Large Scale Systems
CS 410 / 510

Lecture 7:

Deterministic Databases

Suyash Gupta

Assistant Professor

Distopia Labs and ONRG

Dept. of Computer Science

(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

UNIVERSITY OF OREGON

Assignment 2 is Out!

- **Assignment 2 is out!**
- Please work with your groups to understand the underlying system.
- Assignment 2 report **deadline** → April 30, 2026 at 11:59pm.

Last Class

- Last class we looked at:

- Replication
- Distributed Consistency Levels
- Isolation Levels

How do Modern systems look like?

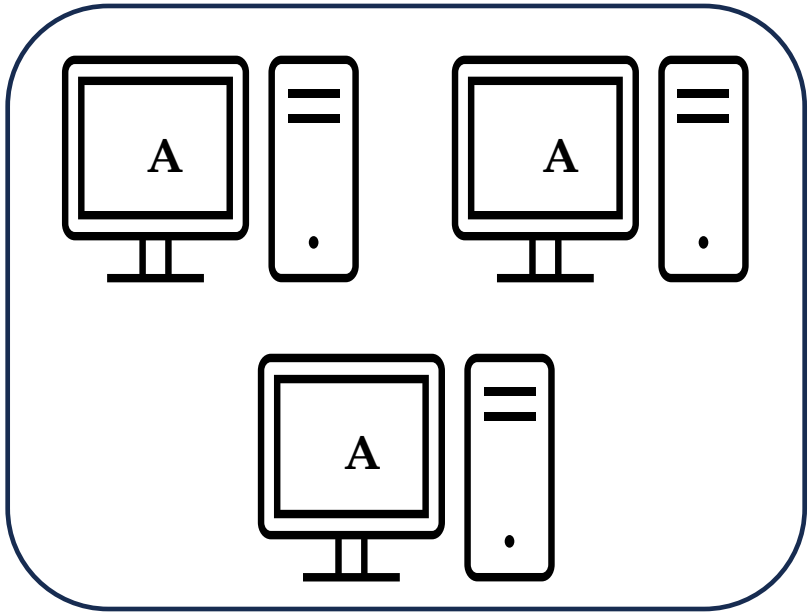
- Are modern distributed systems fully replicated or sharded?

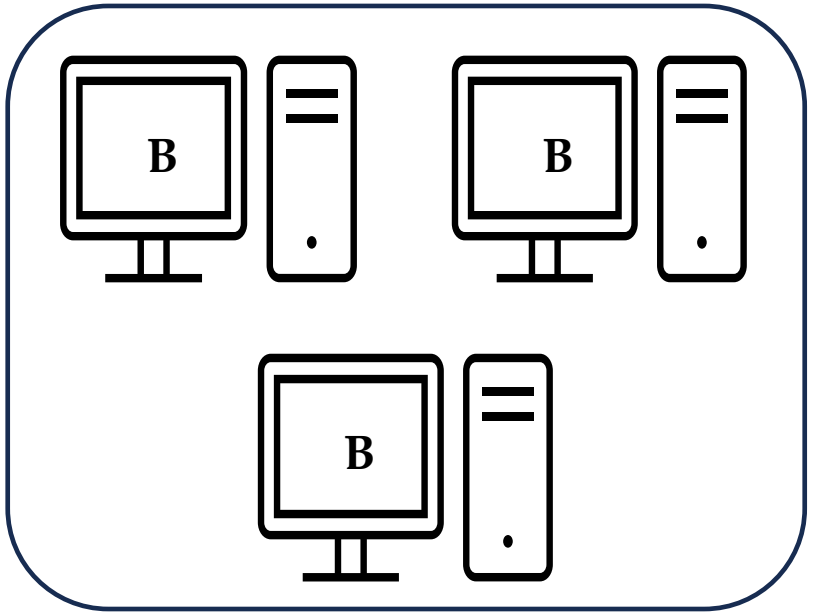
How do Modern systems look like?

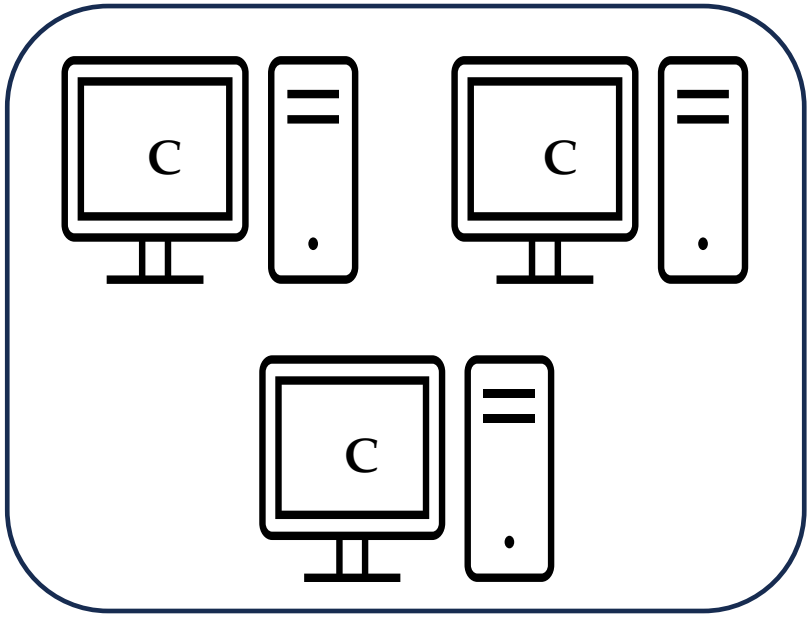
- Are modern distributed systems fully replicated or sharded?

