CS848 Paper Presentation
Design and Evaluation of a Continuous Consistency Model for Replicated Services

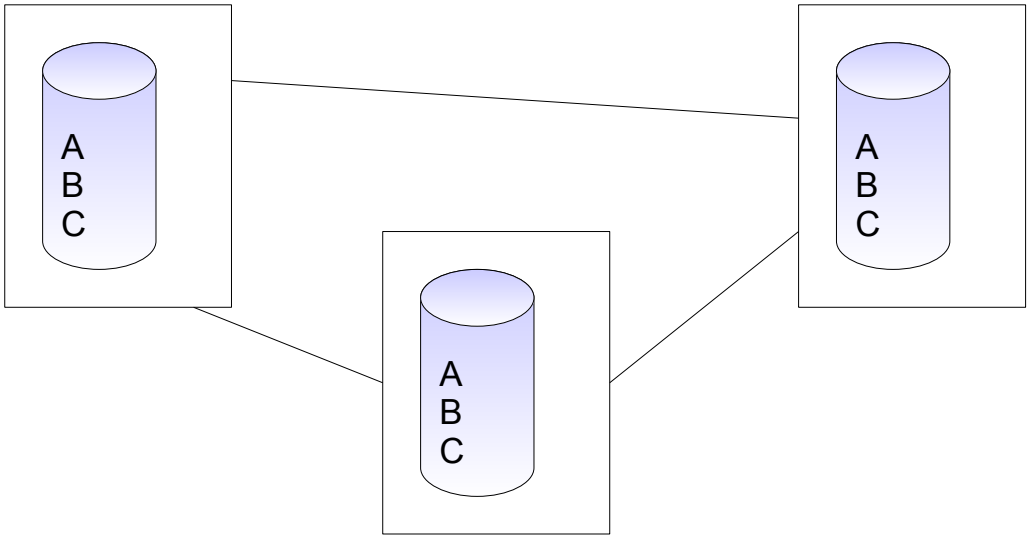
Yu, Vahdat
Duke University

Presented by Brian VanSchyndel

David R. Cheriton School of Computer Science
University of Waterloo

25 January 2010

Scenario: Distributed database with multiple replicas



- Multiple database servers connected by network
- Not partitioned

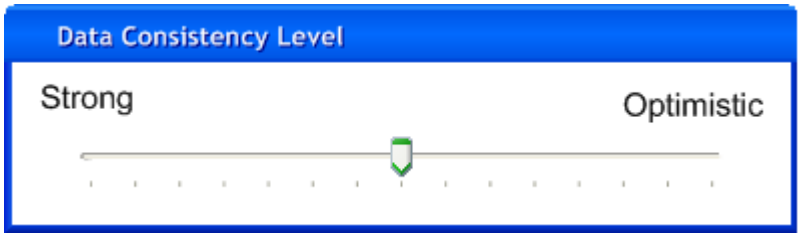
Motivation

- Optimistic consistency models typically provide **no** bounds on the inconsistency of the data

Motivation

- Optimistic consistency models typically provide **no** bounds on the inconsistency of the data

- Purpose of the paper:
  - Investigate the continuum between **strong** and **optimistic** consistency



Goal (1)

