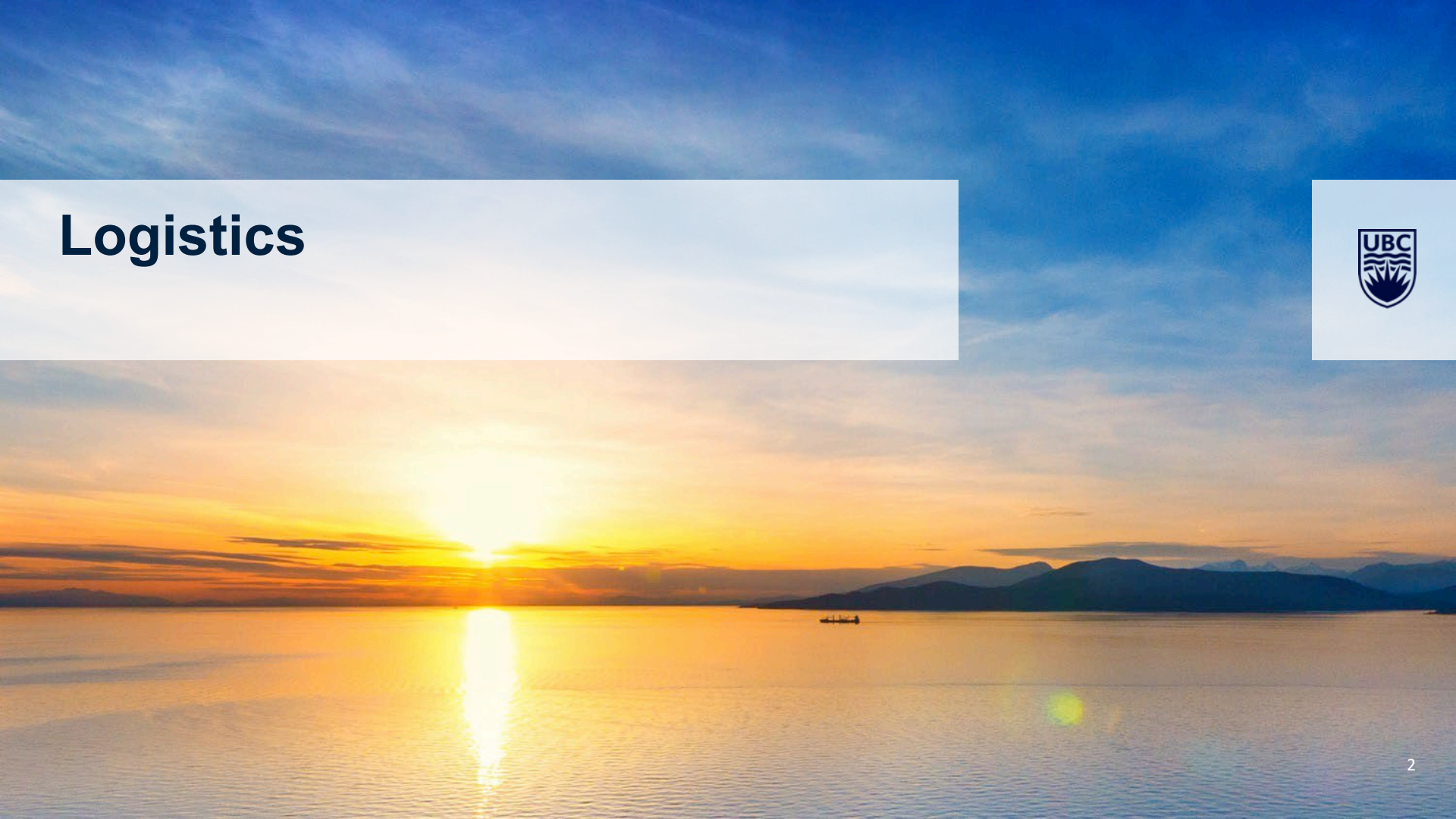
CPSC 416 Distributed Systems

Winter 2022 Term 2 (March 9, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer

Logistics



Deadlines

Project 4 Released. Initially Due: March 13, 2023. Late Due: April 13, 2023.

Project 5 Released Due: April 13, 2023

All project work is due April 13, 2023. Late projects are scaled to 75% of the on-time max.

Note: We switch from PST to PDT on March 12, 2023 at 02:00 (so 02:00 PST = 03:00 PDT). You have one hour less than you probably thought you had.

Deadlines

Alternate Path 1 & 2: Review in progress

- Piazza private threads need TLC
  - **Weekly updates due each Monday @ 23:59 PT**
- Final reports due no later than Thursday April 13, 2023 @ 23:59 PT

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00

TA Office Hours:

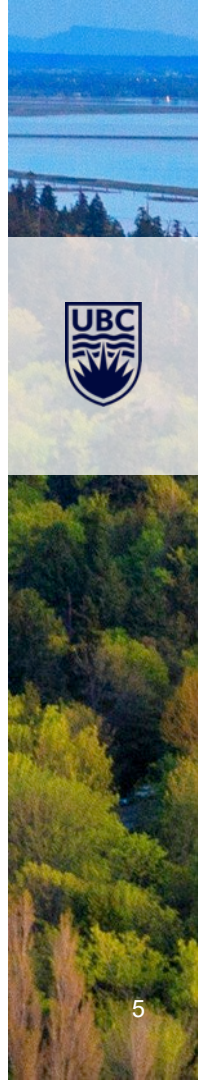
- Eric: Friday 9-11 am (in-person and Zoom)
- Japraj: Wednesday 3-5 pm (Zoom)
- Yennis: Thursday 2-4 (Zoom), Friday 2-4 (in-person)

Readings

Required:

Recommended:

- Scaling Memcache at Facebook
- TAO: Facebook's Distributed Data Store for the Social Graph
- Costs and Limits of Availability for Replicated Services

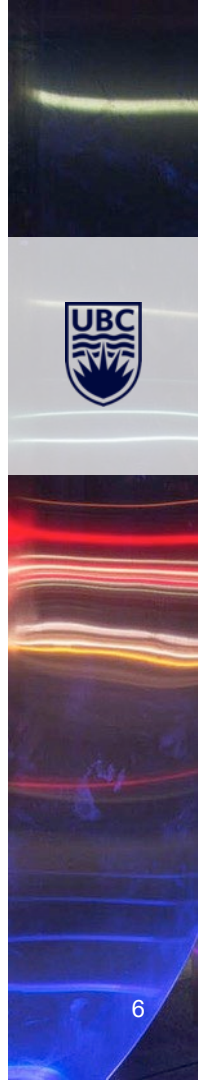


Questions?

