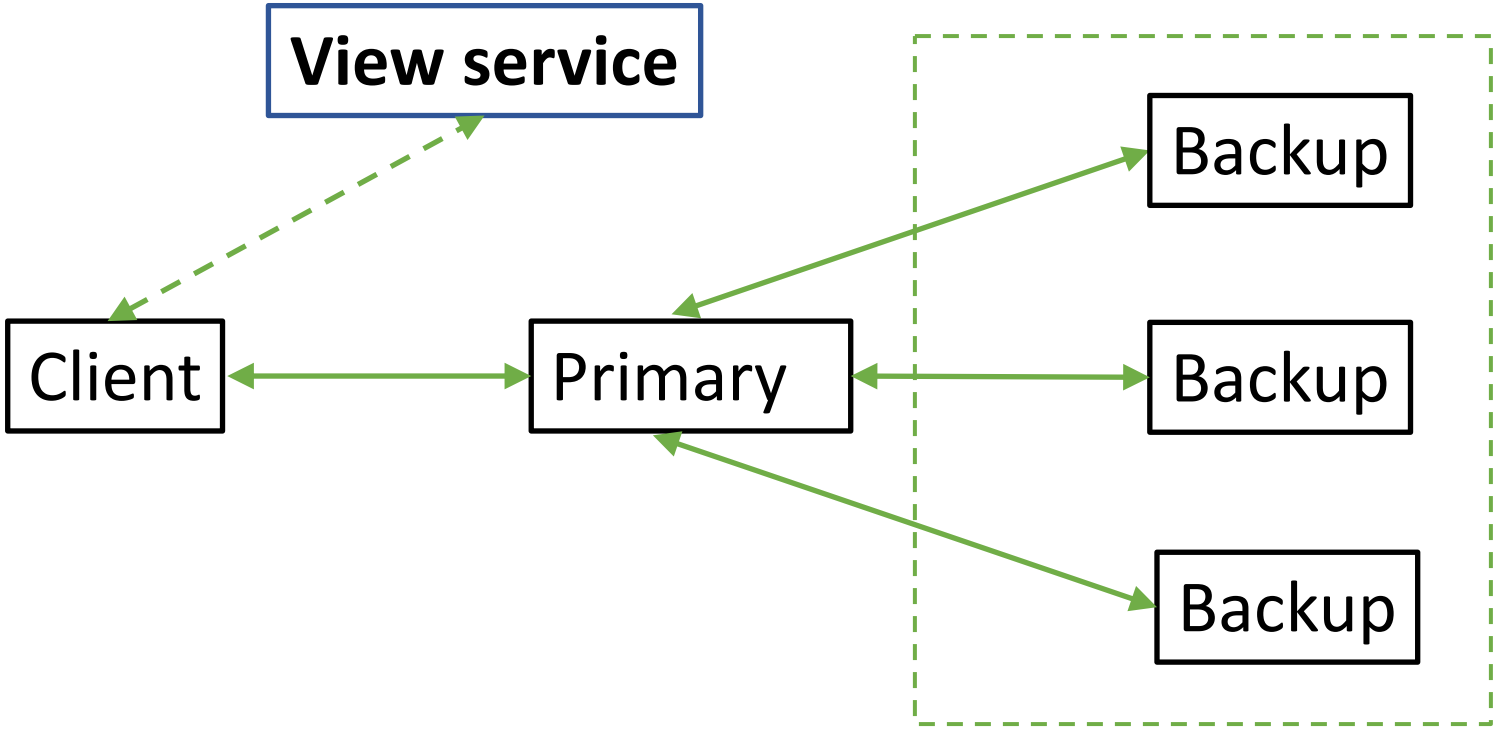
Week 4
Spyros Mastorakis

Outline

- Recap: primary-backup replication
- Guarantees of primary-backup replication (linearizability)