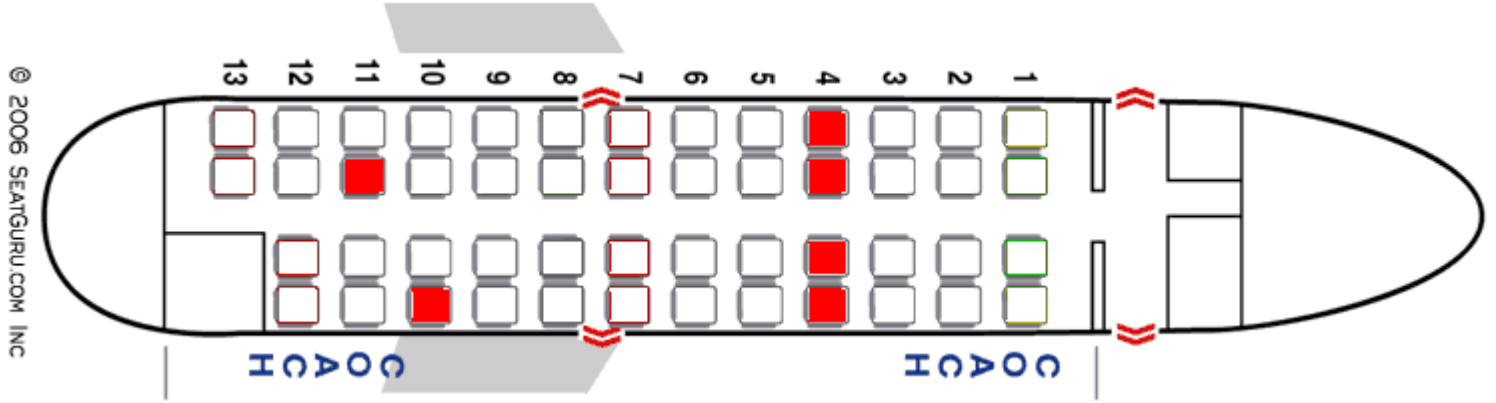
- Understand data consistency by using concrete examples:
  - Airline Reservation System
  - News System
  - Load Balancing System

- Consistency: "Closeness" of data among replicas

Consistency: Airline Reservation System

Operations:

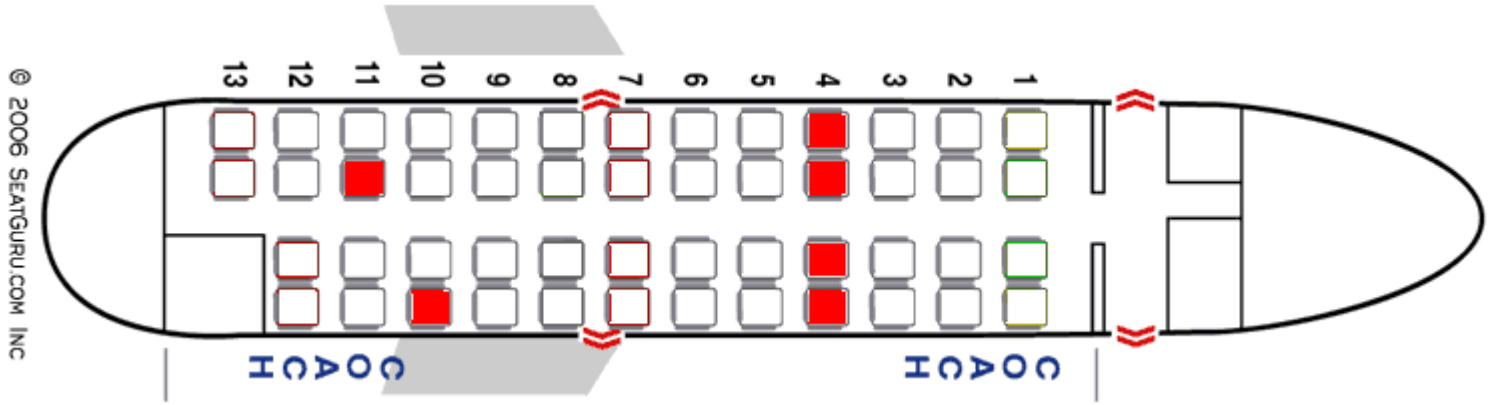
- Query seat availability
- Reserve a random seat on the plane



Consistency: Airline Reservation System

Consistency:

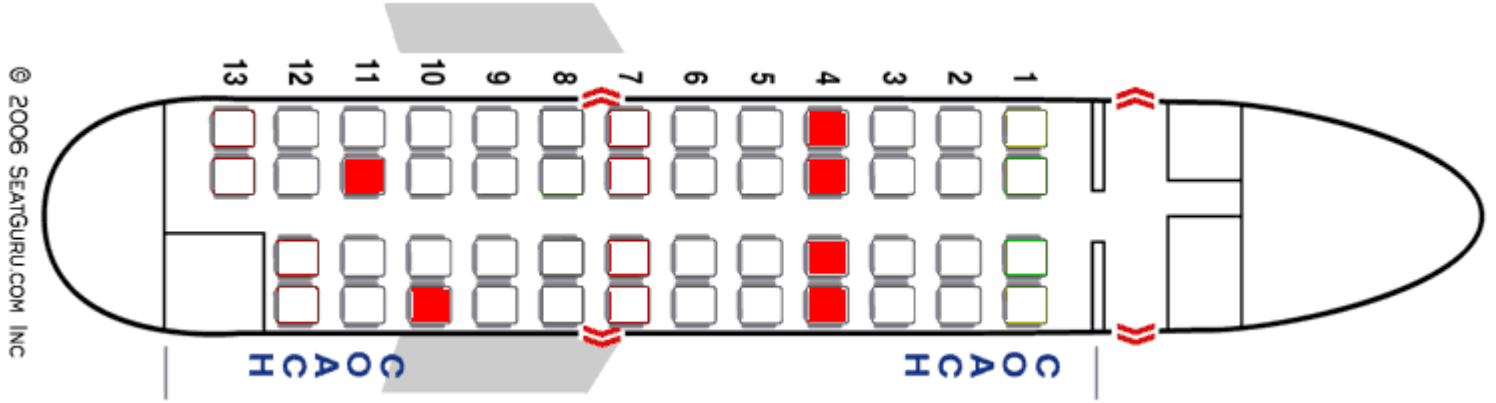
- Seat states { Reserved, Available }
- Seats have same state among replicas



Consistency: Airline Reservation System

Consequences of inconsistency:

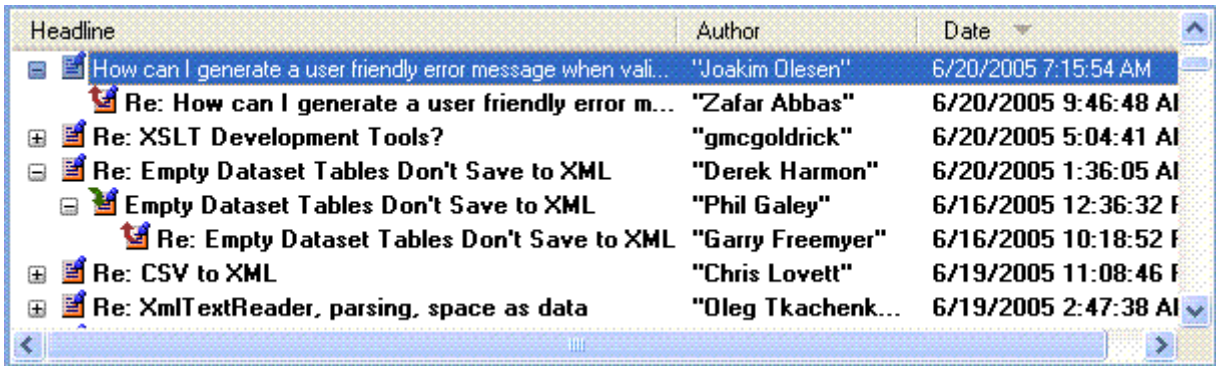
- Query returns incorrect locations of available seats
- Query returns incorrect number of available seats
- Reservation conflict, so:
  - Automatically reserve a different seat
  - Revoke reservation if no more seats available



Consistency: News System

Operations:

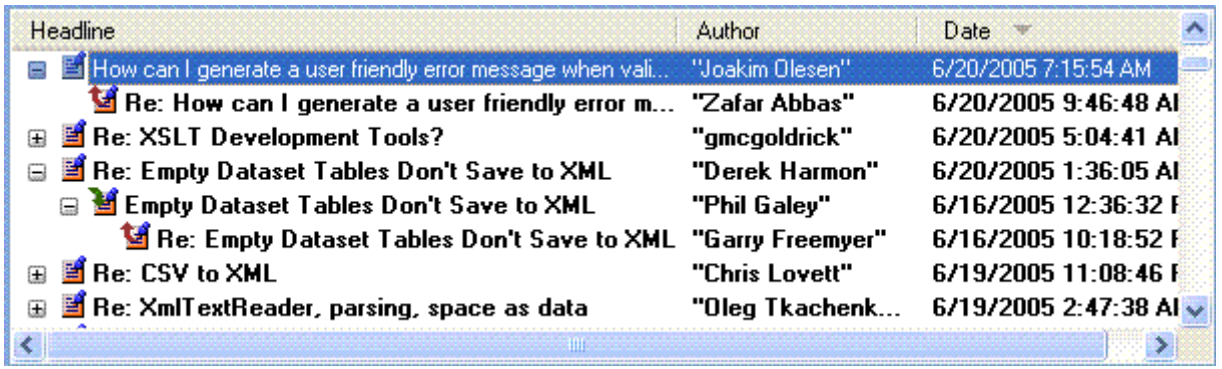
- Post new message
- Post a reply



Consistency: News System

Consistency:

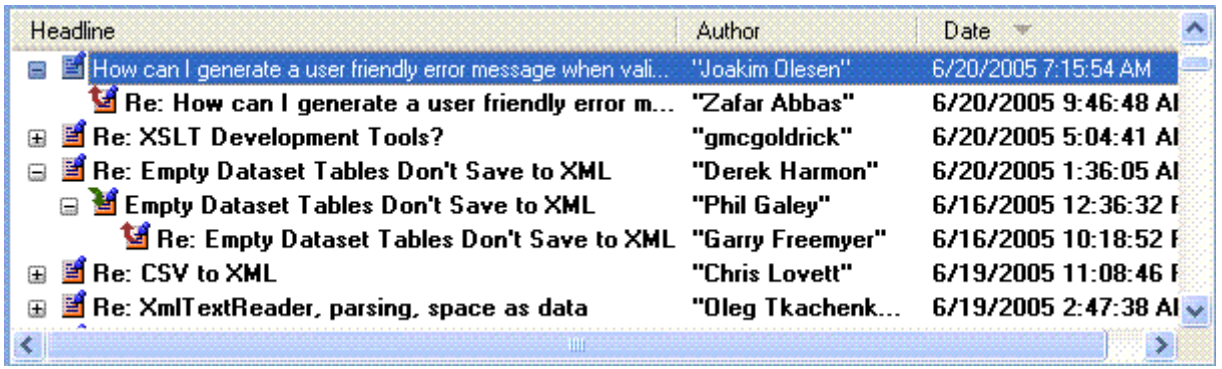
- Messages appear on all replicas
- Replies appear after original message
- Message threads appear in the same order on all replicas



Consistency: News System

Consequences of inconsistency:

- Confusion (messages of discussions are randomly ordered)
- Incomplete information (missing messages)