Questions about the class?

Questions about the previous lecture?

Funny stories to share?



Today’s Failure

Today's Failure: Case Study (Airline Failure)

Source: *Release It!: Design and Deploy Production-Ready Software*

Case Study: The Exception That Grounded an Airline

Scenario: Planned failover on the database cluster serving Core Facilities (CF)

Environment:

- J2EE Application Server cluster (high availability)
- Replicated storage (RAID disk arrays)
- Twice daily offsite backups
- Redundant Oracle Database
- Redundant network load balancers
- Backup facility in physically separate data center

Today’s Failure: Case Study (Airline Failure)

Thursday evening, 11 pm PT

- Routine maintenance
- Switch from primary to backup
- IP address switch: transparent to applications
- Apply change to (old) primary
- Switch back: (old) primary becomes primary again
- Update backup

Friday morning, 00:30 am PT

- Maintenance complete

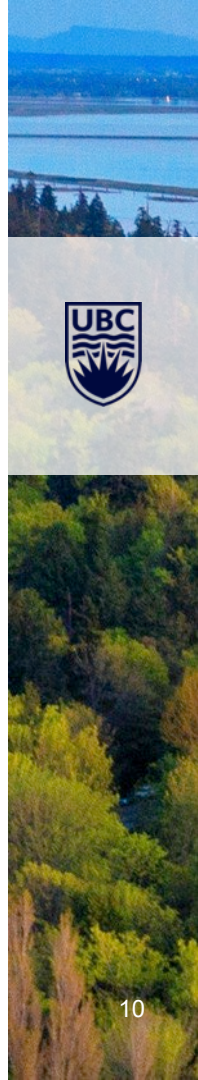
Today's Failure: Case Study (Airline Failure)

Thursday evening, 11 pm PT

- Routine maintenance
- Switch from primary to backup
- IP address switch: transparent to applications
- Apply change to (old) primary
- Switch back: (old) primary becomes primary again
- Update backup

Friday morning, 00:30 am PT

- Maintenance complete



Today's Failure: Case Study (Airline Failure)

Friday morning 02:30 PT

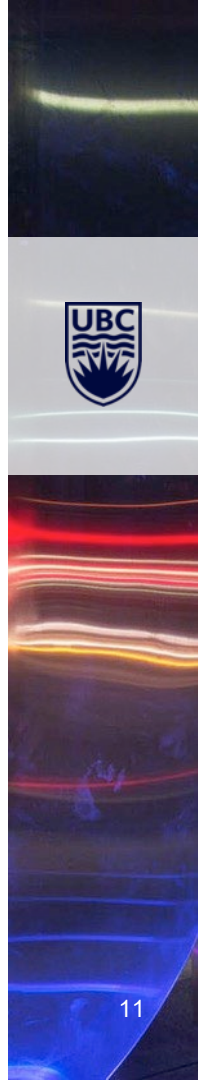
- Airline check-in kiosks stop working. **All of them, everywhere**.
- This is start of day for flights ET. **Code red**

Suspect: the Core Facilities (CF) database switchover.

SLA: **One hour**

Restart Core Facilities: kiosks still red

Restart kiosk application servers: *problem resolved*



Today's Failure: Case Study (Airline Failure)

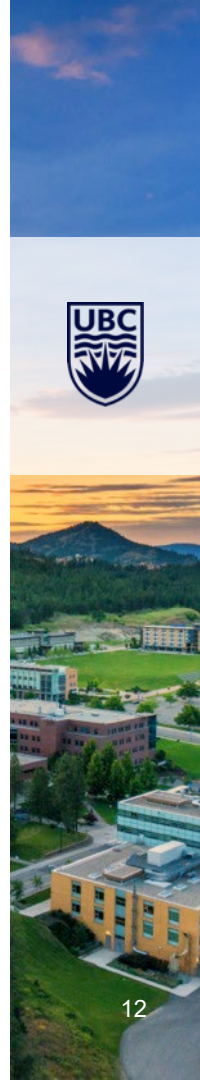
Post-mortem:

- Review logs
- Review configuration
- Review network configuration
- Review Java thread dumps

Applications were all hung waiting for a response

- Tracked to database connection from a resource pool
- JDBC experienced an exception due to database failover
  - Connection was "dead" but continued accepting commands
  - Commands blocked
  - Resource leak!0

