

# **FAULT-TOLERANCE IN LARGE DATA SYSTEMS**

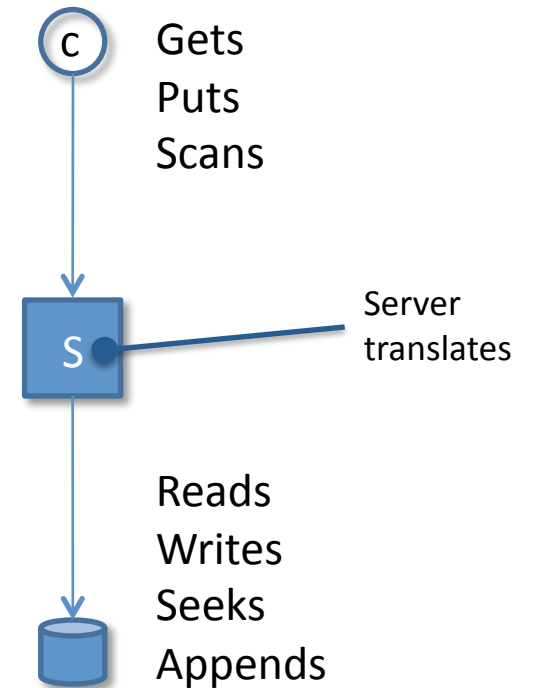
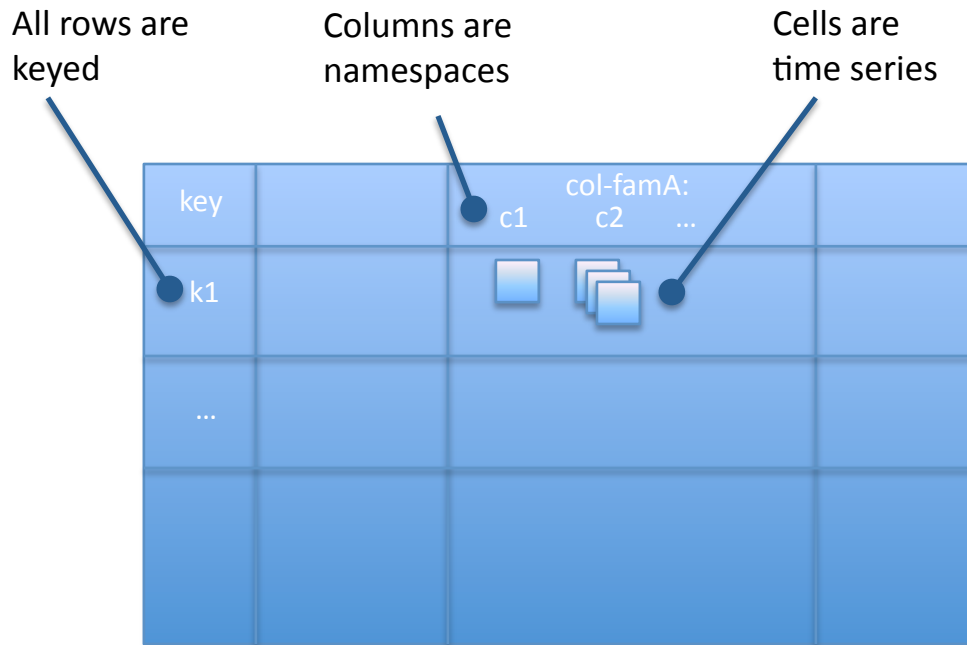
**(CONTRASTING DYNAMO'S AND BIGTABLE'S APPROACHES)**

NYU Distributed System Class, Invited Lecture, Dec/2009  
alberto.lerner@gmail.com

- Bigtable and Dynamo (Cassandra) represent a class of systems that sits in between Distributed File Systems and Database Management Systems
- They provide more functionality than a file system in that they export a more powerful API (data model) than reads, writes, and seeks.
- This data model is rich enough to build applications directly atop of it but has less querying capabilities than a full-fledged database with a query language