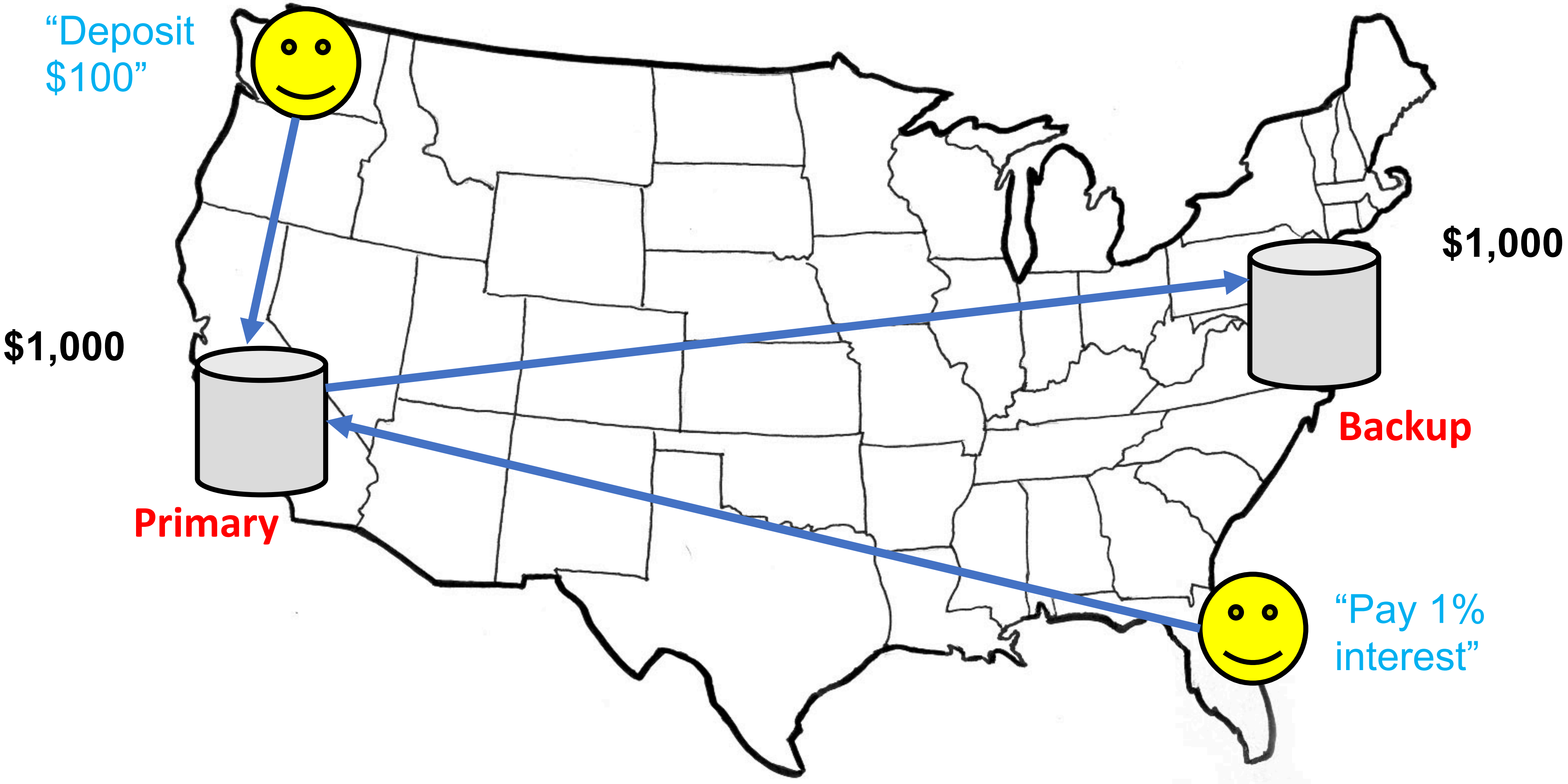
Primary Backup Replication



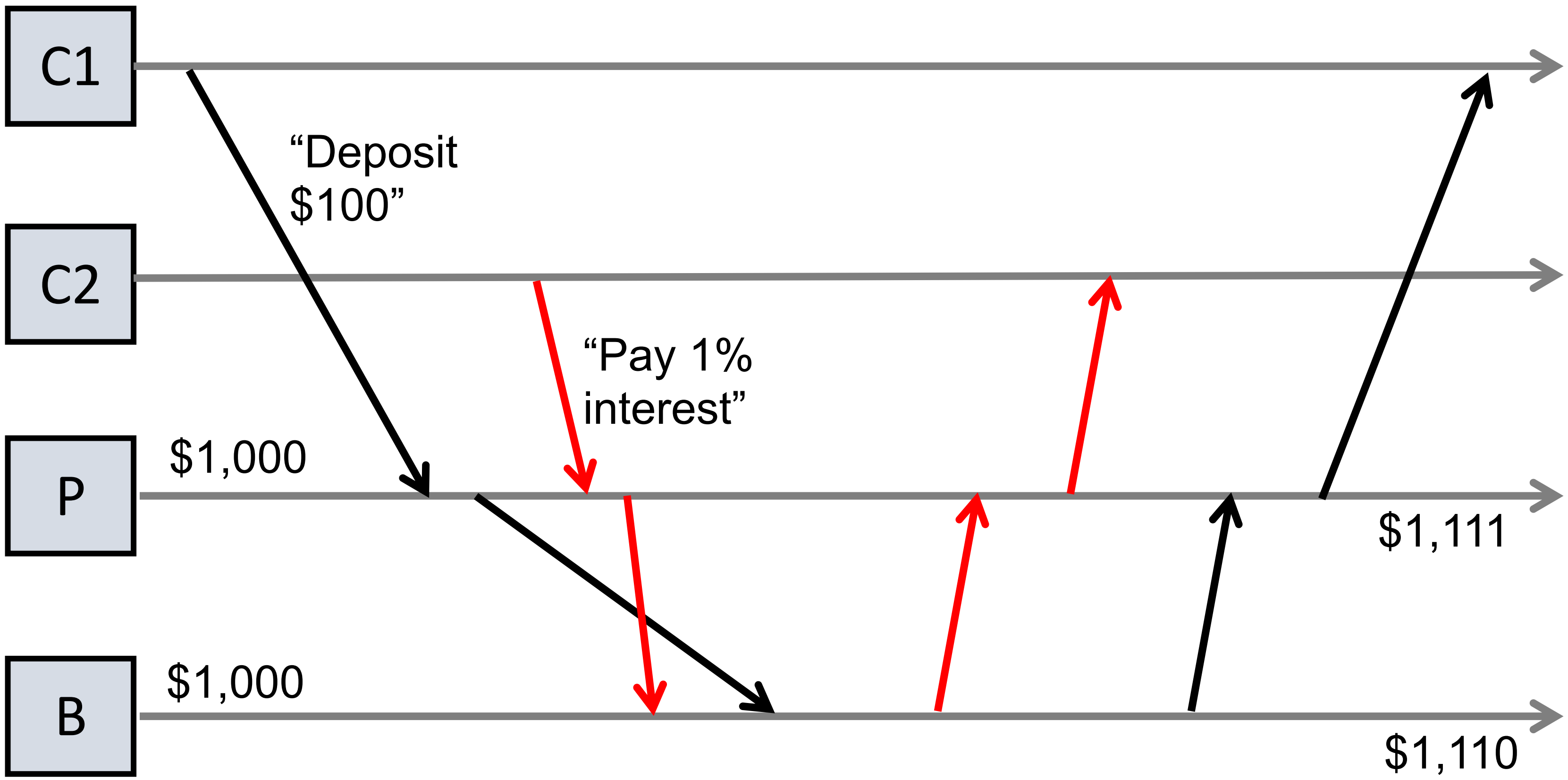
View service

- Task: monitor primary and backups to detect when to change view
  - Change only after primary has acknowledged current view
  - Primary sends acknowledgement only after syncing with all backups
- Scaling the view service
  - Clients cache view (do not send request to service each time)
  - Challenge: split brain scenario
    - Primary must check with backups before serving client

Example: bank database



Primary Backup Sync



How to order updates?

- RSM rule: all updates must be applied in same order at all replicas
- Primary is the serializer of all the updates
    - Primary decides how to handle concurrent updates
    - Updates are faithfully reproduced by backups