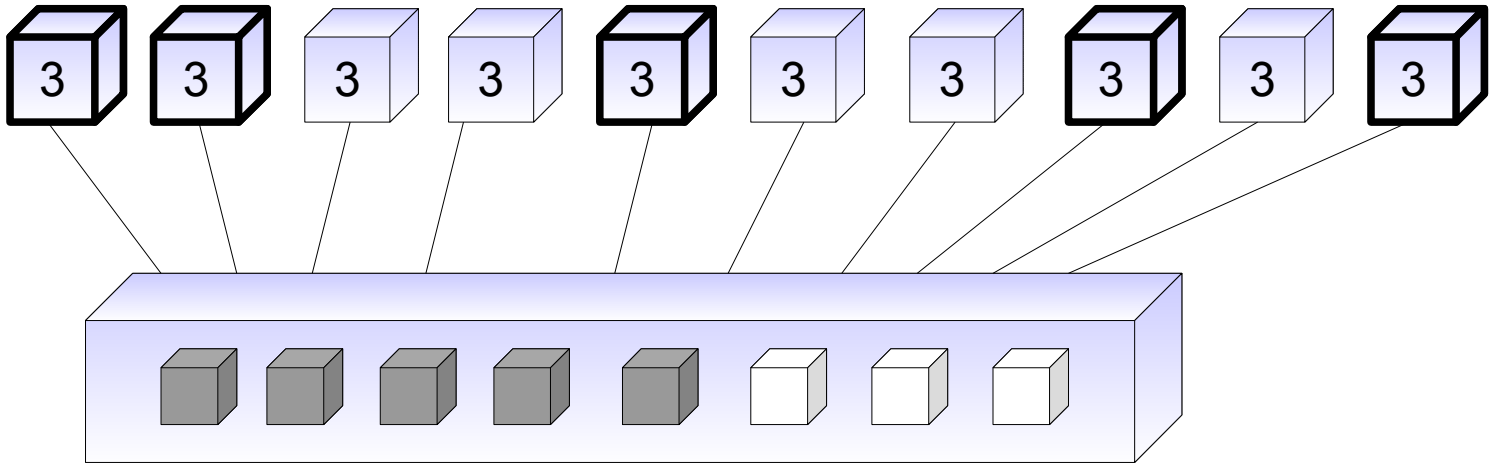
Consistency: Load Balancing System

Operations:

- Preferred client requires “service”
- Standard client requires “service”

Consistency:

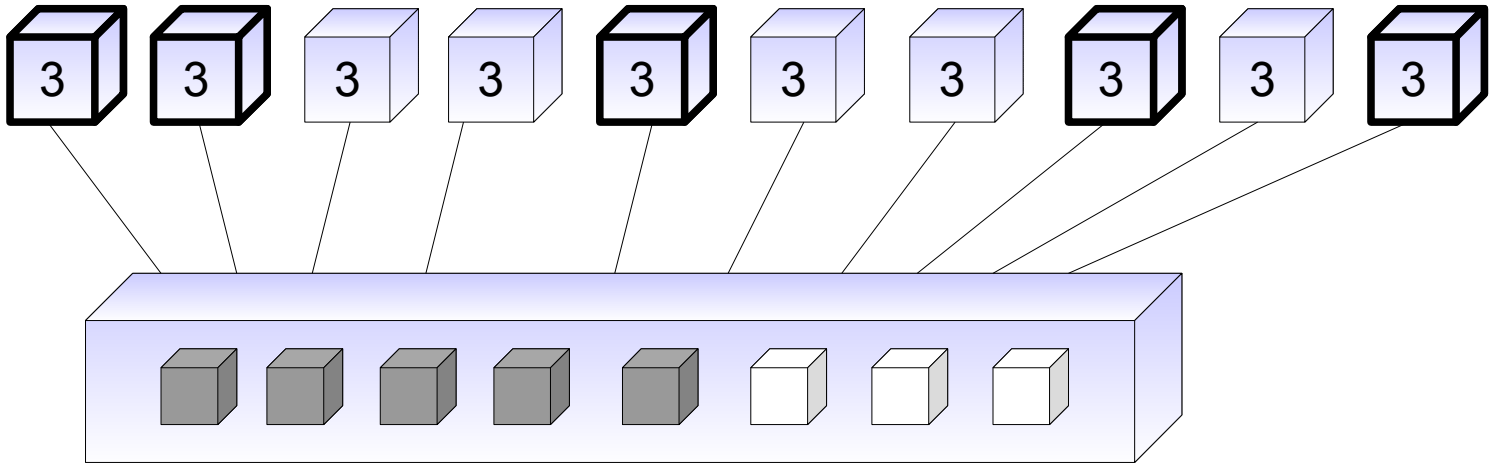
- Perceived available server capacity same among replicas



Consistency: Load Balancing System

Consequences of inconsistency:

- Server becomes overloaded when a client thinks the server is available
- Client waits for an idle server to become available when client thinks the server is too busy



“Conit”

Definition: Unit of consistency

- The data that is bounded by the configured “level of consistency”
- Consistency of a conit is the “closeness” between the same conit on different replicas.

“Conit”

Definition: Unit of consistency

- The data that is bounded by the configured “level of consistency”
- Consistency of a conit is the “closeness” between the same conit on different replicas.
- e.g. Flight Reservation System – all the seats on the plane
- e.g. News System – all messages in a newsgroup
- e.g. Load Balancing System – server capacity

“Conit”

Definition: Unit of consistency

- The data that is bounded by the configured “level of consistency”
- Consistency of a conit is the “closeness” between the same conit on different replicas.
- e.g. Flight Reservation System – all the seats on the plane
- e.g. News System – all messages in a newsgroup
- e.g. Load Balancing System – server capacity
- Conits should be big enough to keep the number of guarantees about the level of consistency of the database manageable.
- Conits should be small enough so inconsistencies among unrelated data does not affect another conit's performance.