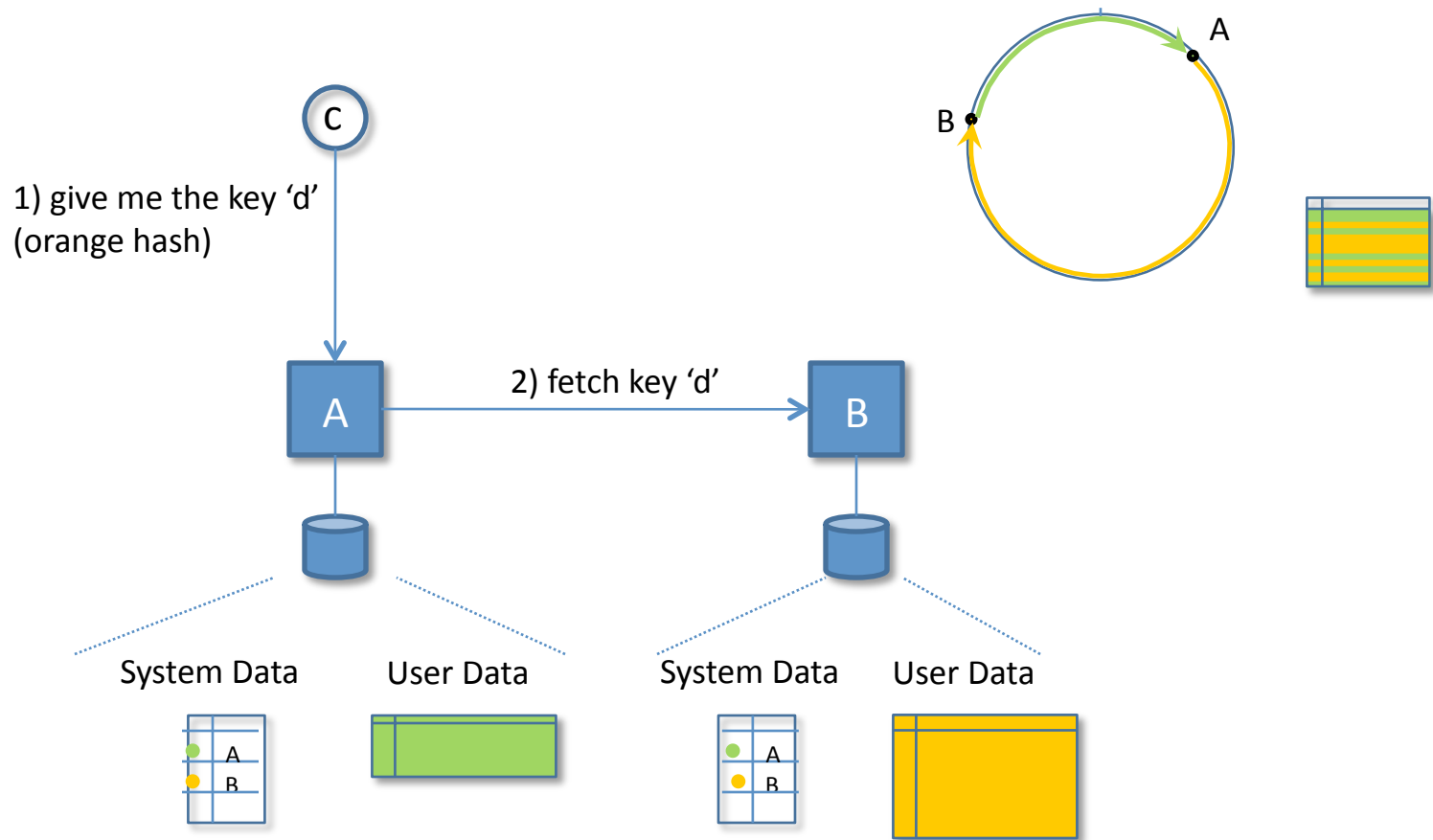
## **Key-value Stores**

They have been deployed in very large data management scenarios.



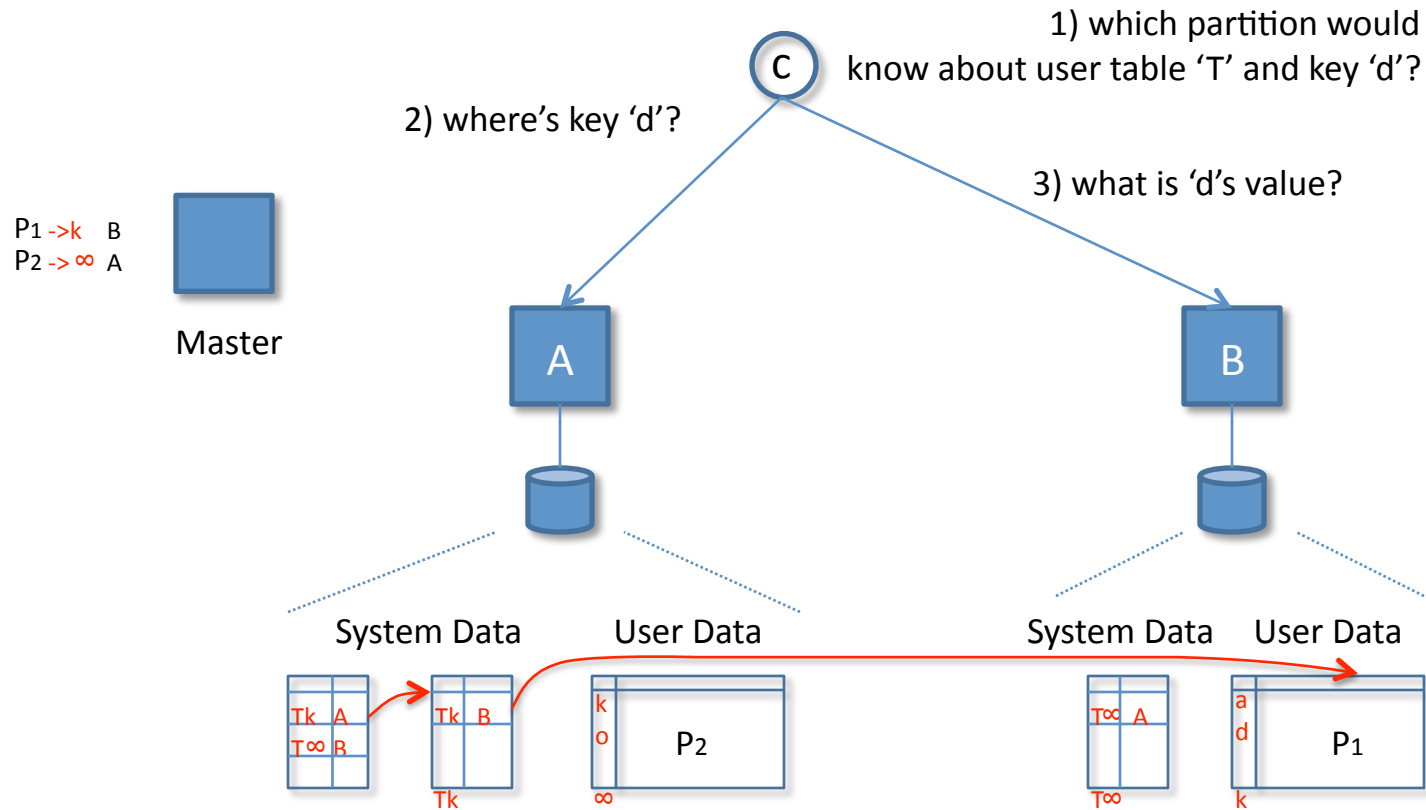
## 10.000 ft view of a server

A server transforms requests against an abstract key-value model into file reads and writes. A system is a group of servers cooperating to support what looks like a unified instance of the data model.