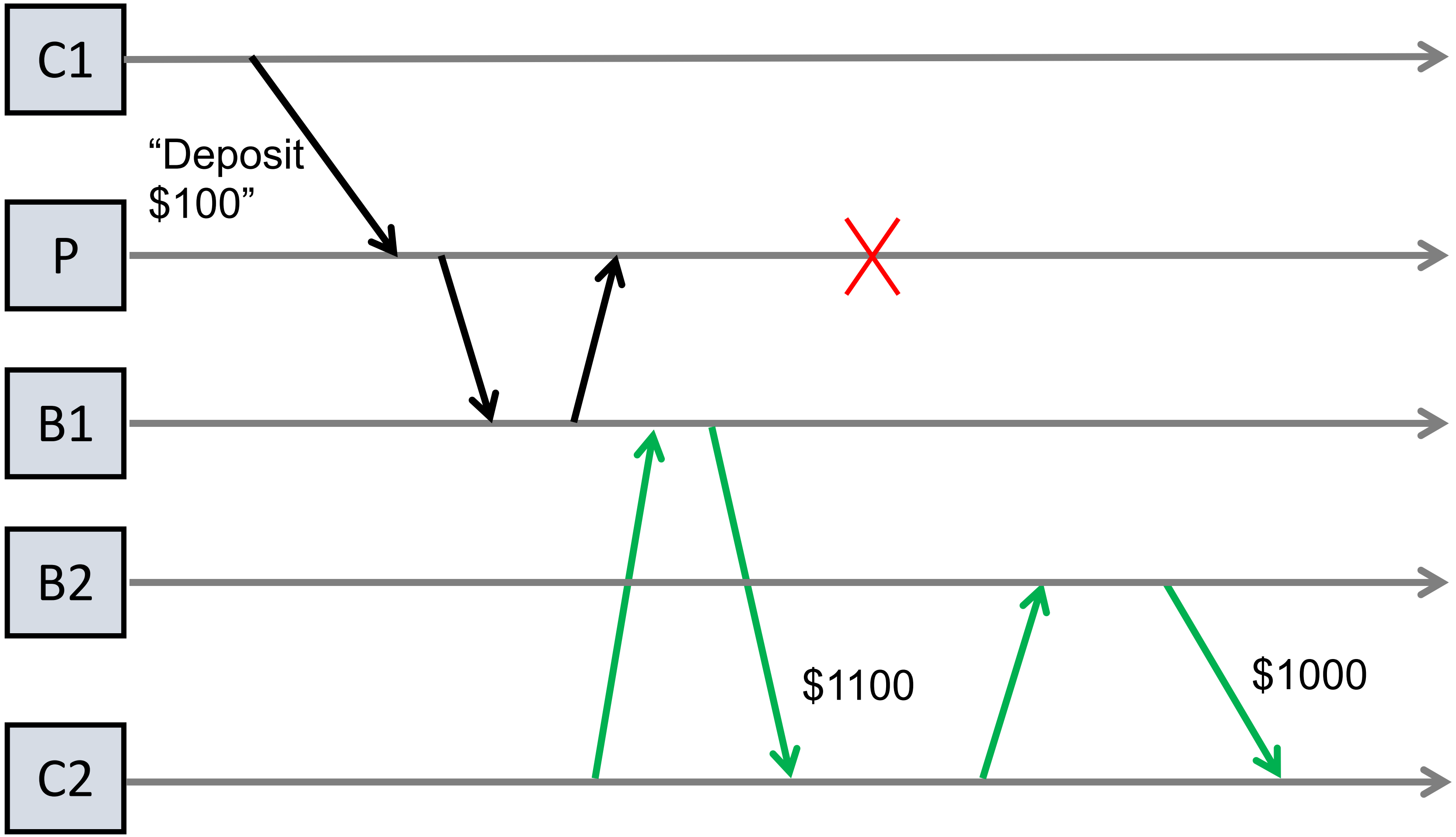
Handling Reads

- Can backups serve reads?
  - Reduces load on primary

- What if primary’s state is ahead of backup?
  - Updates to primary are not yet externally visible
  - Effect of read equivalent to if primary fails at this point

- What if backup’s state is ahead of primary?
  - Different backups may not be in sync
  - Primary may get replaced before backups apply update

