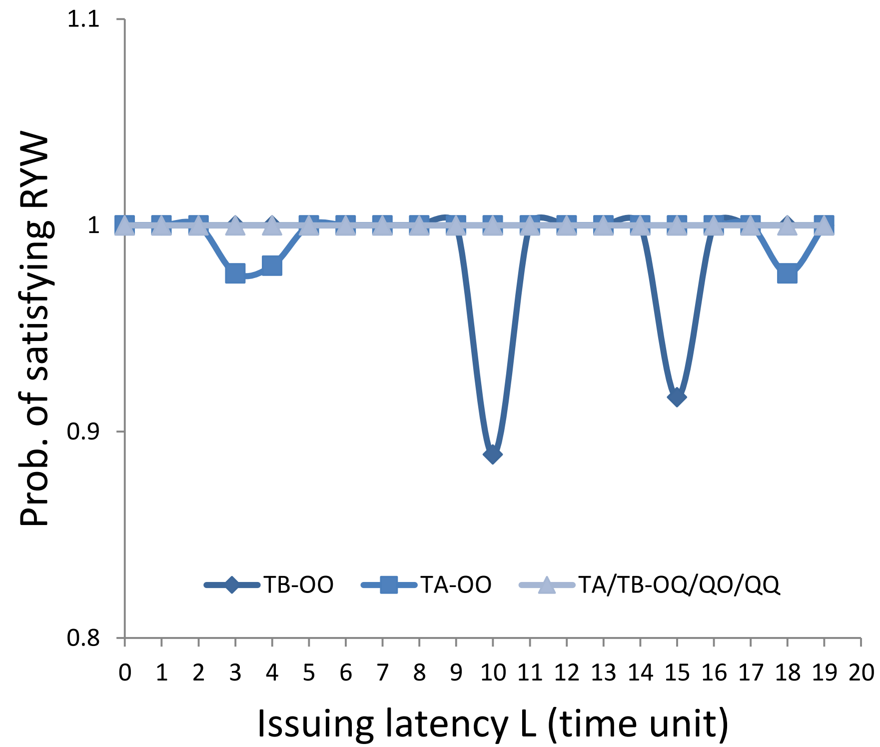


Statistical Model Checker **Real-deployed cluster**

- **Conclusion: Both TA and TB provide high SC, especially with** QUORUM **and** ALL **reads.**