## Dynamo uses Consistent Hashing to distribute rows

Logic of key location can be pushed to clients. Or the client can ask a node to redirect requests.



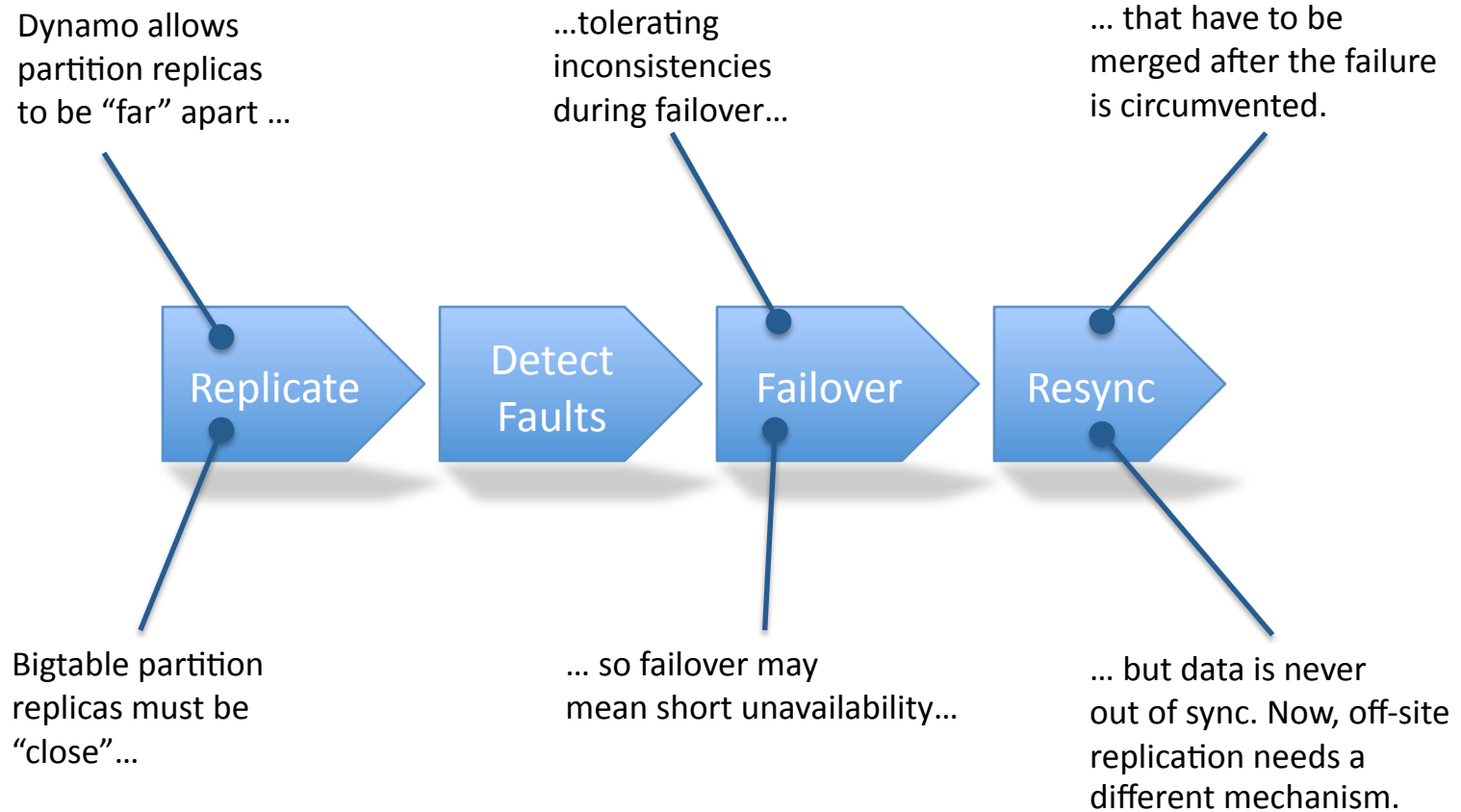
## Bigtable uses key ranges

Lookup consists of traversing the Metadata (partitions) table.

- Why **build** such systems?
  - In short, scalability and availability in the level these companies needed was not ready, off-the-shelf
  - There is also the strategic argument of not depending on third party software for core functionality
- How to go about the choice of **algorithms** in the design?
  - Start with what is important for users: e.g. an “always writable” system for Amazon
  - Leverage existing technology: e.g. use of sstables at Google
  - That will tie the relevant algorithms; in turn, the design space gets reduced