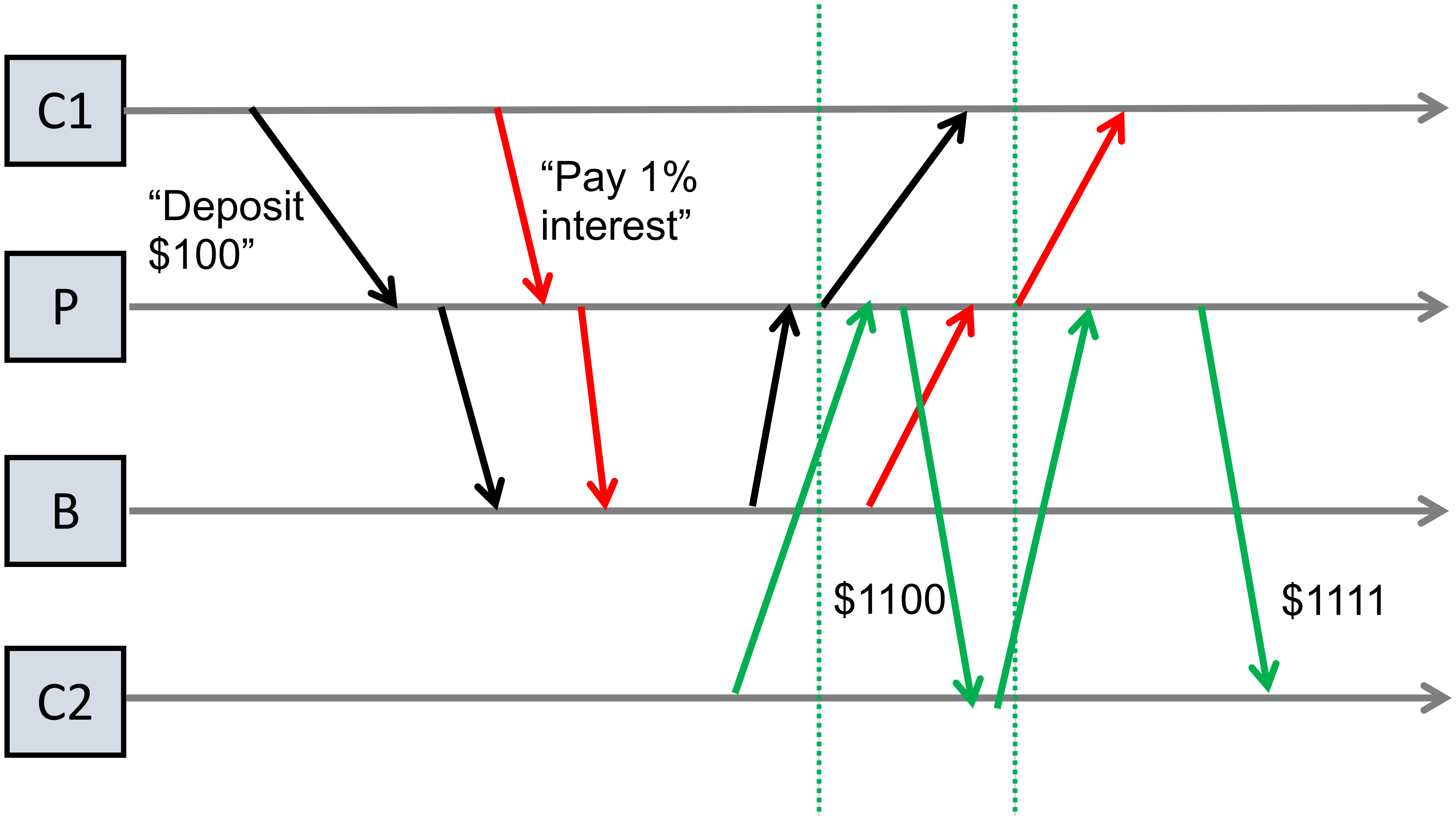
Reads: Primary vs. Backup



Desired Properties

- All writes are totally ordered
- Once a read returns a particular value, all later reads should return that value or the value of a later write
- Once a write completes, all later reads should return the value of that write or the value of a later write