- A combination of both:
  - Sharded Replicated
  - Replicated Sharded

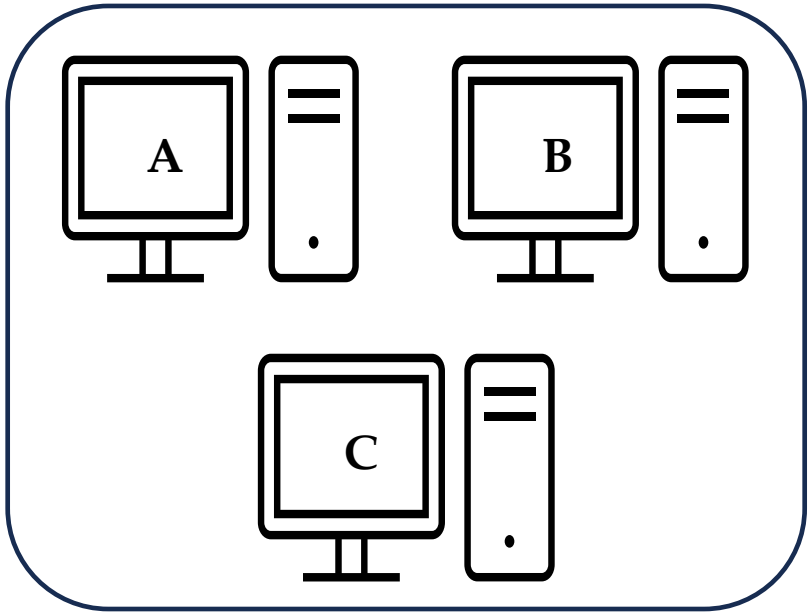
Sharded Replicated Systems

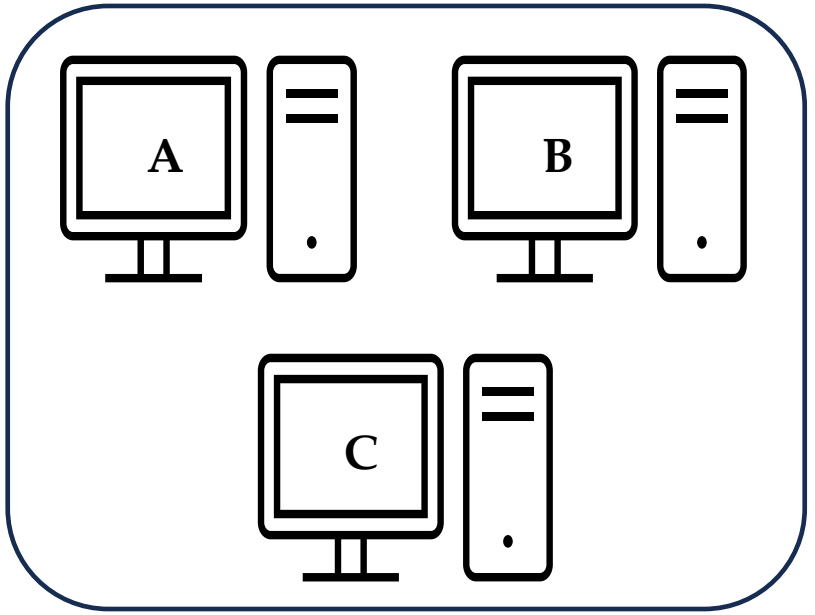


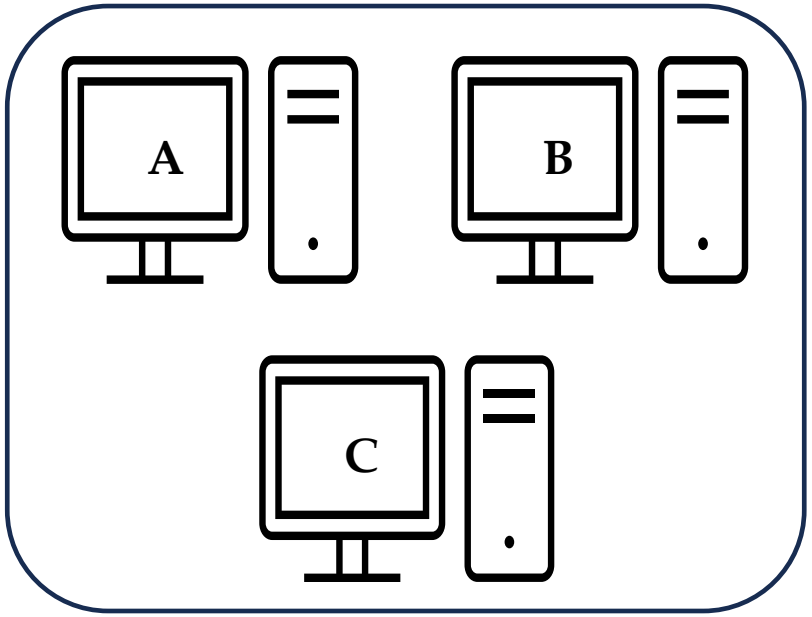




Replicated Sharded Systems







Disadvantage of Commit Protocols

- Couple of classes ago, we also looked at commit protocols.
- These protocols help to reach a decision to commit/abort a distributed transaction.
- Are there any **disadvantages** of using a commit protocol?

Disadvantage of Commit Protocols

- Couple of classes ago, we also looked at commit protocols.
- These protocols help to reach a decision to commit/abort a distributed transaction.
- Are there any **disadvantages** of using a commit protocol?

    1) 2PC is blocking.

Disadvantage of Commit Protocols

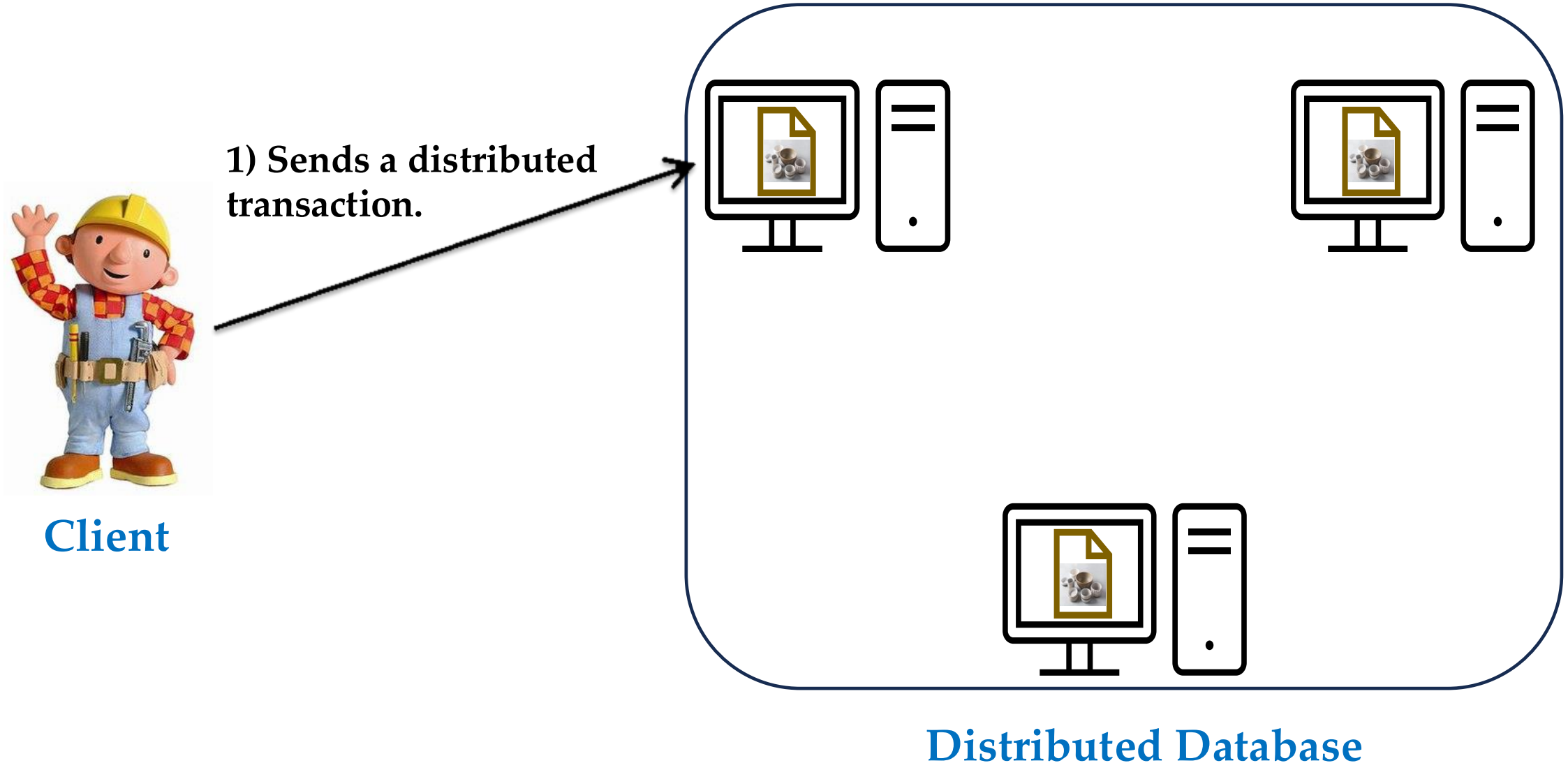
- Couple of classes ago, we also looked at commit protocols.
- These protocols help to reach a decision to commit/abort a distributed transaction.
- Are there any **disadvantages** of using a commit protocol?
    1) 2PC is blocking.
    2) 3PC is non-blocking but requires three network round trips.