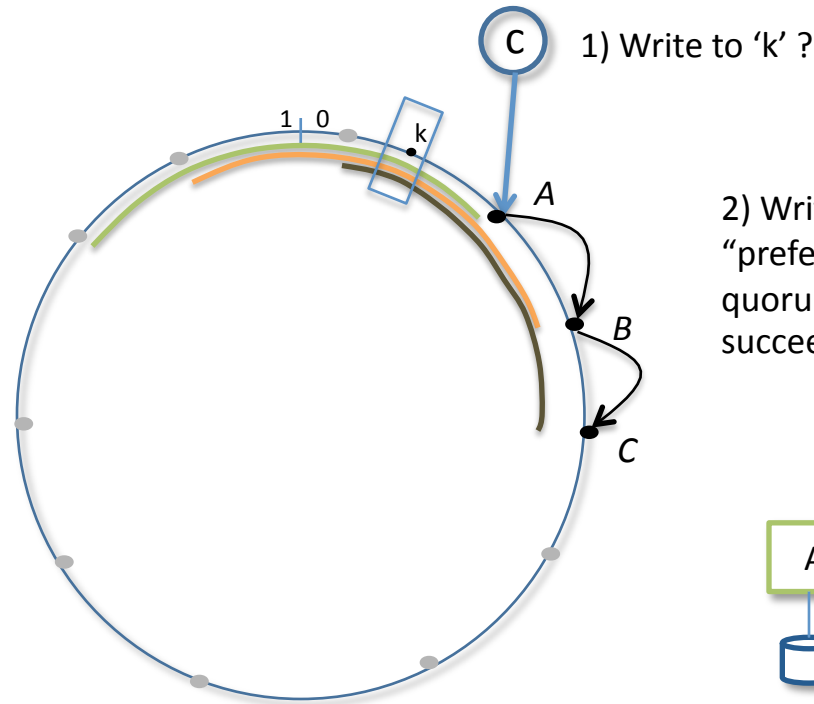
## Designing large systems

You’ve seen a lot of useful techniques during this course.

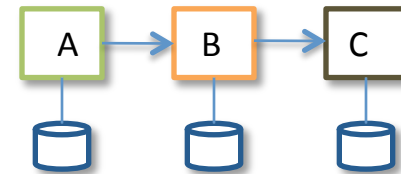


## Fault Tolerance Features

Dynamo and Bigtable made quite different choices. Among them, consistency and availability decisions may have been the most important ones.



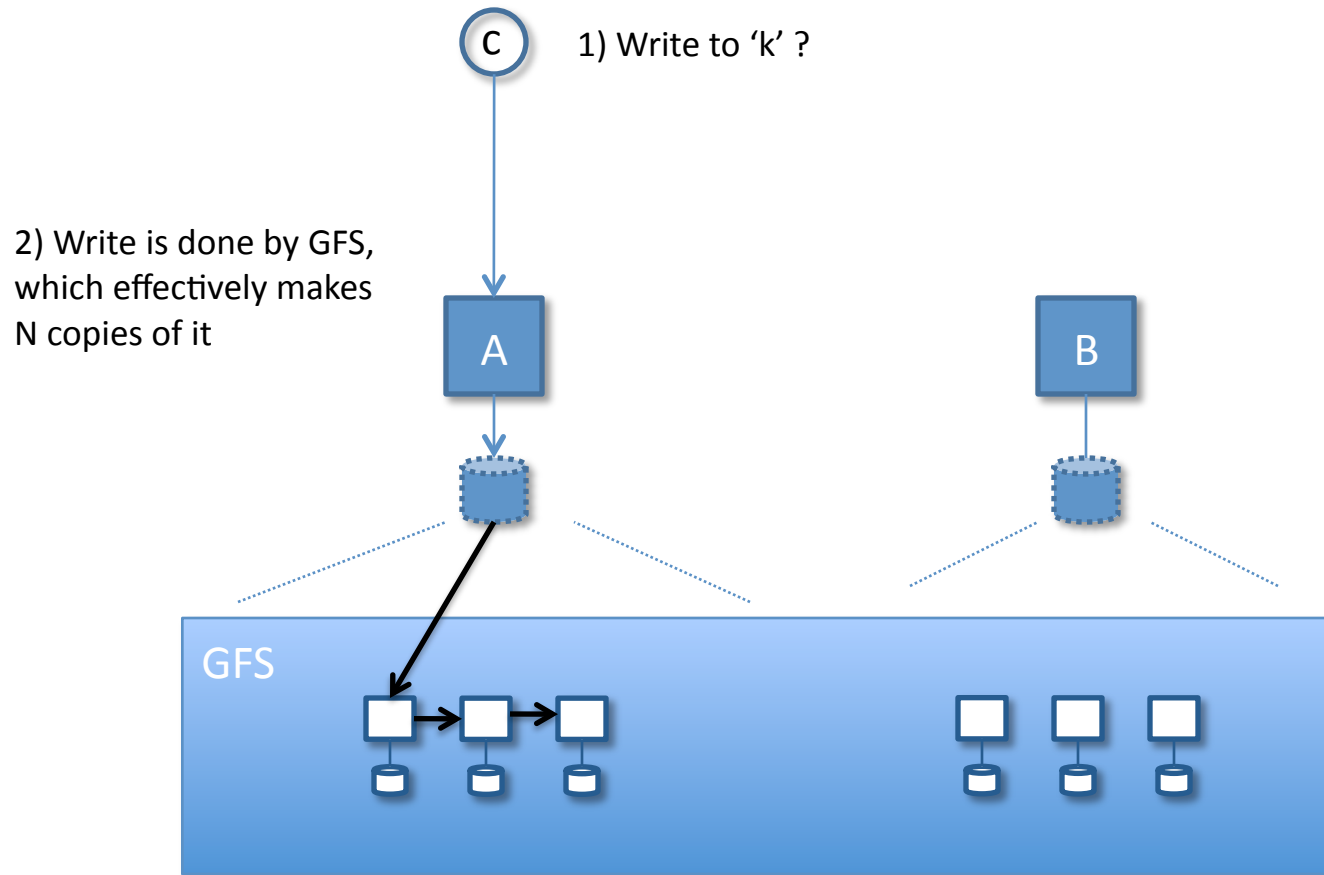
2) Write is forwarded to the "preference list" (sort of a write quorum). In this case, the two succeeding servers



## Starting with copies of data, Dynamo first

Effectively, every server is taking care of the  $W$  ranges before it. Since these ranges overlap, there is data redundancy.