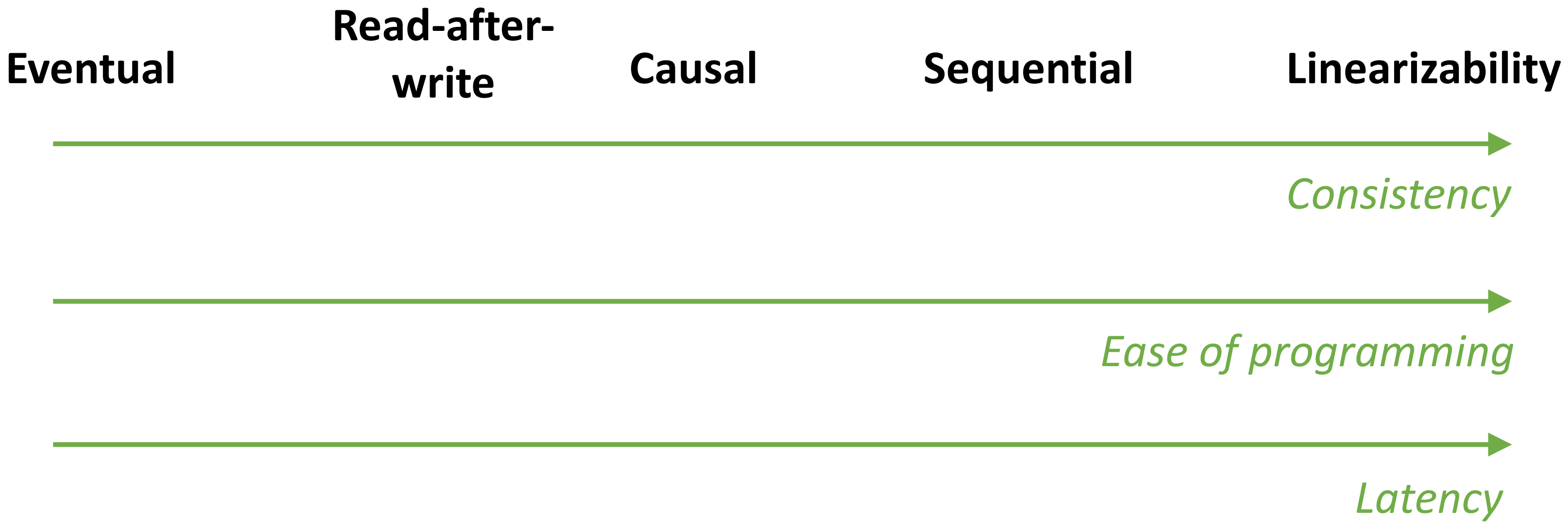
Reads relative to Writes



Linearizability

- Property: Once a write is complete, a read will see the new value
- Effects of writes and reads need to be externally visible by all the clients