Disadvantage of Commit Protocols

- Couple of classes ago, we also looked at commit protocols.
- These protocols help to reach a decision to commit/abort a distributed transaction.
- Are there any **disadvantages** of using a commit protocol?

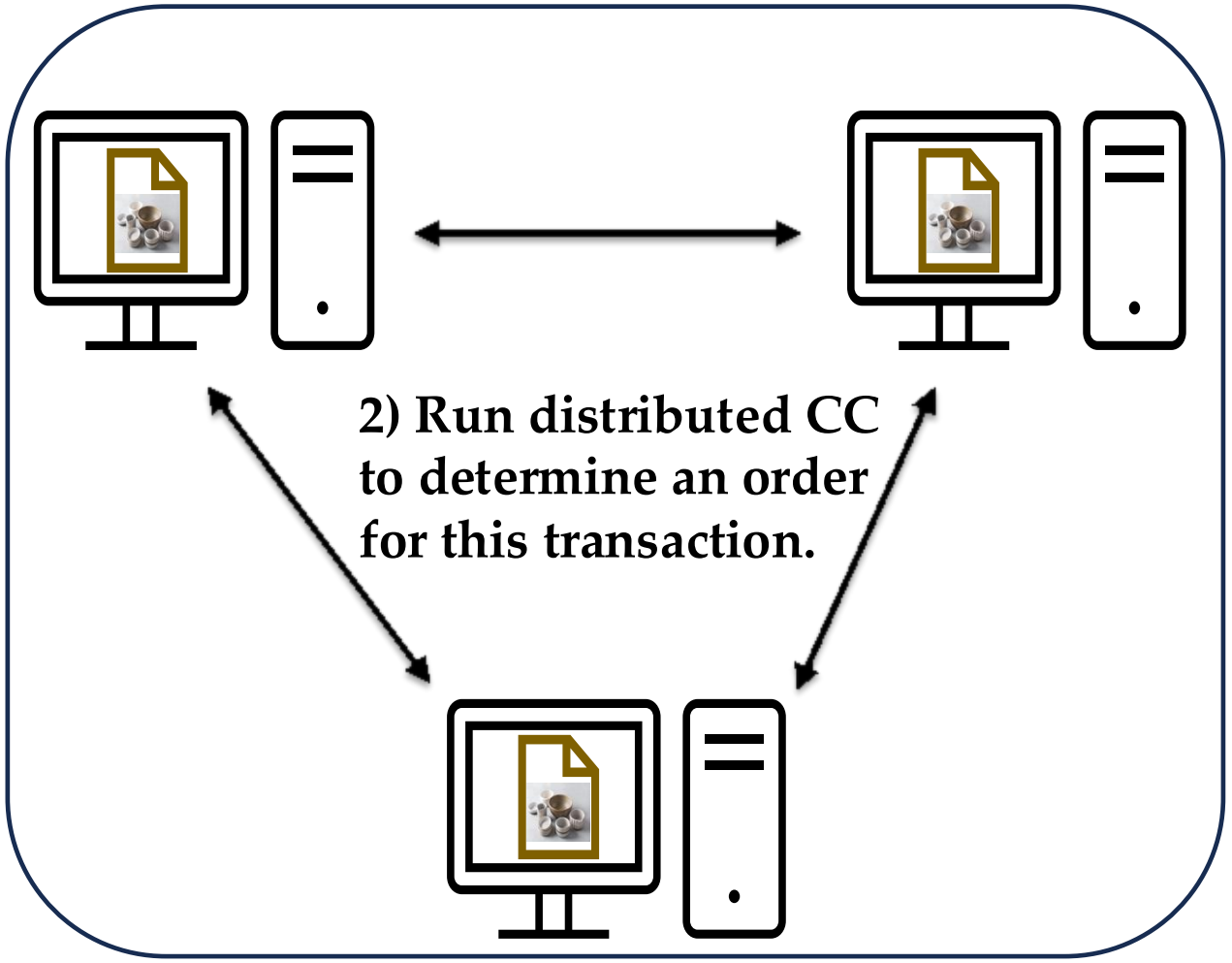
1) 2PC is blocking.
2) 3PC is non-blocking but requires three network round trips.
3) Running a commit protocol by itself is expensive! Why?