Comparison: Read Your Writes



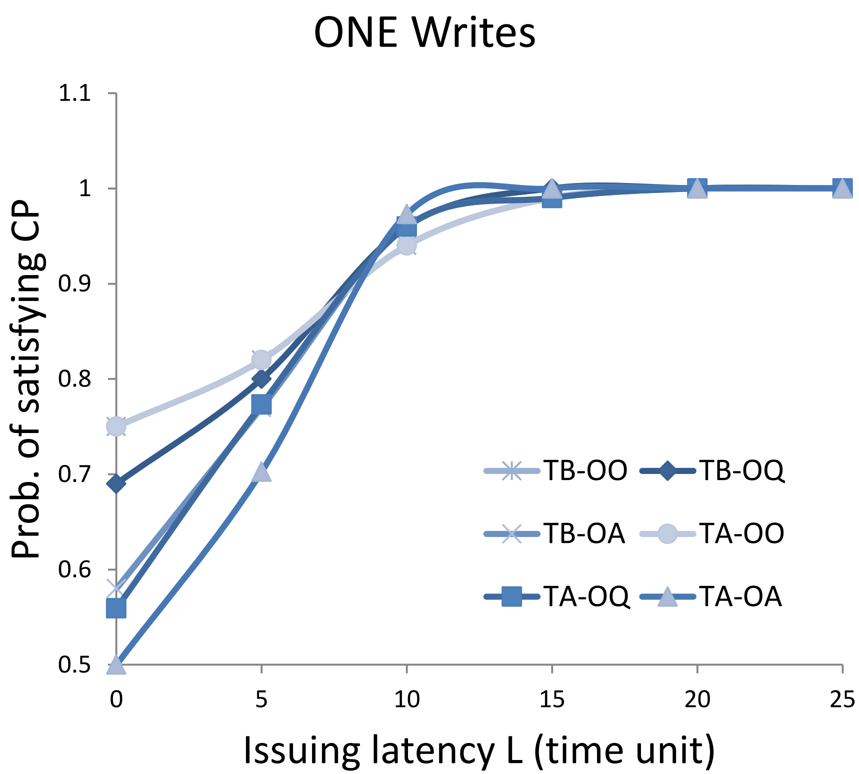


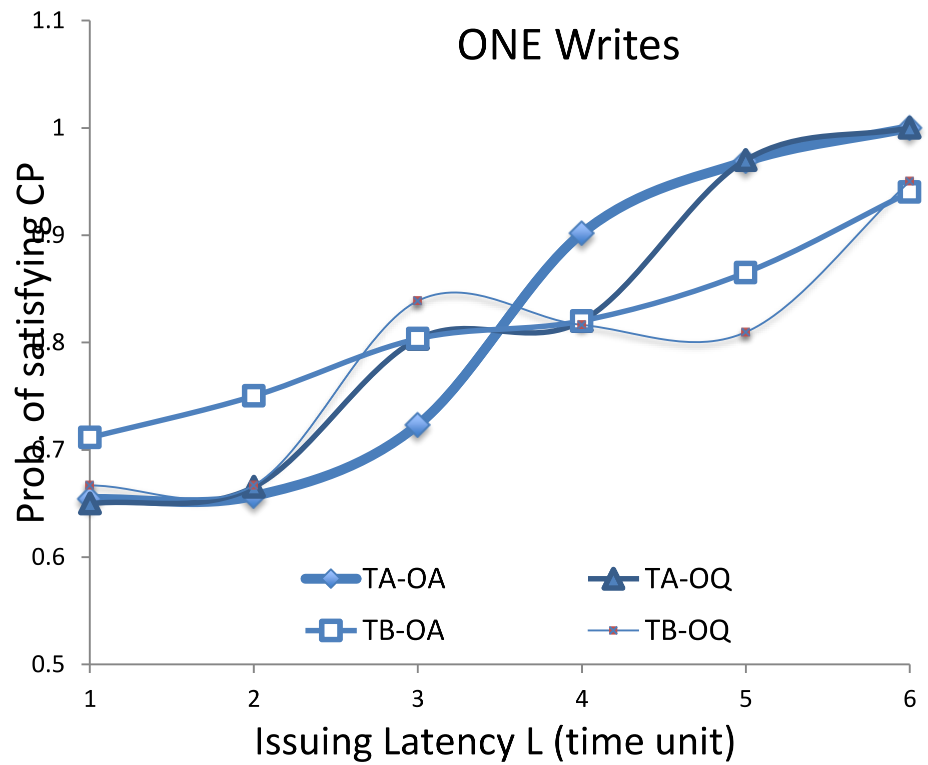
Statistical Model Checker

Real-deployed cluster

- **Conclusion: Both TA and TB offer high RYW consistency as expected; TA provides slightly lower consistency for some points, even though TA's overall performance is close to TB's.**