Consistency Spectrum



Eventual

Read-after-write

Causal

Sequential

Linearizability

Consistency

Ease of programming

Latency

Tradeoff: Consistency vs performance

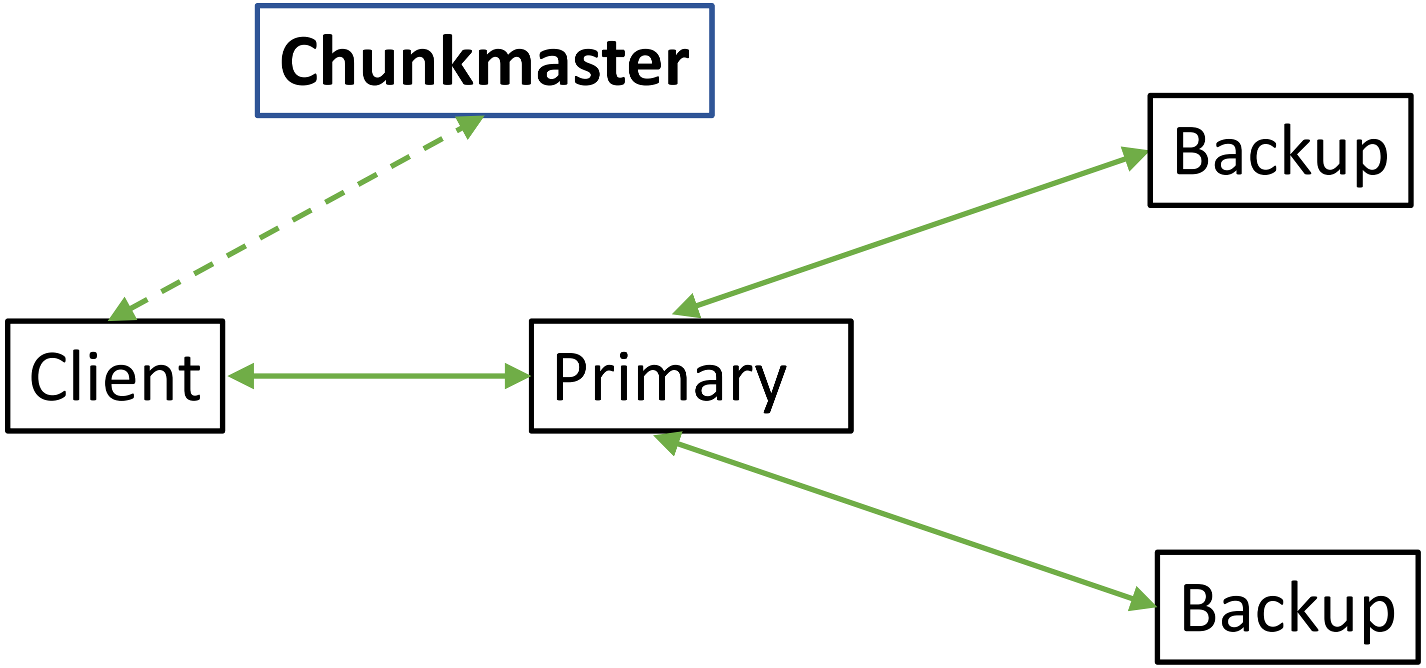
Case study: GFS

- Google File System
  - Distributed storage system tailored to Google’s workloads
- Workload characteristics:
  - Multi-GB files (each consists of multiple chunks)
  - Failures are extremely common

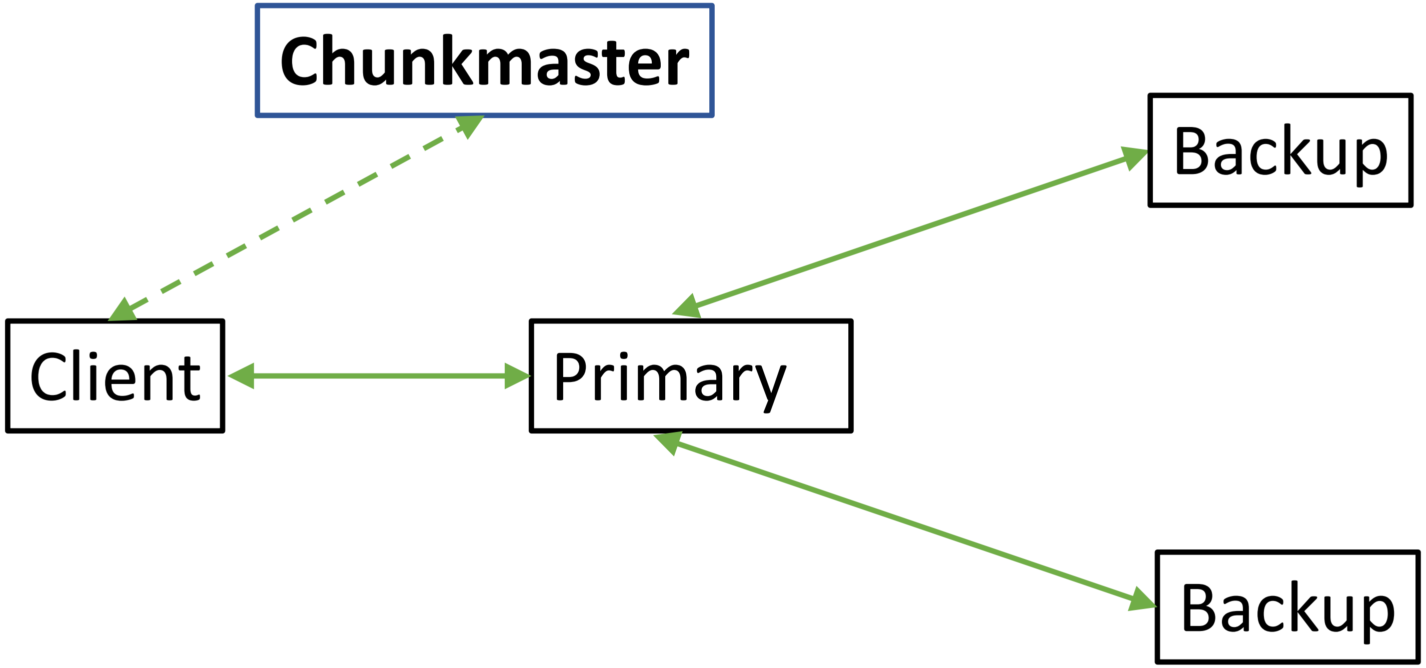
GFS: high-level design

- Files are split into 64 MB chunks
- Every chunk is replicated on three randomly selected machines
- A central chunkmaster server picks (and knows) where every replica of every chunk is stored