Conclusion: A single uncaught SQLException brought down the *entire* system



Lesson Goals

Consistency in Distributed Data Stores

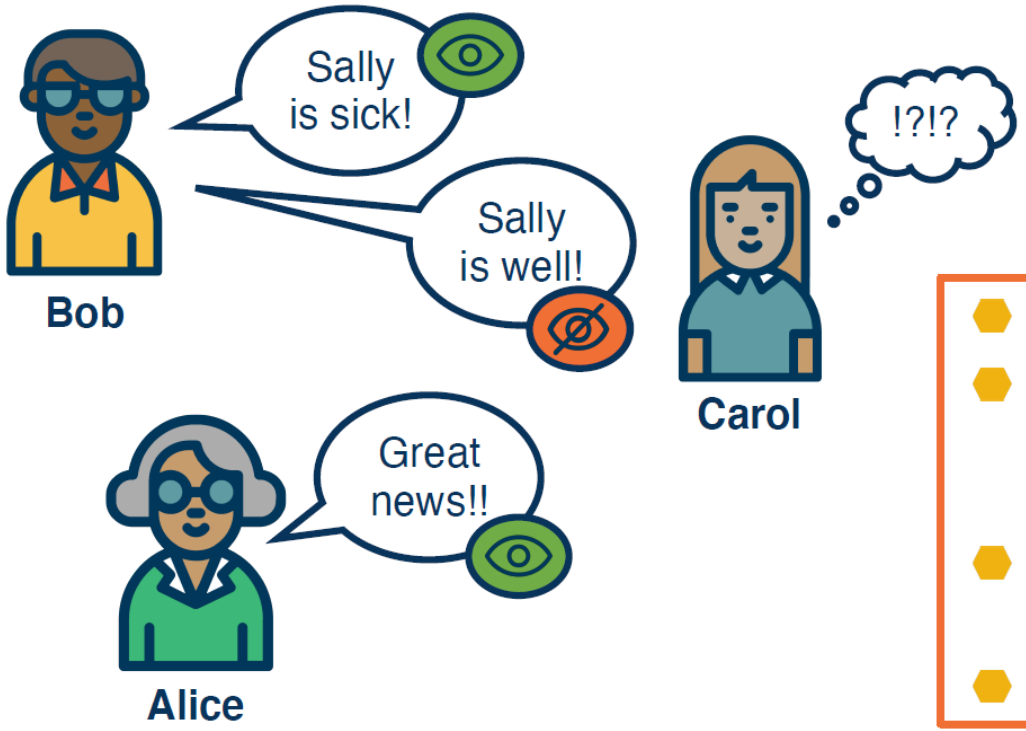
Practical considerations

Distributed data stores (key-value stores)

- Memcached at Facebook
- Tao at Facebook

Different consistency models

Consistency: Important and ChallengingS



- Distributed state
- Replication for availability and fault tolerance
- Caching for performance
- Failures

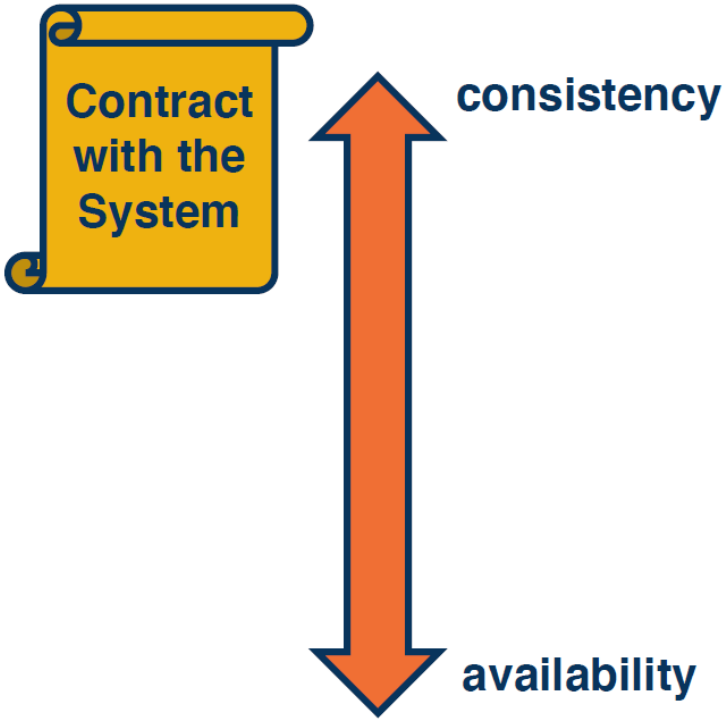
Consistency Models

Strong consistency, linearizability

Sequential consistency: guaranteed single ord

Causal consistency: enforce “happens before”

Eventual consistency: recover from transient e



Key-Value Store

Simple operations

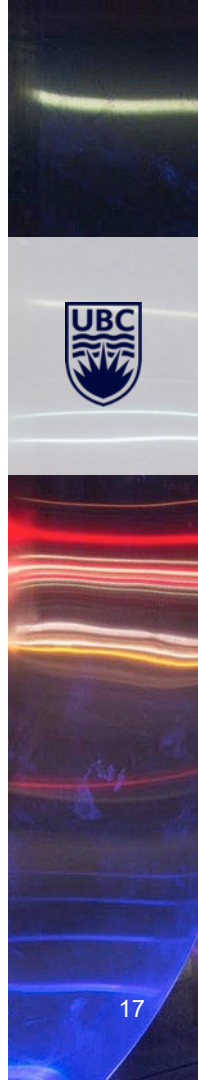
Mirror HTTP protocol

- PUT
- GET

Allows complex operations

- Append
- DeleteS

Key	Value
1	
2	



Memcached

Problem: Optimizing web page rendering

Solution: Cache commonly used objects

Memcached

- Simple key-value store
- Facebook

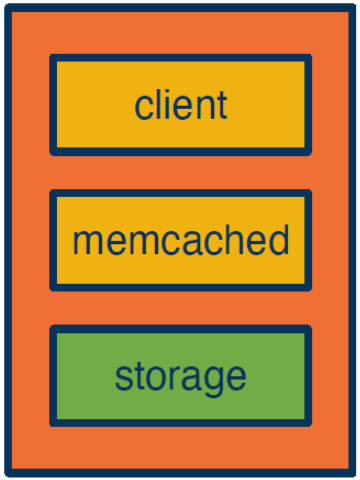
Replaced with Tao

- TaoD (FOSS implementation)