Comparison: Consistent Prefix



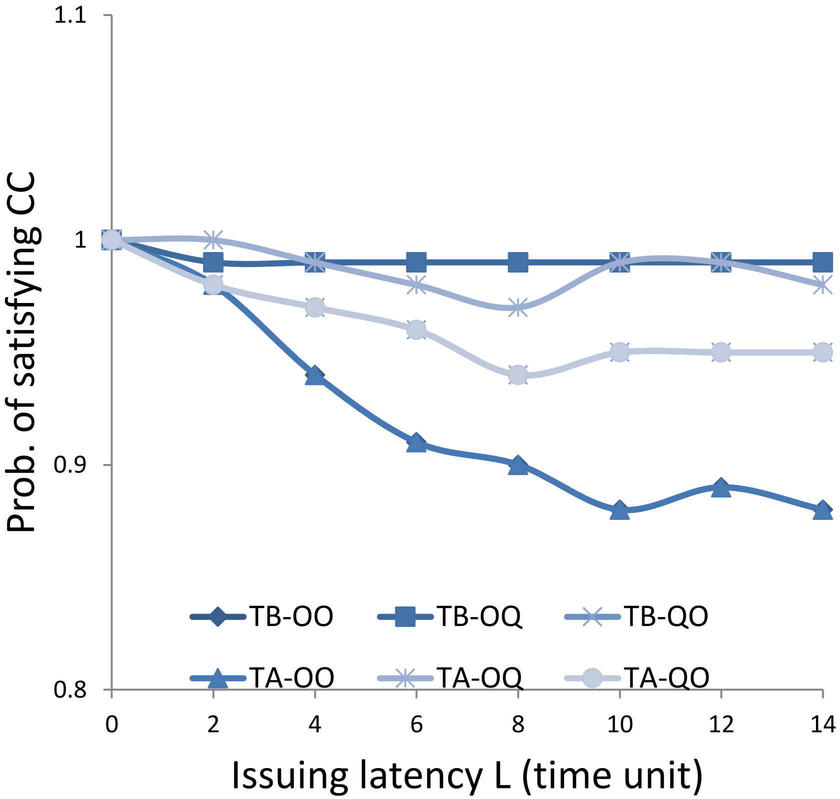


Statistical Model Checker

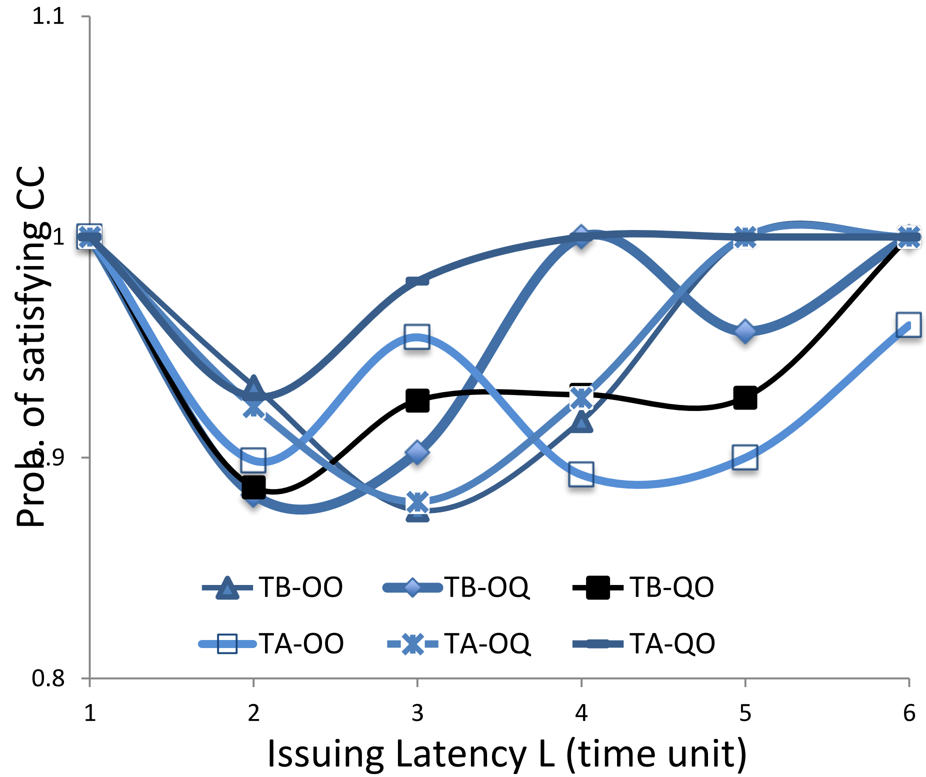
Real-deployed cluster

- **Conclusion: TB gains more CP consistency *only before* some issuing latency for** ALL **reads.**

Comparison: Causality



Statistical Model Checker



Real-deployed cluster

- **Conclusion: Both TA and TB achieve high CC consistency as expected; regarding** OQ**, TA's performance resembles TB's and both surpass each other slightly for some points.**

Comparison & Conclusion

- The resulting trends from both sides are similar, leading to the same conclusions with respect to consistency measurements and strategy comparisons
- Our Cassandra model achieves much higher consistency (up to SC) than the promised EC, with **QUORUM** reads sufficient to provide up to 100% consistency in almost all scenarios
- TA is not a competitive design alternative to TB in terms of those consistency models *except* CP, even though TA behaves close to TB in most cases

- TA surpasses TB with **ALL** reads during a certain interval of issuing latency

Summary

- First formal and executable model of Cassandra
  - Captures all major design decisions
- Predicting Consistency behavior by using Statistical Model-Checking
- Statistical Model-checking matches reality (deployment numbers)
- Our work best to predict Cassandra Performance
  - Faster than simulations
  - Less work than full deployment
  - Repeatable

Ongoing and Future Work

- Other performance metrics

  - throughput, latency

- Model other NoSQL or transactional systems

  - RAMP already modeled & analyzed (SAC 16')

- Build a library of building blocks

  - Mix and match to generate any NoSQL or transactional system with desired consistency and availability trade-offs