GFS: replication

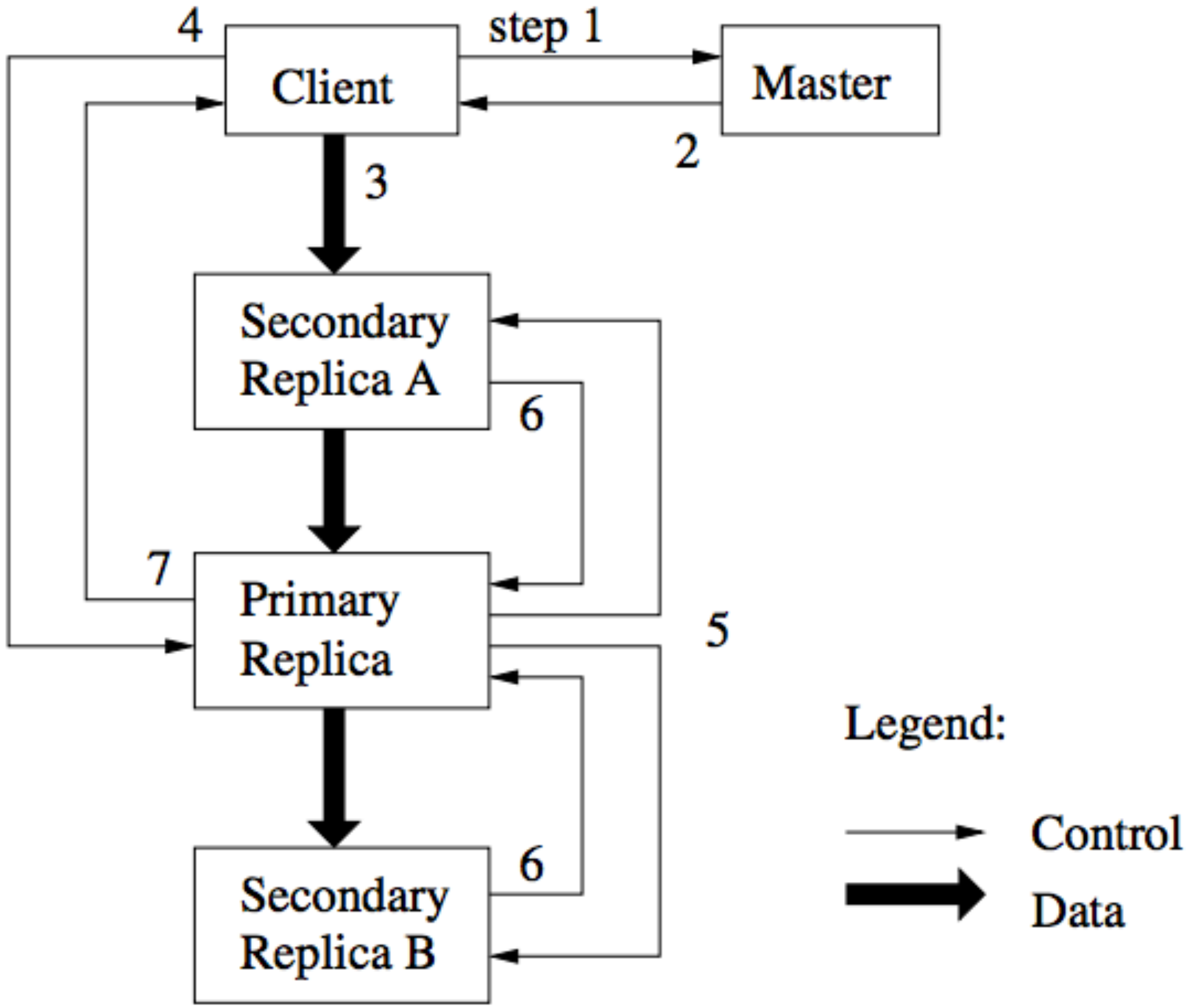


GFS: replication



- Challenge due to large writes: high latency when writing to distant primary
- How to optimize write performance?

GFS: data flow vs. control flow



4
Client
step 1
Master
2
3
Secondary Replica A
6
7
Primary Replica
5
Secondary Replica B
6
Legend:
Control
Data