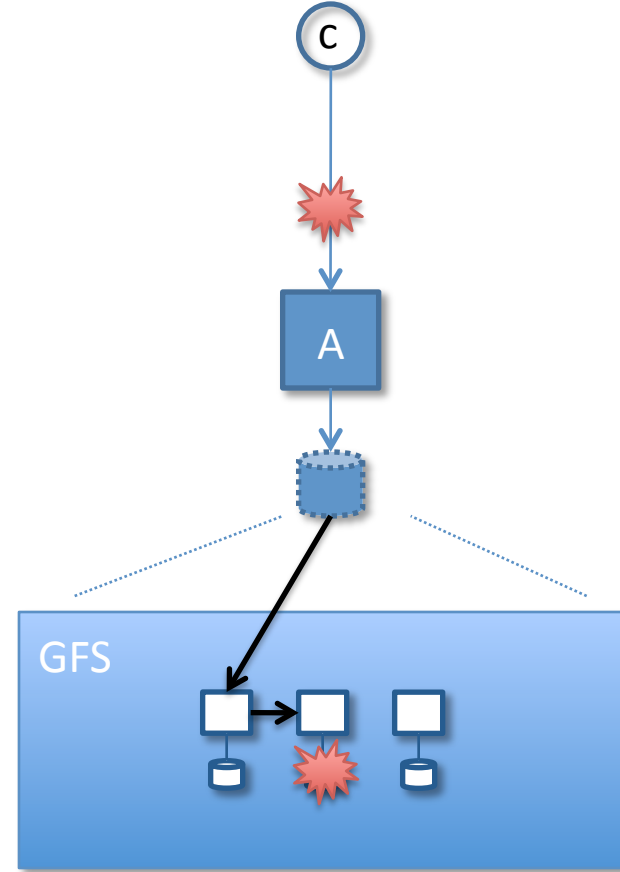
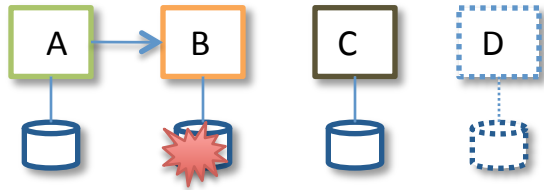
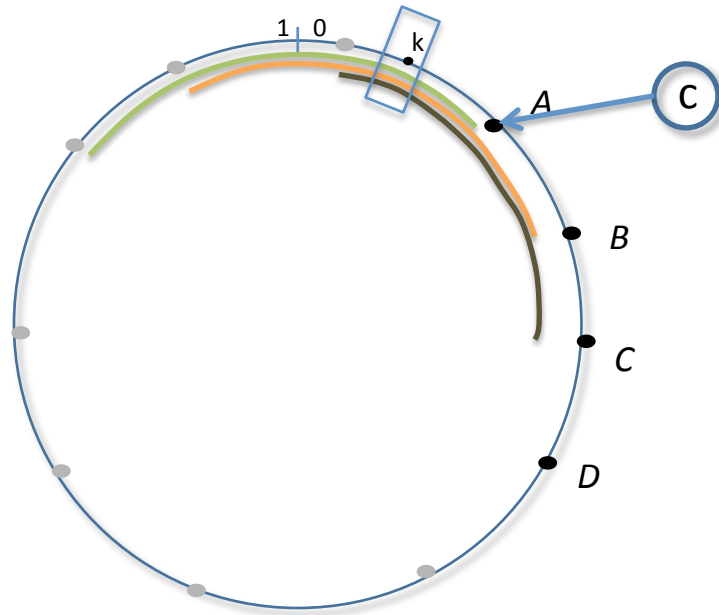
Replicate



## Bigtable delegates to GFS

The difference here is that a server stores data in a networked file system.

Replicate



## Data Consistency

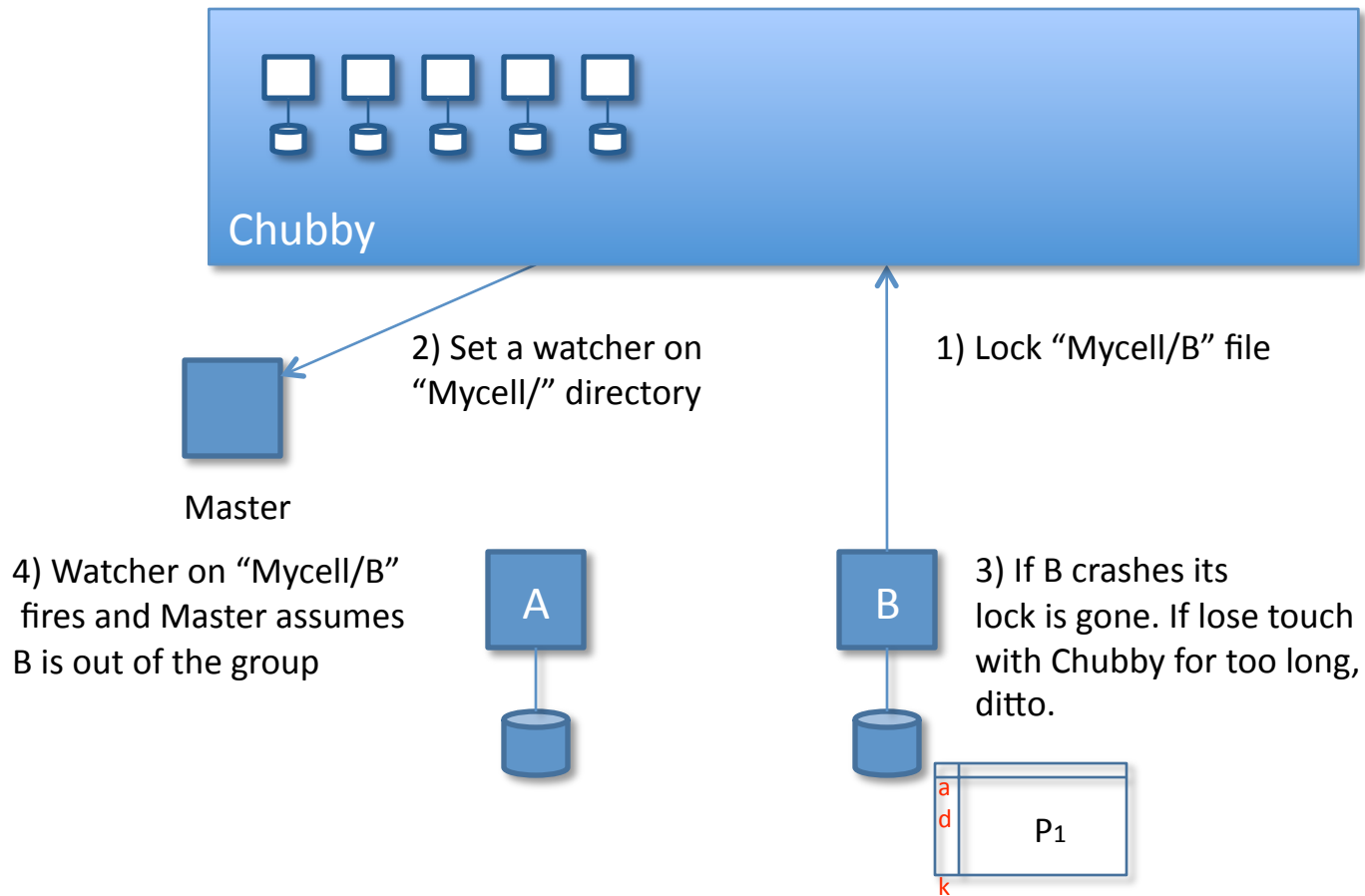
Write quorum can be relaxed in Dynamo. Normally, not so in GFS. So a Dynamo client accepts reading inconsistent copies (with vector clocks). A Bigtable client retry writes on failures (writes are made idempotent).