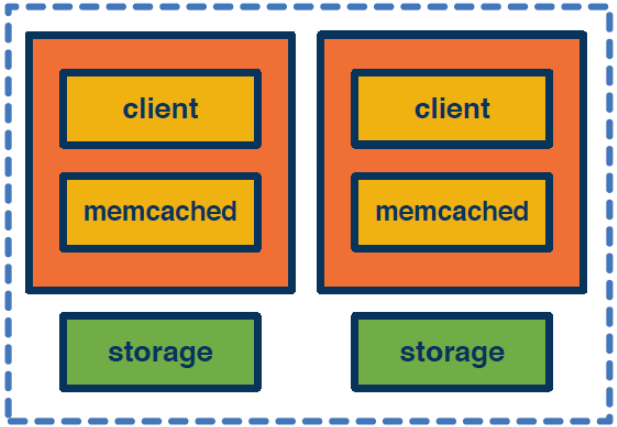
See also *Redis*

Memcached at Facebook

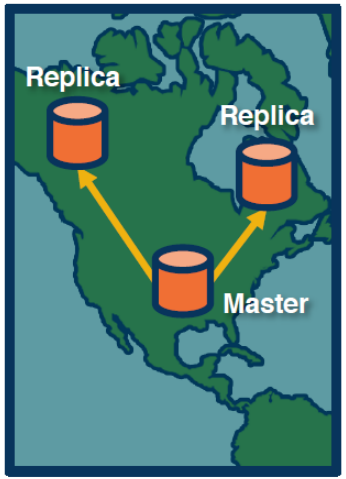
Within a Cluster



Across Clusters



Geographically



Memcached: Look-aside Cache

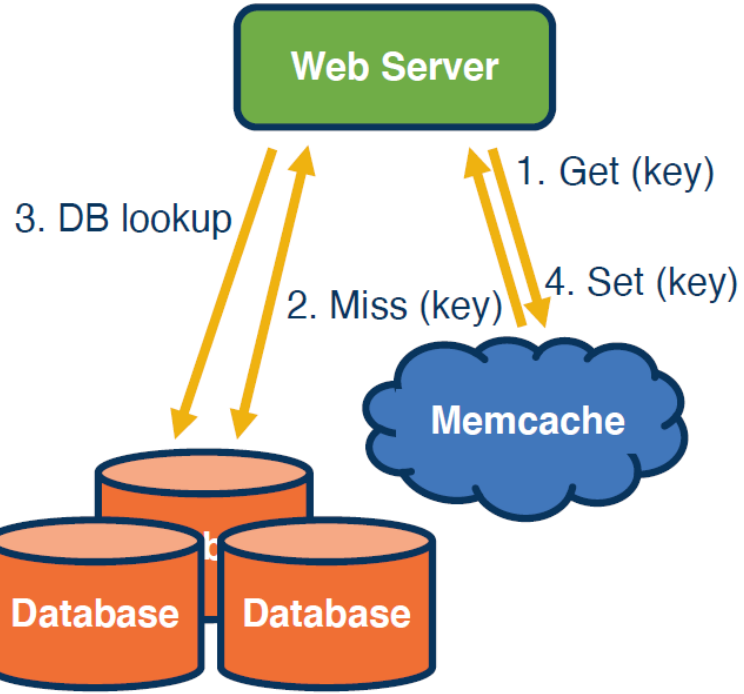
Workload

- Read intensive (100x writes)
- Large data capacity
- Hot data
- Temporal locality

Initial implementation: SQL Databases

Simple in-memory key-value object cache

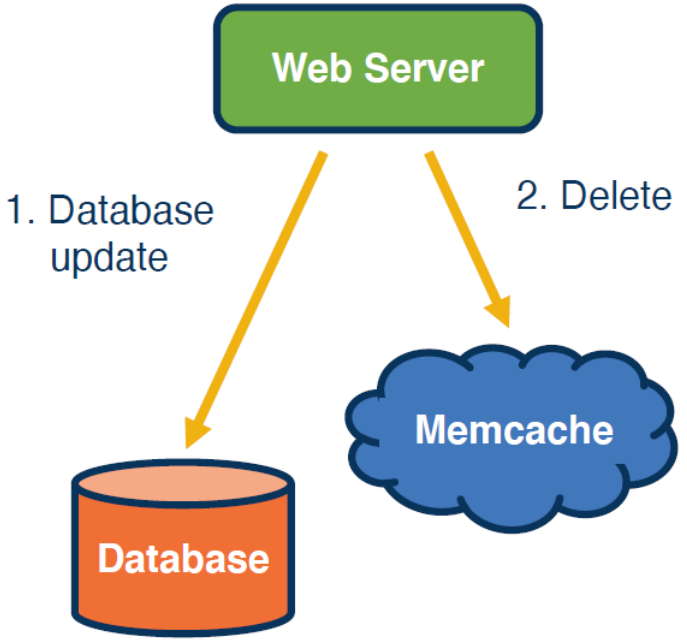
- Look-aside cache
- Demand-filled cache



Memcached: Cache Consistency

Simple in-memory KV object cache

- Look-aside cache
- Demand filled cache
- Database updates = cache deletion
- LRU cache recycling
- Multiple clusters: read-only data
- Non-authoritative

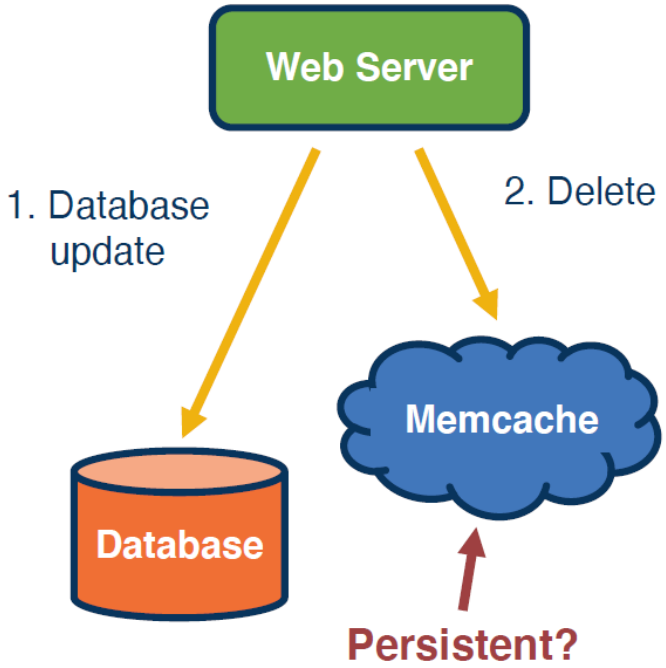


Memcached: Optimizations

Simple in-memory KV object cache

- Look-aside cache
- Demand filled cache
- Database updates = cache deletion
- LRU cache recycling
- Multiple clusters: read-only data
- Non-authoritative

Look-aside design allows optimizations



Memcached: Lessons

Problems:

- Stale Data
- Thundering Herd

Solution: Leases

- Issued on cache miss
- Detect concurrent writes
  - Ordering of writes maintained
- Flexible leases (client choice):
  - Allow stale (fast)
  - Require current (waits)