Cost of running Commit Protocols



Cost of running Commit Protocols

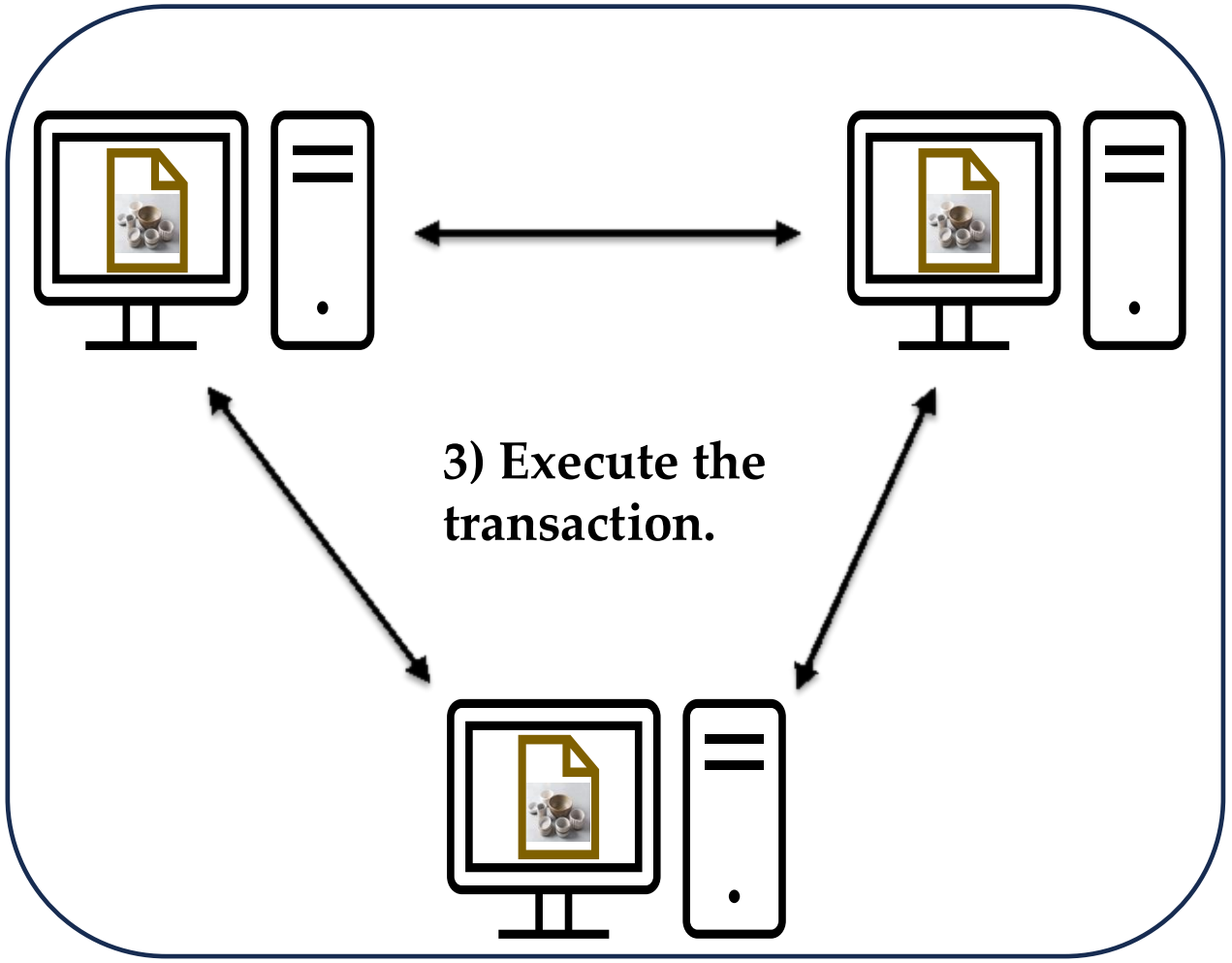
Client



Distributed Database

Cost of running Commit Protocols

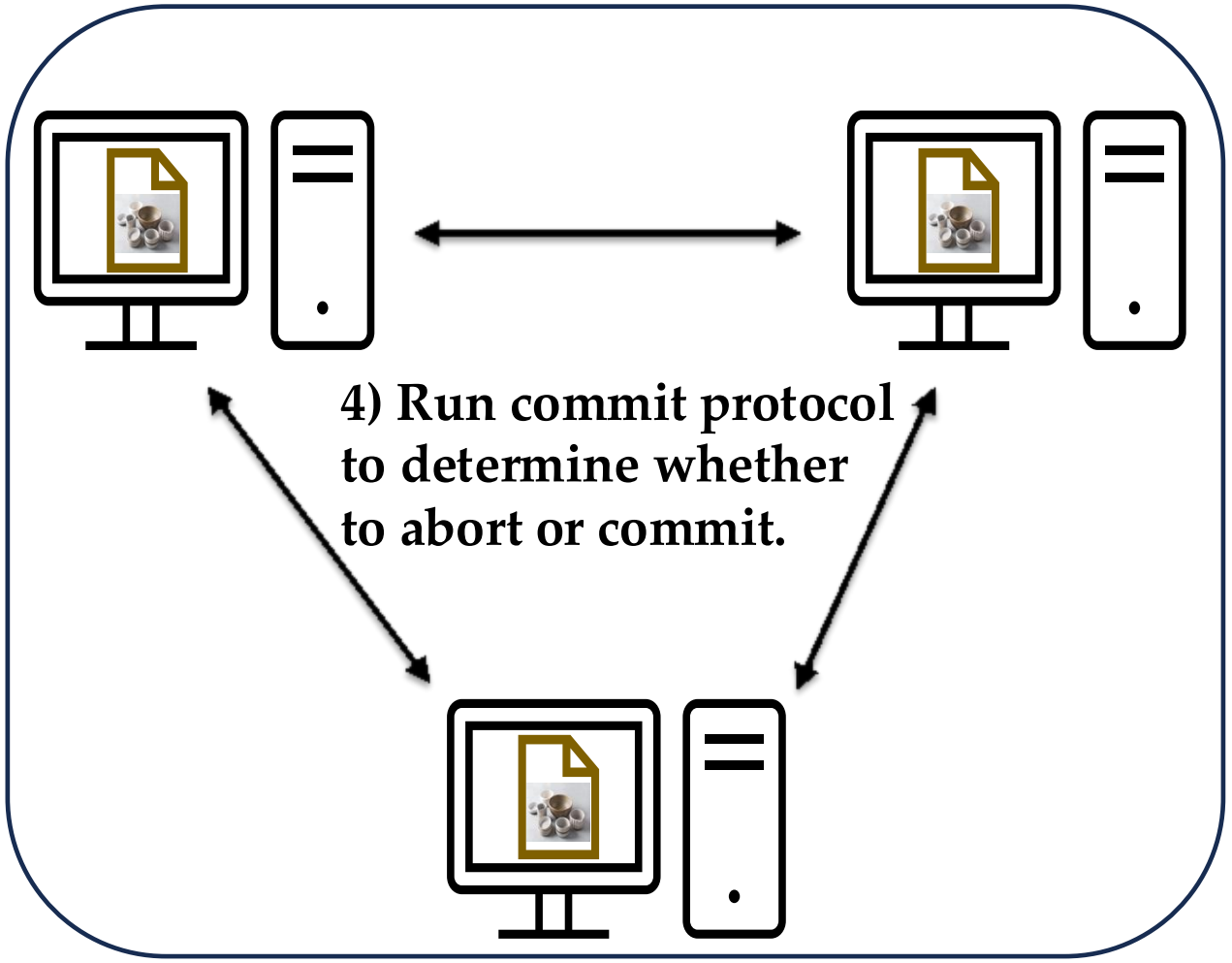
Client



Distributed Database

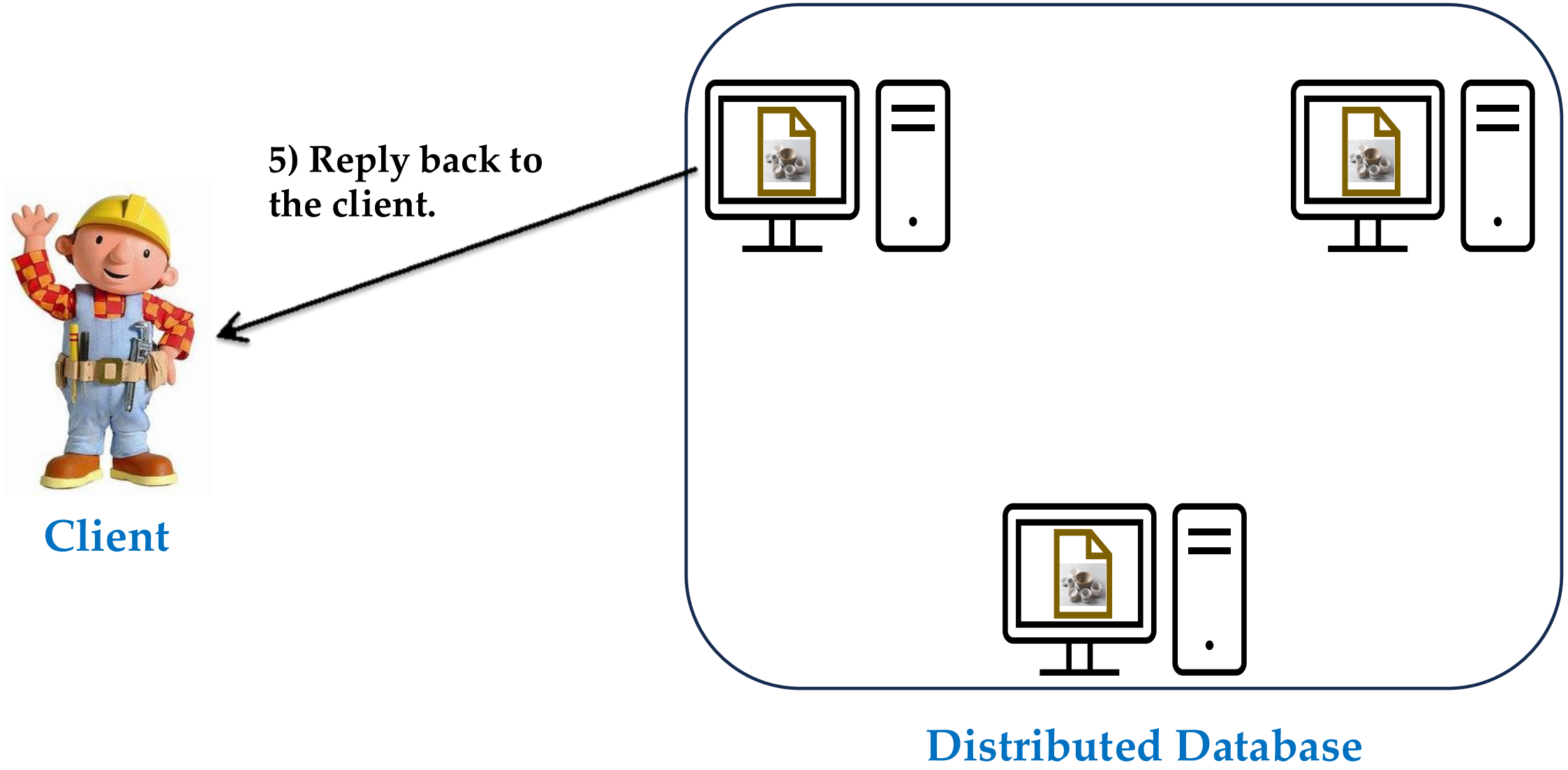
Cost of running Commit Protocols

Client



Distributed Database

Cost of running Commit Protocols



Disadvantage of Commit Protocols

- Concurrency Control is expensive → requires communication among partitions.
- Imagine, your system uses **2PL** for concurrency control, which requires **holding locks until commit**.