- Every  $t$  times, each server reaches out to a random peer
- During this exchange, the pair trade information and update their view of the servers in the system
- Assuming  $N$  servers, in  $O(\log N)$  each server would know about every other
- It would also learn about the most recent time that a server was successfully contacted
- The longer it takes for a server to be contacted, the largest the probability of it being down or unreachable
- A server is considered to be out of the ring roughly if enough peers haven't managed to contact it for a period of time

## **In Dynamo, how is a faulty server detected?**

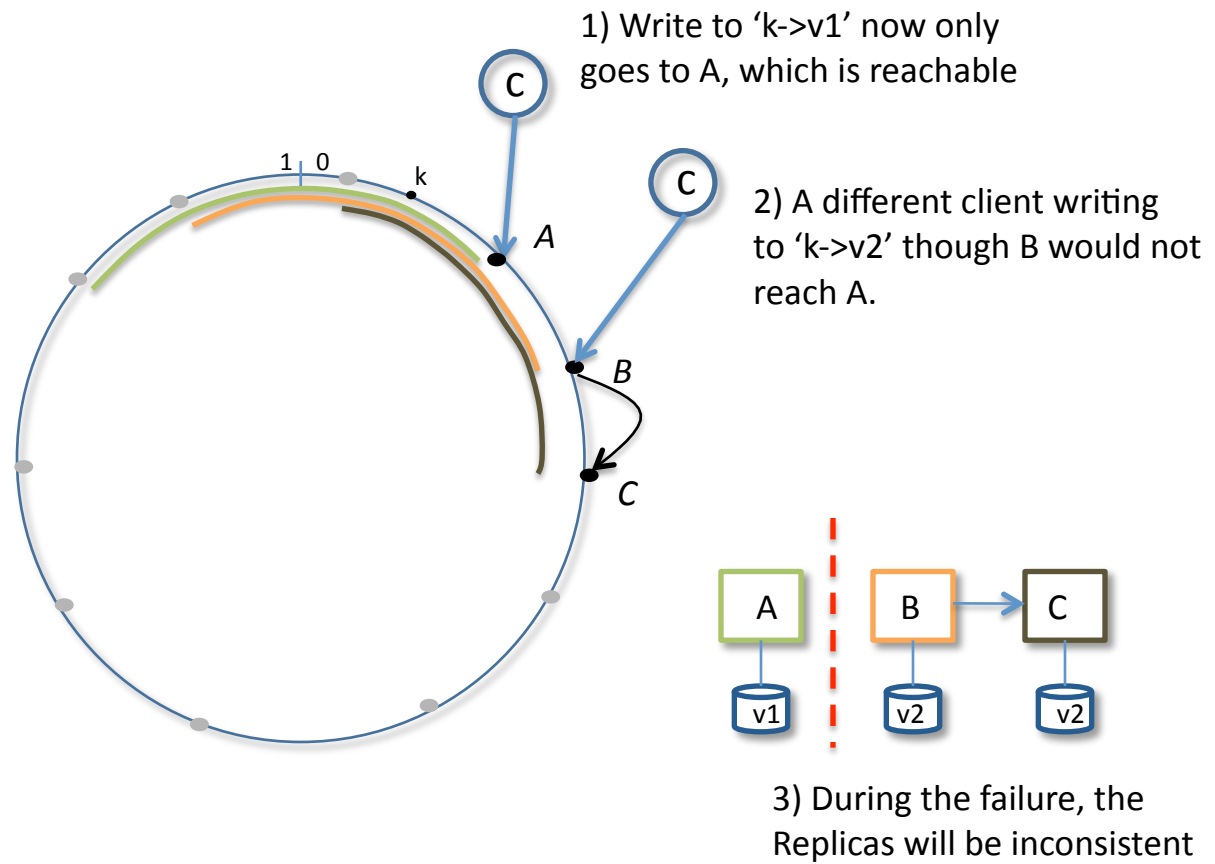
Gossip-based protocol

Fault Det.