Memcached: Lessons

Scaling

- Use *consistent hashing* to distribute requests
  - Independent of number of servers
  - Allows dynamic reconfiguration
- All web servers talk to all memcached servers
  - Single user request might require accessing 100s of memcached servers

Congestion

- Problem: Shared networking resources can be overwhelmed
- Solution: Limit the number of outstanding requests with a sliding window
  - Big window means more congestions
  - Small window means more network round trips

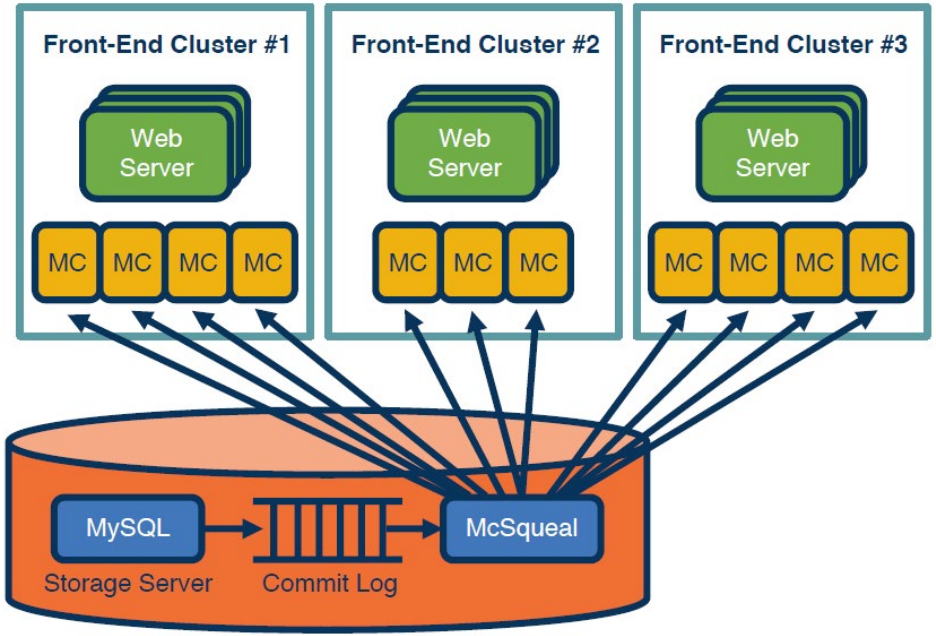
Memcached: Lessons

Multi-cluster

- Redundancy
- Separate failure domains

Multi-caching

- Database driven invalidati



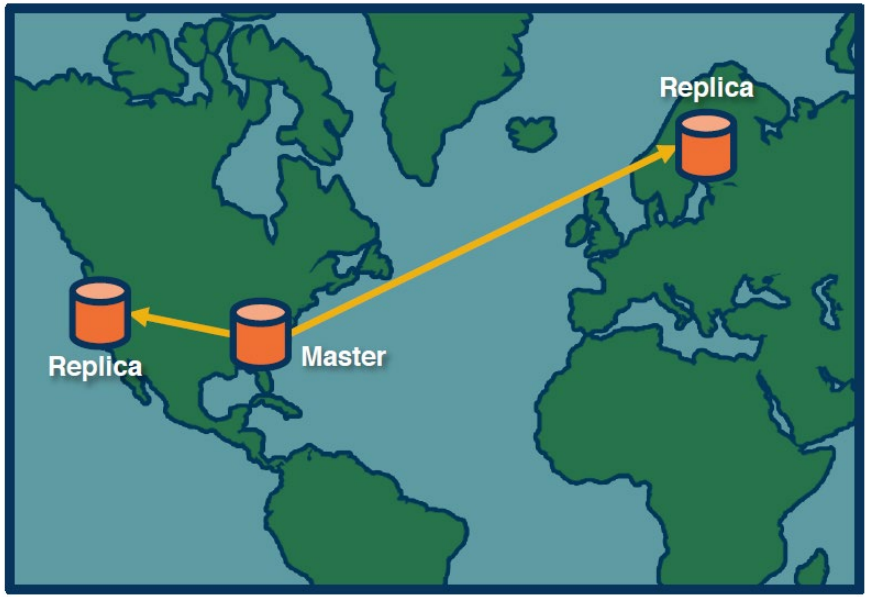
Memcached: Lessons

Problem: Multi-cluster consistency

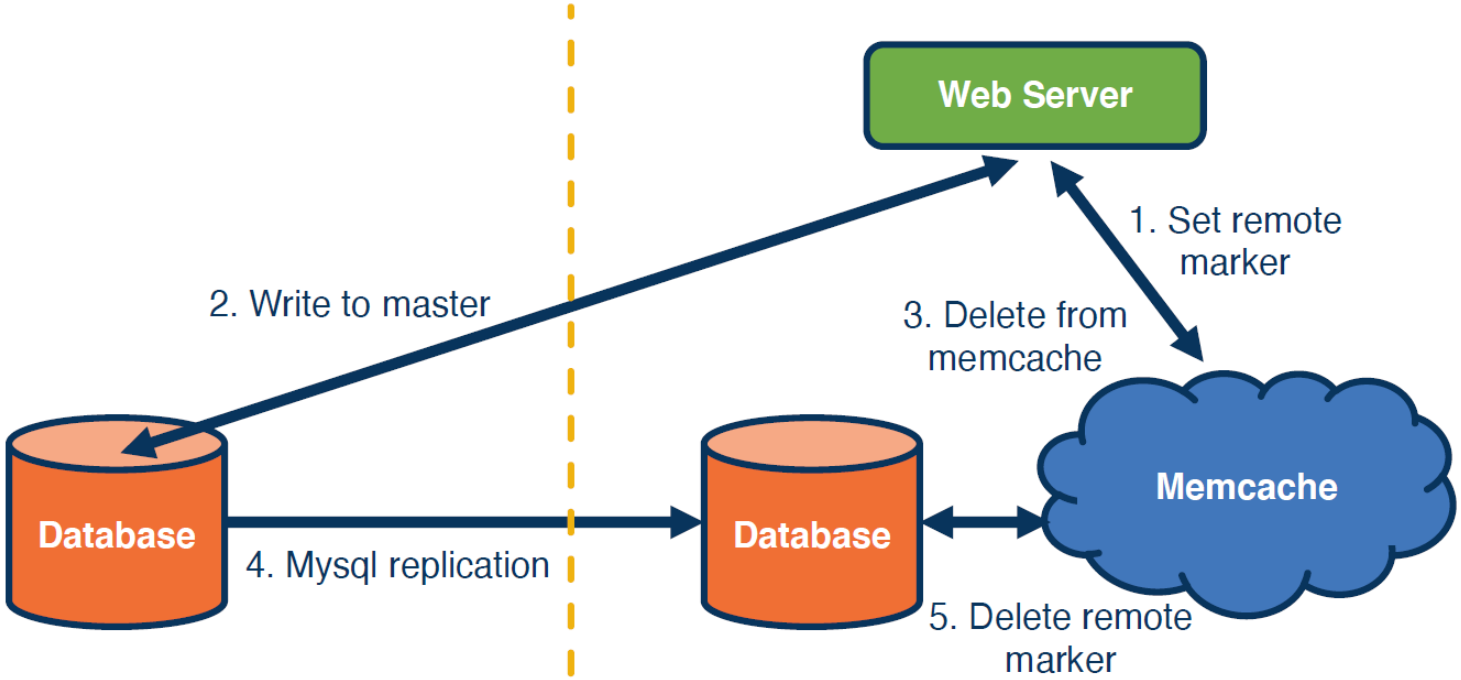
- Separate failure domains
- Does not scale geographically