Disadvantage of Commit Protocols

- Concurrency Control is expensive → requires communication among partitions.
- Imagine, your system uses **2PL** for concurrency control, which requires **holding locks until commit**.
- In the case of a distributed transaction, each involved partition needs to hold locks until the **end of 2PC** protocol.

Disadvantage of Commit Protocols

- Concurrency Control is expensive → requires communication among partitions.
- Imagine, your system uses **2PL** for concurrency control, which requires **holding locks until commit**.
- In the case of a distributed transaction, each involved partition needs to hold locks until the **end of 2PC** protocol.
- If some records are **popular** → highly accessed by a lot of distributed transactions.
  - Holding locks will prevent transactional concurrency.
  - Resulting in a **drop in throughput**!
  - Moreover, **high latency** due to communication among partitions.

Disadvantage of Commit Protocols

- Can we do anything more?

Deterministic Databases

- Determine the **plan** to execute the transactions ahead of time.

- Every partition executes the distributed transaction in accordance with the plan.

Deterministic Databases

- Determine the **plan** to execute the transactions ahead of time.
- Every partition executes the distributed transaction in accordance with the plan.
- This allows **skipping** the use of 2PC protocol!

Intuition behind Deterministic Databases

- Deterministic databases aim to eliminate commit protocols as they are expensive.
- But **why** are commit protocols needed in the first place?

Intuition behind Deterministic Databases

- Deterministic databases aim to eliminate commit protocols as they are expensive.
- But **why** are commit protocols needed in the first place?
  - Two partitions may decide differently on the fate on a transaction.
  - But **why** do two partitions decide differently? **Why** wouldn't all partitions want to commit the transaction?

Intuition behind Deterministic Databases

- Deterministic databases aim to eliminate commit protocols as they are expensive.
- But **why** are commit protocols needed in the first place?
  - Two partitions may decide differently on the fate on a transaction.
  - But **why** do two partitions decide differently? **Why** wouldn't all partitions want to commit the transaction?
  - A partition would abort the transaction in two cases:
    - **Non-deterministic events**: failures.
    - **Deterministic events**: aborts forced by transaction logic .