Goal (2)

- Quantify a level of data consistency for an individual conit
- 3 metrics:
  - Numerical error bound
  - Order error bound
  - Staleness bound

Metrics: Numerical Error

- Definition: Total "weight" (importance) of writes that the replica has not seen
  - e.g. 2 unseen writes with a weight of 200 is more important to propagate versus 50 unseen writes with a weight of 5
  - e.g. weight = priority of a newsgroup message
  - e.g. weight = number of shared resources unconsumed by clients
  - e.g. $C_{system} = 5$, $C_{replicaA} = 2 \rightarrow$ Numerical Error = 3

Metrics: Numerical Error

- Definition: Total "weight" (importance) of writes that the replica has not seen
  - e.g. 2 unseen writes with a weight of 200 is more important to propagate versus 50 unseen writes with a weight of 5
  - e.g. weight = priority of a newsgroup message
  - e.g. weight = number of shared resources unconsumed by clients
  - e.g. $C_{system} = 5$, $C_{replicaA} = 2 \rightarrow$ Numerical Error = 3
- **Absolute error**: Difference between actual and perceived weight
- **Relative error**: Difference between actual and perceived weight as a percentage of actual weight

Metrics: Numerical Error

- Higher bound on numerical error → Better performance
  - Less frequent syncing between replicas

- Difficult to know the numerical error at any given time
  - Need to know the perceived and actual weight of writes of other replicas
  - Getting weights from other replicas requires data transfers which is what we are trying to restrict

Metrics: Order Error

- Definition: Total number of tentative writes
  - Recall that tentative (un-committed) writes are subject to re-ordering

- Higher bound on order error → Better performance
  - Less frequent syncing between replicas
  - Less frequent re-ordering of tentative writes
  - But more tentative writes need to be re-ordered each time

Metrics: Staleness

- Definition: Real time required to "see" a write that occurred on a remote replica

- Higher bound on staleness → Better performance
  - Less frequent syncing between replicas

Goal (3)

- Understand how to set the bounds on data consistency metrics with respect to concrete examples
  - Airline Reservation System
    - Numerical Error
    - Order Error
    - Staleness
  - News System
    - Numerical Error
    - Order Error
    - Staleness
  - Load Balancing System
    - Numerical Error

Bounds: Airline Reservation System

- Numerical Error
  - Affects *Reservation Conflict Rate* because conflict rate is inversely proportional to the number of unseen reservations
  - Weight: Seat reservation = 1
  - Formula derived for calculating *Reservation Conflict Rate* as a function of the Numerical Error bound

Bounds: Airline Reservation System

- Numerical Error
  - Affects *Reservation Conflict Rate* because conflict rate is inversely proportional to the number of unseen reservations
  - Weight: Seat reservation = 1
  - Formula derived for calculating *Reservation Conflict Rate* as a function of the Numerical Error bound

- Order Error
  - Affects query results because tentative writes (reservations) may change due to conflicts

Bounds: Airline Reservation System

- Staleness
  - Affects query results because available seats may no longer be available

Bounds: Airline Reservation System

- Staleness
  - Affects query results because available seats may no longer be available

- Dynamic Factors
  - Preferred vs. Standard clients may demand higher consistency
  - Network capacity may be good enough to have high performance AND high consistency
  - Reservation Conflict Rate gets higher as seats are reserved
    - Want strong consistency for issuing the last available seat to avoid revoking many issued tickets

Bounds: News System

- Numerical Error
  - Affects number of unseen messages
  - Weight: Each message = 1
- Order Error
  - Affects order of messages (reply/original, multiple threads)
- Staleness
  - Affects the delay that a message posted on another replica takes to appear on your replica

Bounds: News System

- Dynamic Factors
  - Important messages require a higher numerical weight in order to force their propagation sooner