Solution:

- Add storage layer replication

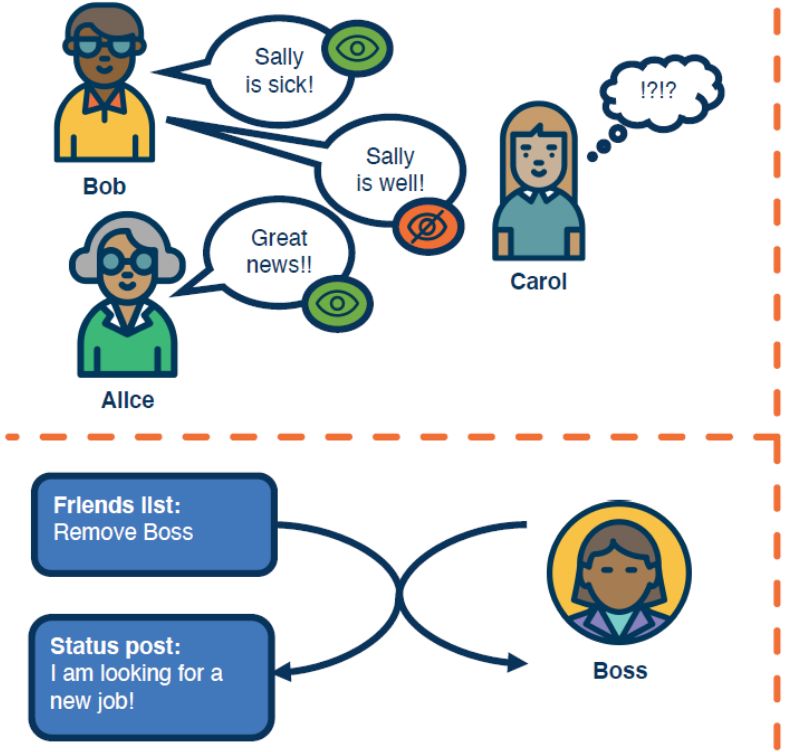


Memached: Lessons



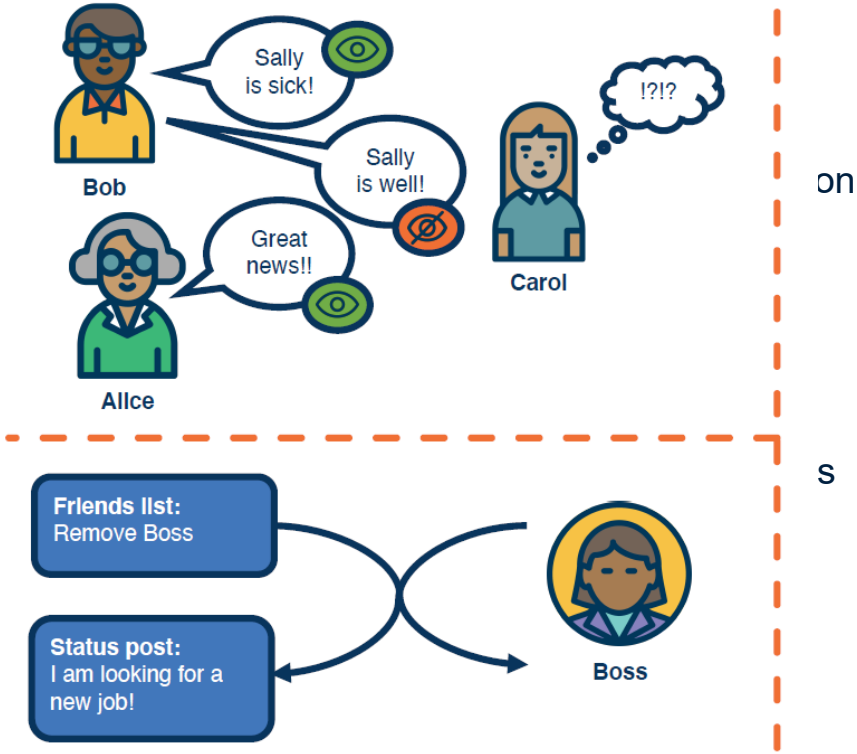
Consistency Models

Causal Consistency: Not Enough



ation

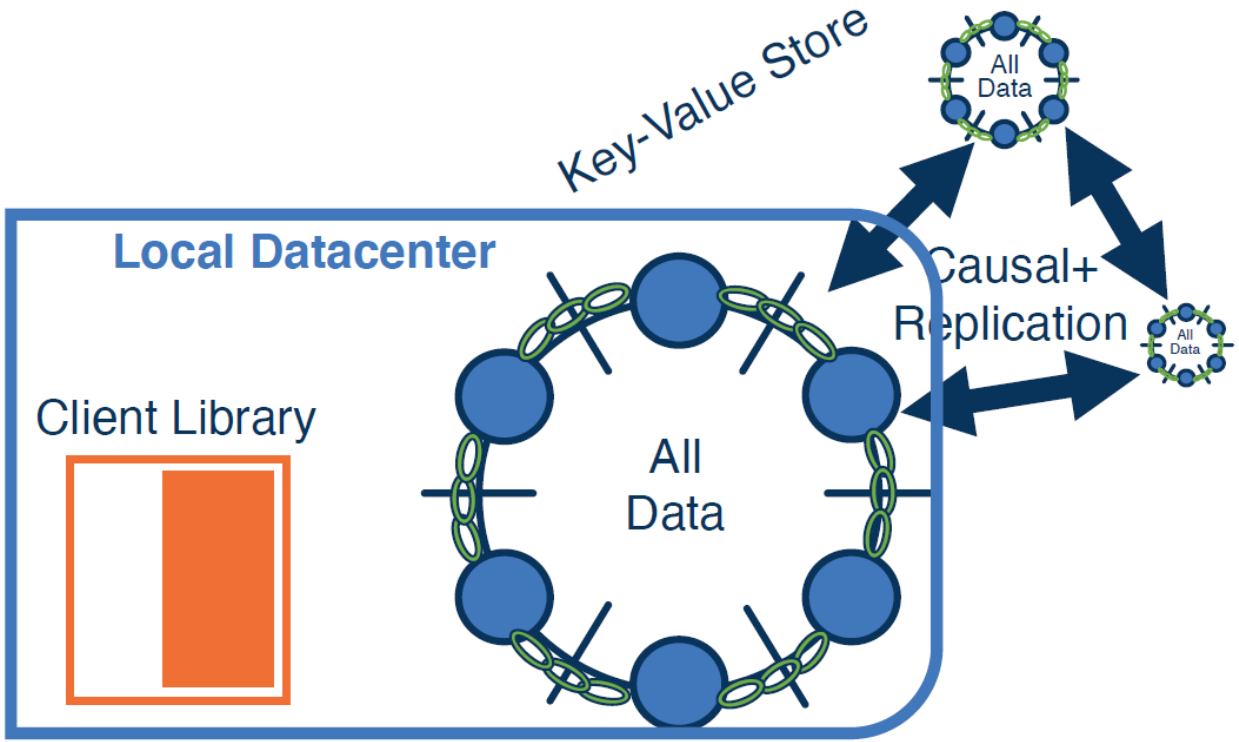