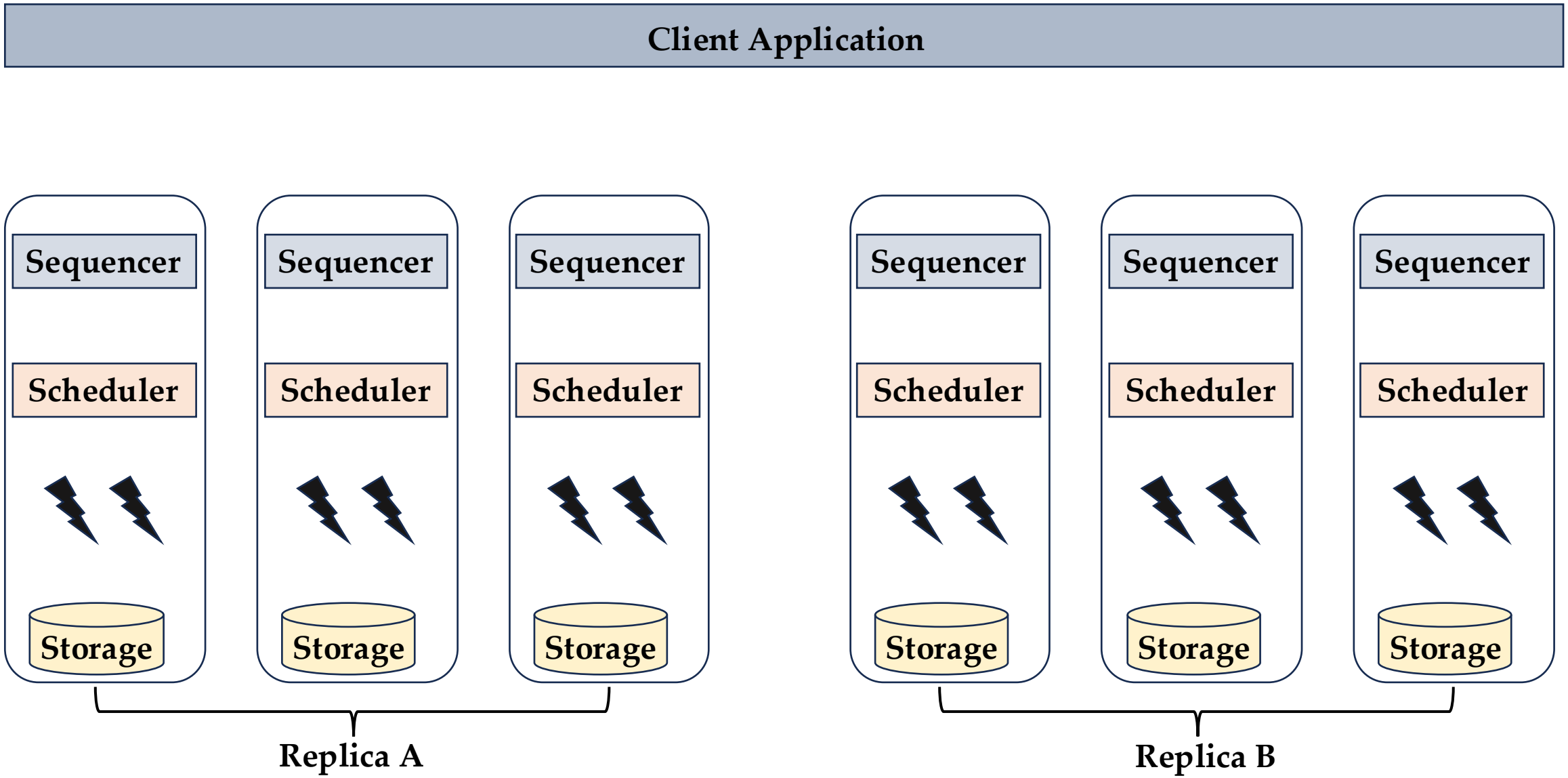
Calvin

Calvin

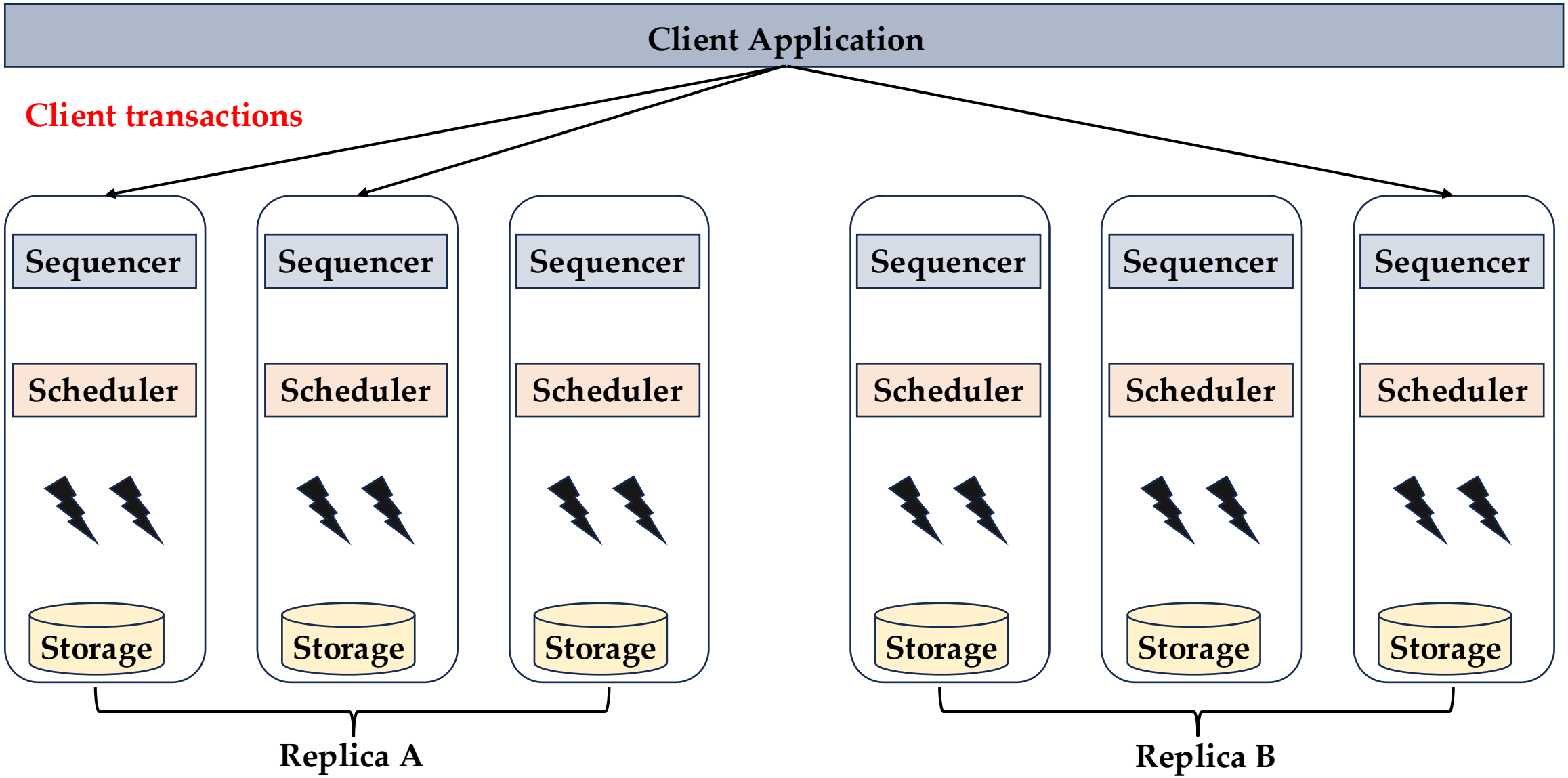
- A transactional architecture that supports the design of a deterministic database.

- Another key motivation: provide transactional support for applications that were moving away from the use of transactions.
  - NoSQL databases.