Bounds: Load Balancing System

- Numerical Error
  - Affects accuracy of perceived current server capacity
  - Weight: Each request = 1, Each return = -1
- Order Error
  - Doesn't matter because summation of the counter is commutative
- Staleness
  - Doesn't matter because there is no added benefit

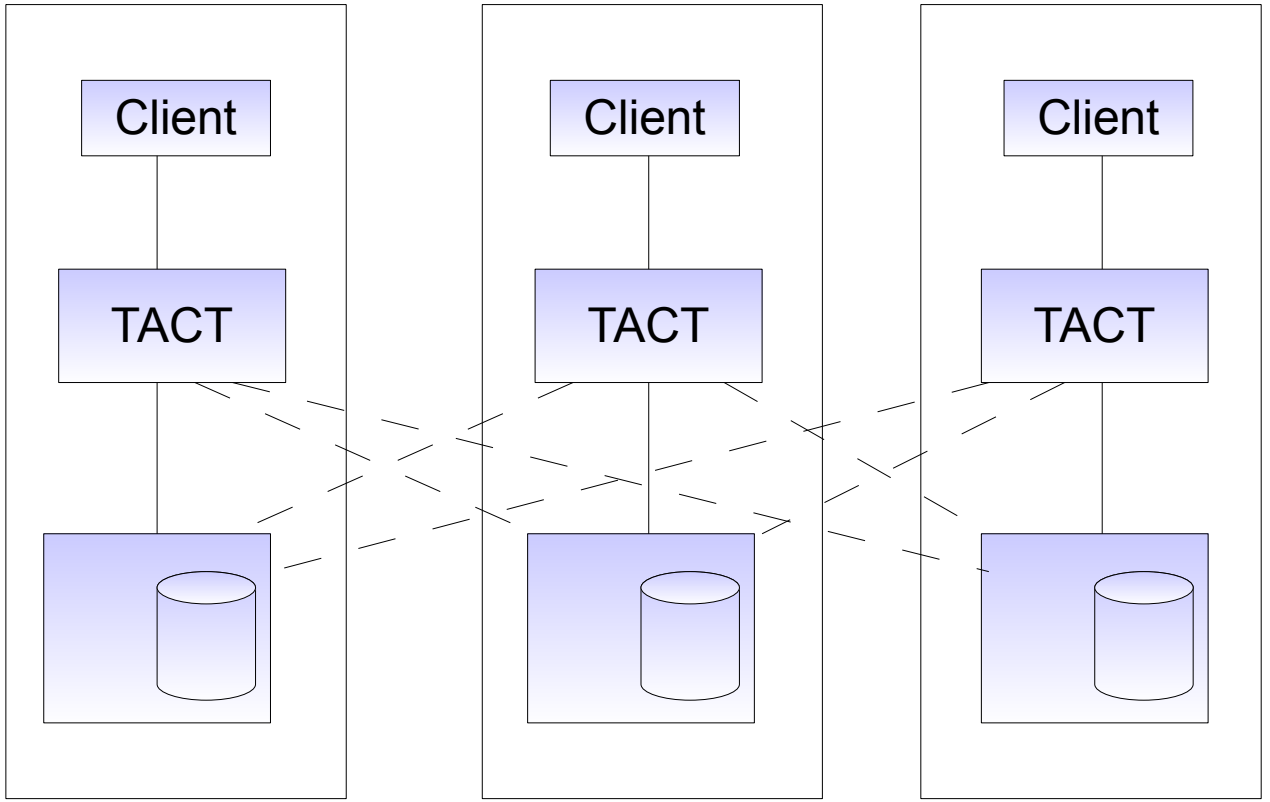
Bounds: Optimization

- What are the consequences of write conflicts?
- What are the consequences of incorrect reads?
- Acceptability depends on system requirements
  - Loss of customers, reduced revenue, broken agreements and laws, etc.
- All factors have tradeoffs
- Use probabilistic formulas to identify good choices
- Test various combinations of consistency bounds and compare resulting performance and consequences

Goal (4)

- Understand the TACT (Tunable Availability and Consistency Tradeoffs) implementation