## Group membership can be done through Paxos

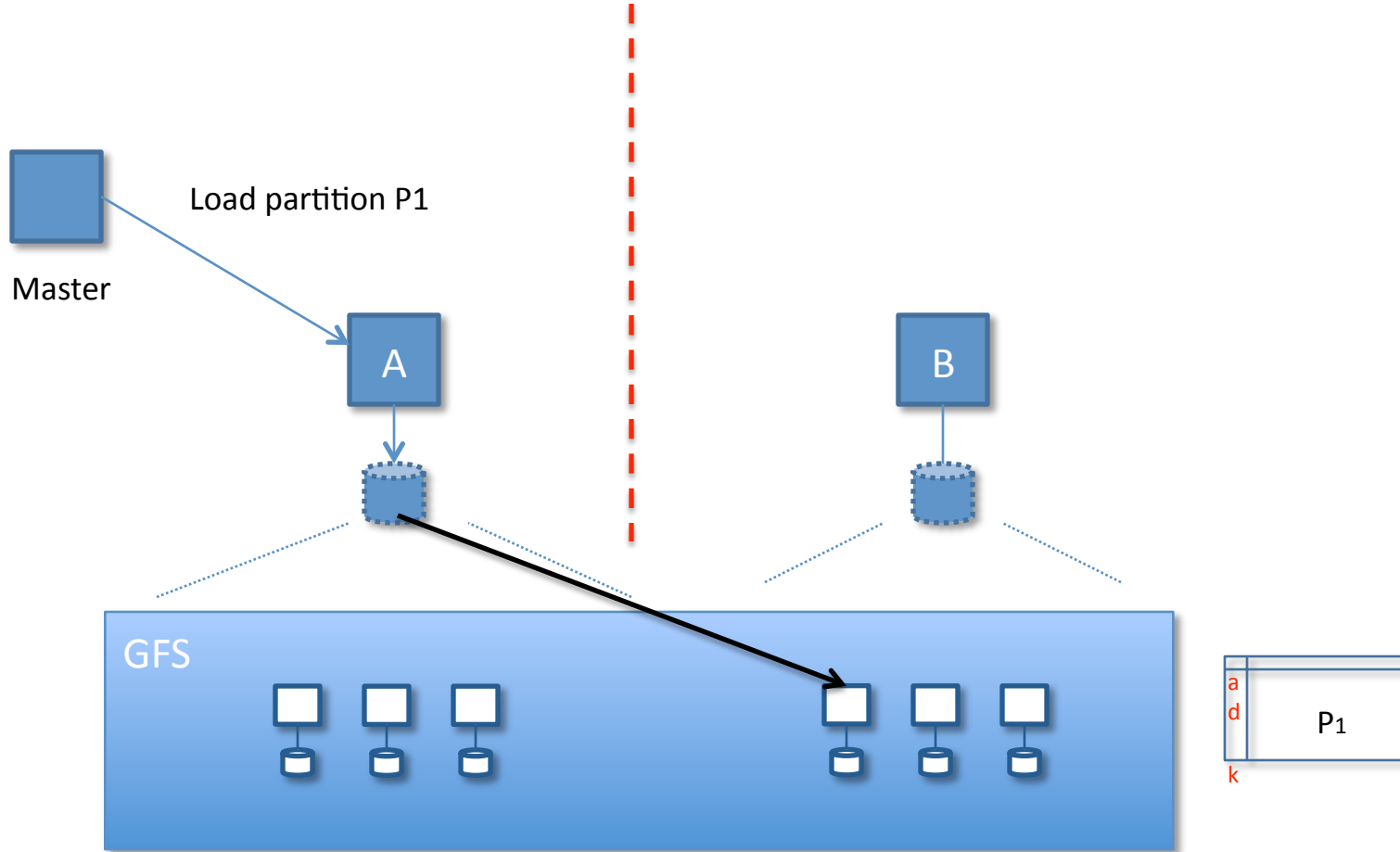
Chubby runs Paxos internally for its replicated state machine scheme. It exports abstractions such as locks and watchers over files.