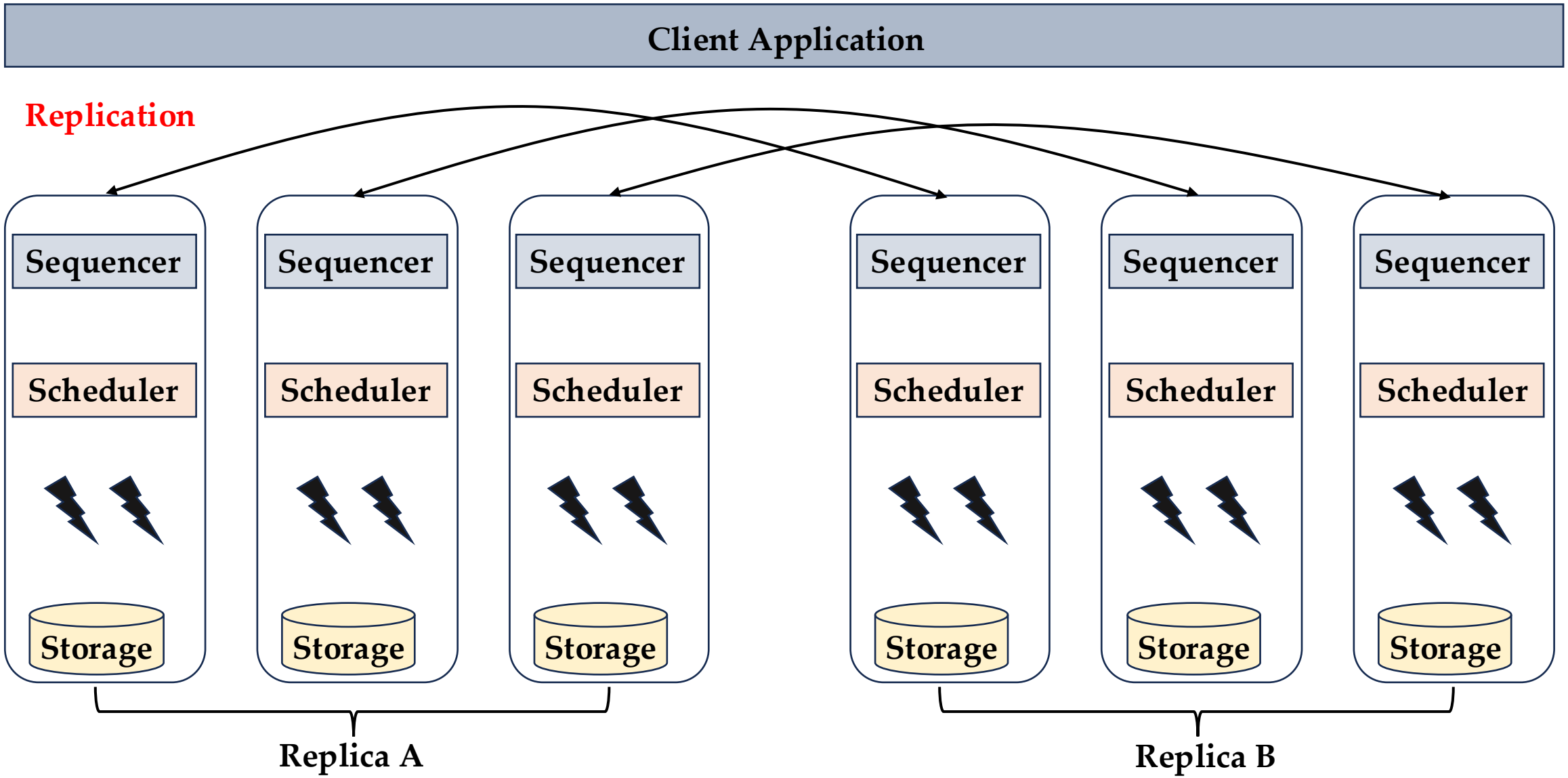
Calvin Architecture



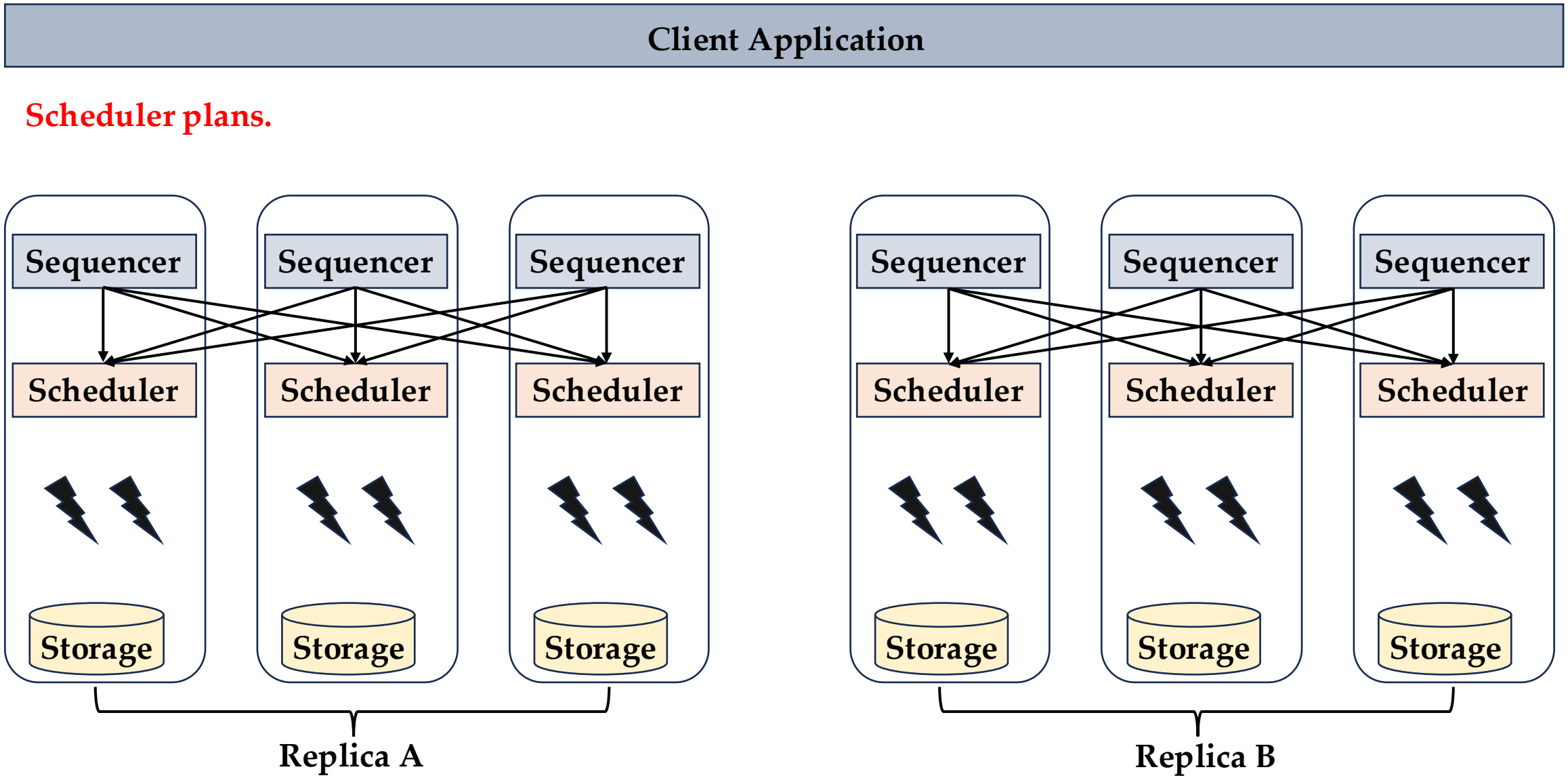
Calvin Architecture



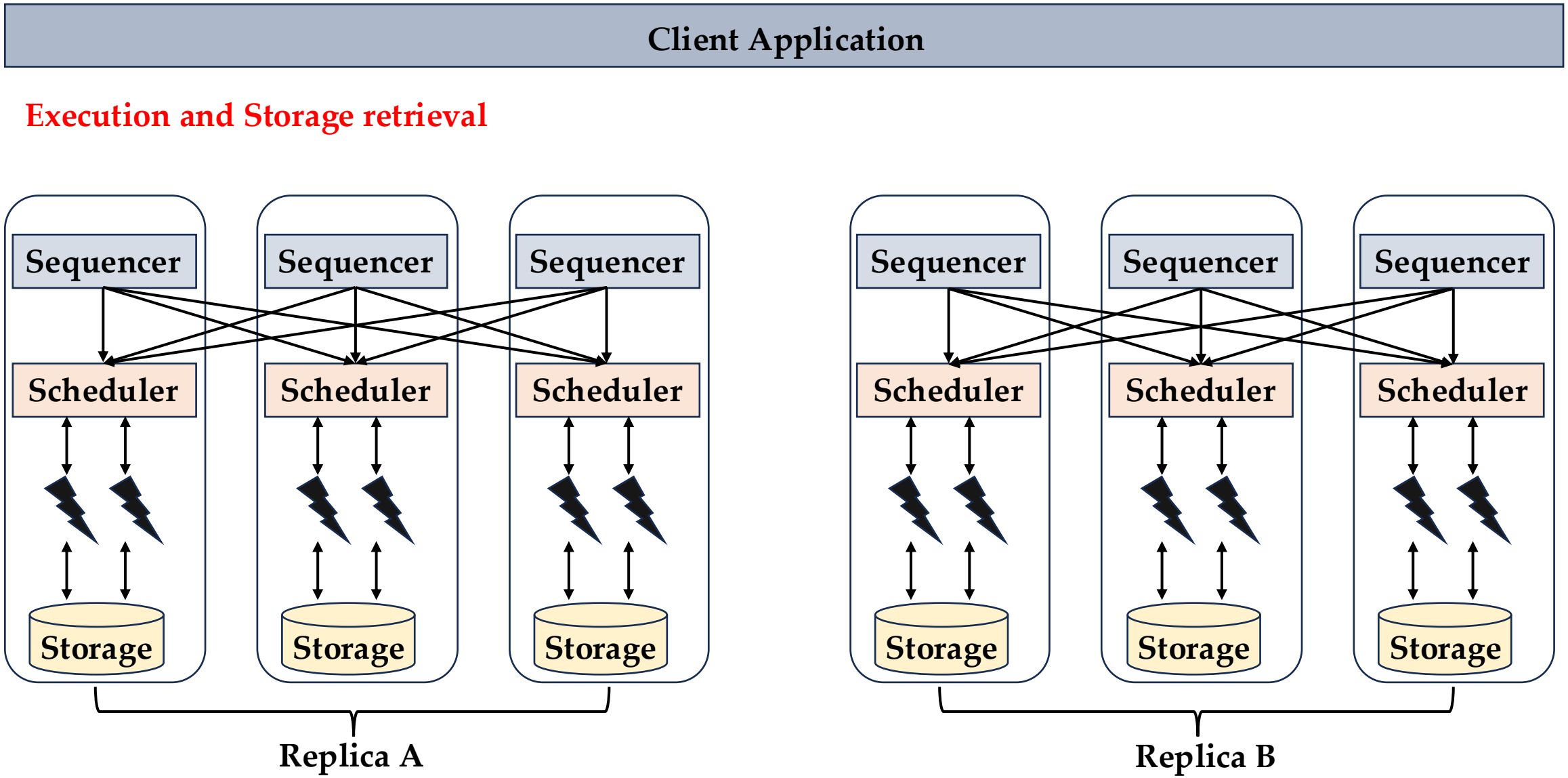
Calvin Architecture



Calvin Architecture



Calvin Architecture



Calvin Architecture

- Three key components:
  - Sequencer
  - Scheduler
  - Execution and Storage

Sequencer

- Calvin suggests having a sequencer at every partition of a replica. **Why?**

Sequencer

- Calvin suggests having a sequencer at every partition of a replica. **Why?**
  - Scalability → Single sequencer bottlenecks the system and single point of failure.