Causal Consistency: Not Enough



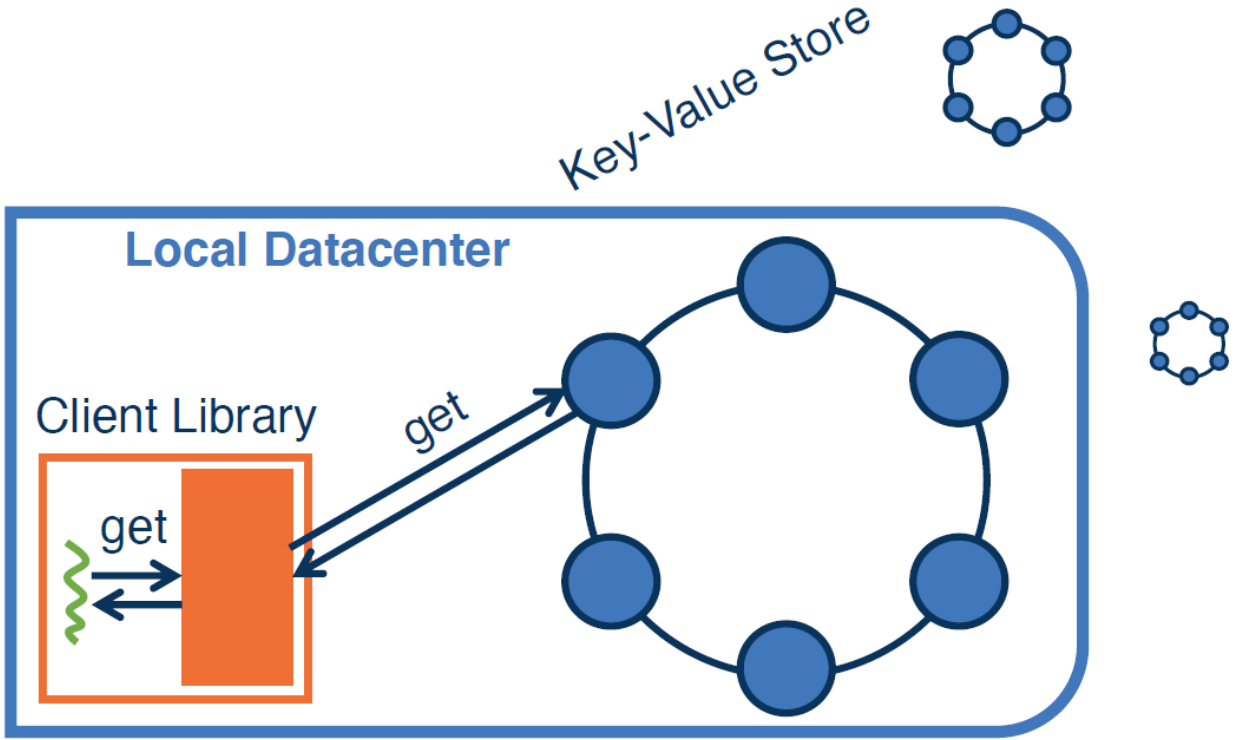
on

s

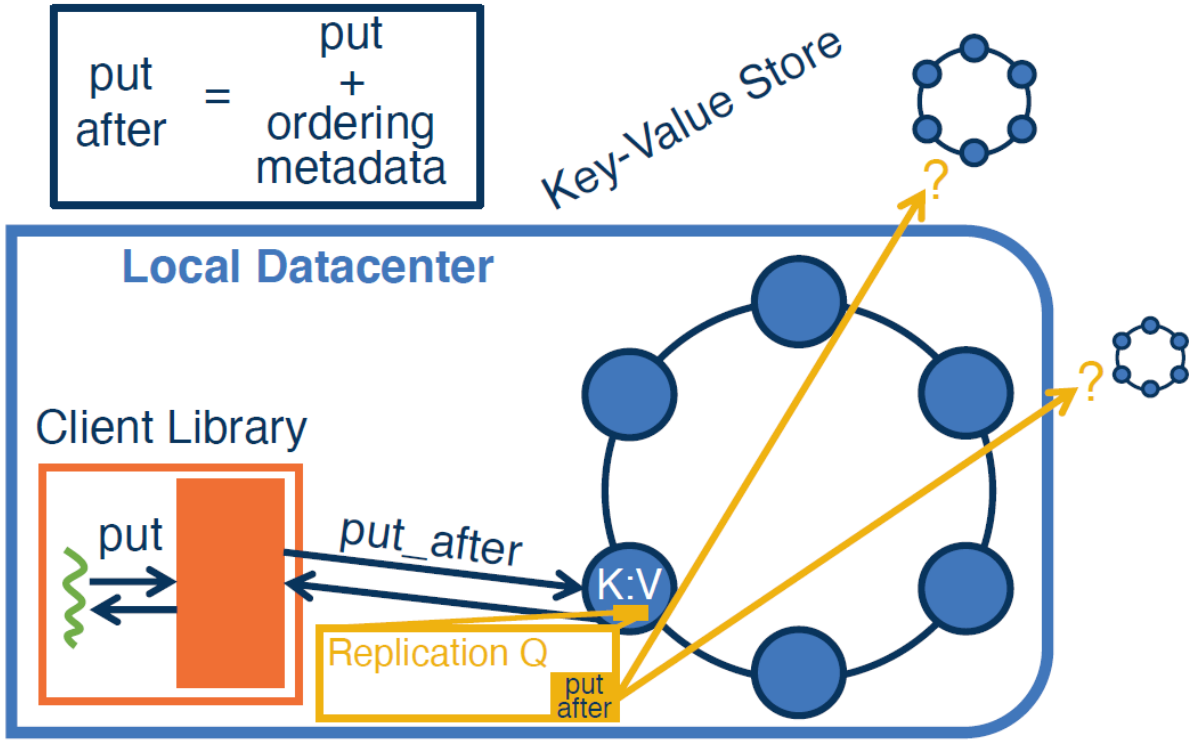
COPS: Wide Area Causal Consistency



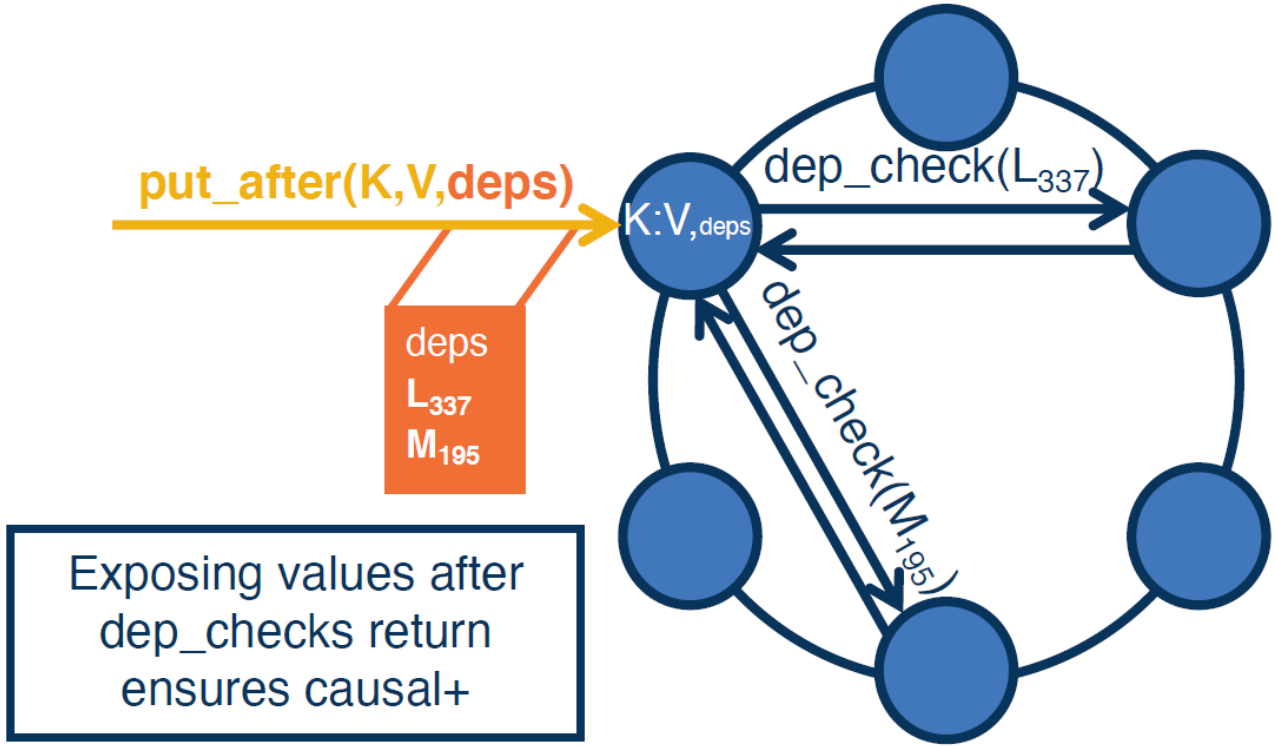
COPS: Get operation



COPS: Put operation



COPS: Causal+ Replication



Exposing values after dep_checks return ensures causal+

Lesson Review

Consistency:

- Models
- Tradeoffs
- Techniques

Memcached

- Architecture
- Design Decisions

Causal+ Consistency with COPS

Questions?


UBC
THE UNIVERSITY OF BRITISH COLUMBIA