TACT



- Middleware layer between client application and replicated data store

TACT

- Replica synchronization doesn't happen without the approval of TACT
- Synchronization uses anti-entropy exchanges
- Each replica-conit-request is configurable by its own consistency bounds (Numerical Error, Order Error, Staleness)
  - Very fine configurability

TACT

- When none of the consistency requirements are violated the local data store is used (high performance)
- When a consistency requirement is violated (too inconsistent) the client waits for local data store to sync with other replicas so consistency requirements are met (lower performance)
- Syncing also takes place at arbitrary “optimal” times
- Bounds of 0 → strong consistency
- Bounds of ∞ → optimistic consistency

TACT

- Maintaining Numerical Error bound
  - Estimate other replica's Numerical Error by estimating the total weight we have kept secret from each replica
  - Infer total weight of each replica based on patterns of the replica
  - Requires consensus algorithm or approximation algorithm (overhead)
  - Push local data to other replicas to ensure other replicas are aware of our writes

TACT

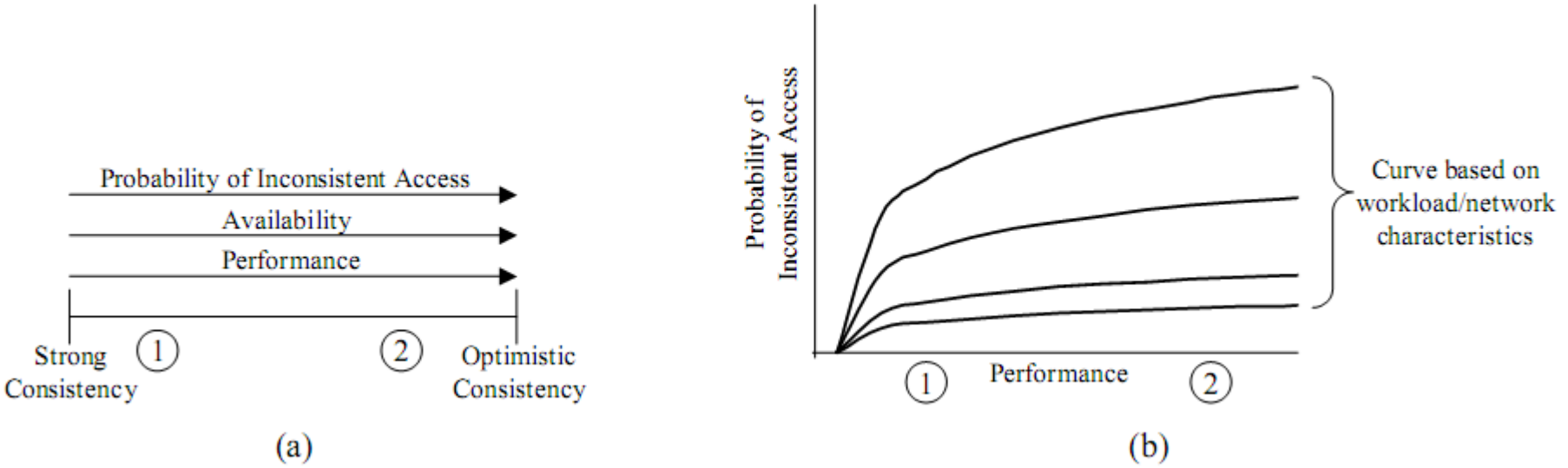
- Maintaining Order Error bound
  - When our number of tentative writes reach the limit we pull data from other replicas in order to commit our writes

- Maintaining Staleness bound
  - When the **current time – last update time** reaches the limit of staleness for a replica we pull data from the replica

TACT Experiments

- Ran many operations from the examples
  - Flight Reservation System, News System, Load Balancing System
- Used WAN communication to ensure syncing >> local re-ordering and merging
- Measured latency of operations

TACT Evaluation



- The rate of performance-increase with respect to consistency-decrease depends on the application
  - Workload; Read/write ratios; Probability of simultaneous writes; Network latency, bandwidth, error rates; etc.
- All results were positive (bounded consistency → bounded performance)