## Dynamo allows writing through a failover

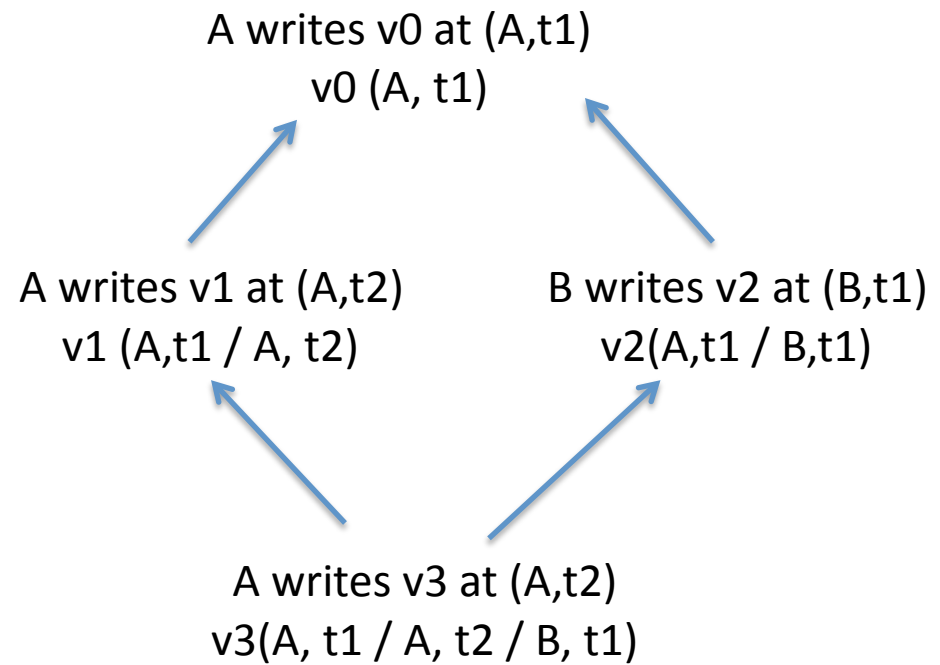
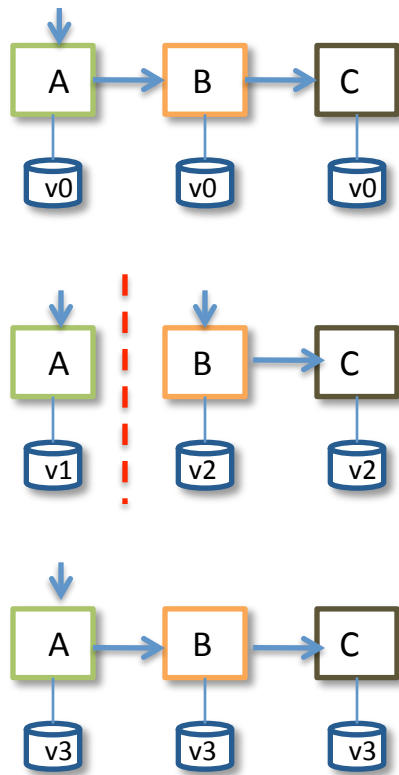
If one wishes to set a low write quorum, one can.

Failover



## Bigtable reassigns partitions to other servers

Is it really just reopening the files that make up P1?



Resync

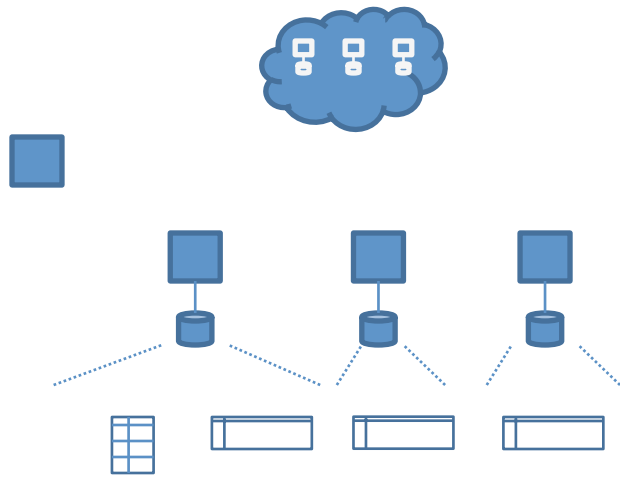
## Vector clocks help resync-ing after a failover

If clocks show conflicts, then it is up to the application to resolve them.  
 "Hinted-handoff" also as resync work.

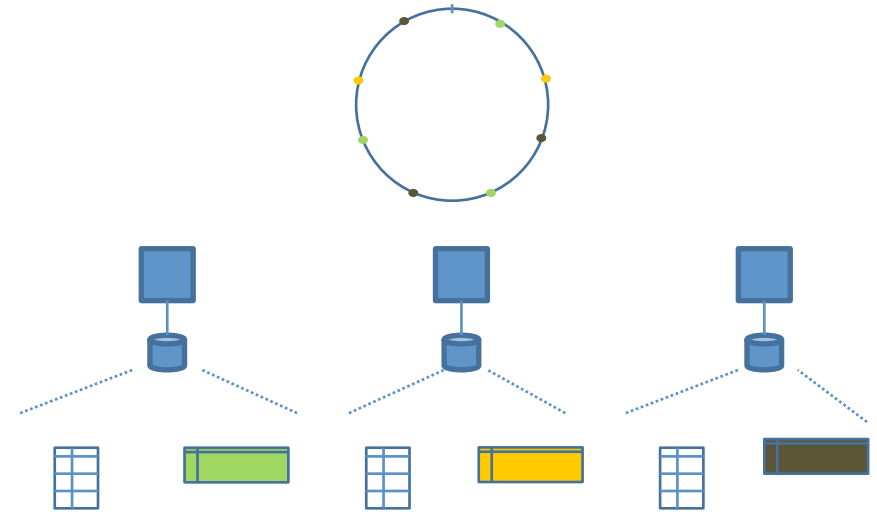
- A “mutation” gets written to a commit log first
- There is one log PER SERVER, that is, writes to several partitions are interleaved in this log
- To load a partition, the new server has to access the log records for *that* partition...
- Why? Some changes may not yet be reflected on data
- This has to be done on a failover situation, that is, as quickly as possible
  
- Solution: Bigtable is a distributed sorting system as well. Log partitions are sorted to speed up moving “tablets”

**No resync in Bigtable. But log sorting for quick failover.**

It could be seen as a “pre-sync” cost.



- Relies on GFS for data copies
- Centralized group membership and fault-detection
- Failover means having another server pick up the work
- No resync necessary (but sorts log to move partitions fast)



- Manages data copies itself
- Local fault-detection
- Failover means client will write to another server (if minimum quorum)
- Pays not to move data (resync with vector clocks)

## Comparing the approaches

## References

- Bigtable: A Distributed Storage System for Structured Data, Chang et al., OSDI'06
- The Chubby Lock Service for Loosely Coupled Distributed Systems, Mike Burrows, OSDI'06
- Paxos Made Simple, Leslie Lamport
  
- Dynamo: Amazon's Highly Available Key Value Store, DeCandia et al., SOSP'07
- Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the WWW, STOC'97
- Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, Stoica et al, SIGCOMM'01