- Sequencer does several tasks:

Sequencer

- Calvin suggests having a sequencer at every partition of a replica. **Why?**
  - Scalability → Single sequencer bottlenecks the system and single point of failure.

- Sequencer does several tasks:
  - Collects incoming client transactions.

Sequencer

- Calvin suggests having a sequencer at every partition of a replica. **Why?**
  - Scalability → Single sequencer bottlenecks the system and single point of failure.

- Sequencer does several tasks:
  - Collects incoming client transactions.

  - Determines all the transaction inputs (read sets).

Sequencer

- Calvin suggests having a sequencer at every partition of a replica. **Why?**
  - Scalability → Single sequencer bottlenecks the system and single point of failure.

- Sequencer does several tasks:
  - Collects incoming client transactions.
  - Determines all the transaction inputs (read sets).
  - Replicates transactional inputs across all replicas.

Sequencer

- Calvin suggests having a sequencer at every partition of a replica. **Why?**
  - Scalability → Single sequencer bottlenecks the system and single point of failure.

- Sequencer does several tasks:
  - Collects incoming client transactions.
  - Determines all the transaction inputs (read sets).
  - Replicates transactional inputs across all replicas.

- It is the task of the sequencer to **create an order** for all the transactions that it has received!

Sequencer

- How does sequencer collect client transactions?
  - How long does it wait?

Sequencer

- How does sequencer collect client transactions?
  - How long does it wait?

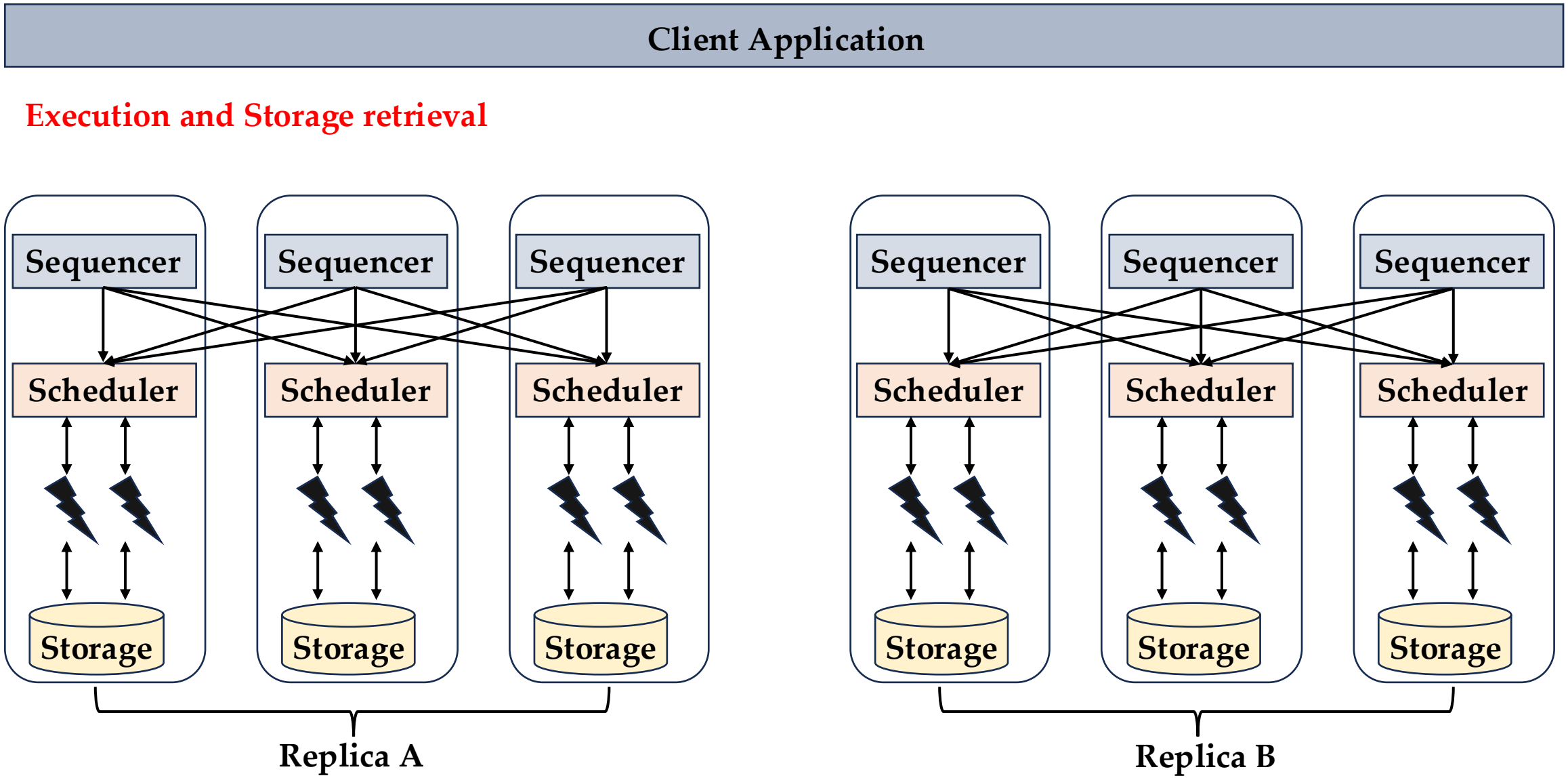
- Calvin follows an **epoch-based design**.
  - An epoch lasts a specific amount of time (say 10ms).

Sequencer

- How does sequencer collect client transactions?
  - How long does it wait?

- Calvin follows an **epoch-based design**.
  - An epoch lasts a specific amount of time (say 10ms).
  - During an epoch each sequencer collects as many transactions as it can.
  - Once an epoch expires, it creates a batch of collected requests and initiates replication.
  - Post this it forwards the batch to a scheduler.
  - Each batch has **a pair of identifiers**: sequencer no and epoch no.

Scheduler


Client Application
Execution and Storage retrieval
Sequencer
Sequencer
Sequencer
Scheduler
Scheduler
Scheduler
Storage
Storage
Storage
Replica A
Sequencer
Sequencer
Sequencer
Scheduler
Scheduler
Scheduler
Storage
Storage
Storage
Replica B

Scheduler

- Scheduler receives transaction batches from the sequencer at each partition.
  - How does it **merge** them into one batch**?**

Scheduler

- Scheduler receives transaction batches from the sequencer at each partition.
  - How does it **merge** them into one batch**?**

- Remember, each batch has a sequence number and an epoch number.
  - These numbers together can help in creating one batch.

  - For example, merging in the ascending order of sequencer identifiers (1,2,…)

Scheduler

- What else does the Scheduler do?

Scheduler

- What else does the Scheduler do?
- Initiate execution of the batch of transactions.
- What is the **first step** to execution?

Scheduler

- What else does the Scheduler do?
- Initiate execution of the batch of transactions.
- What is the **first step** to execution?
- Run a concurrency control protocol → We will run strict 2PL.

Strict 2PL in Calvin Scheduler

- To run strict 2PL, a transaction needs to borrow all the locks in the beginning.
- As read sets are known, Calvin runs a thread to request locks for local reads.
- Two changes for strict 2PL in Calvin:

Strict 2PL in Calvin Scheduler

- To run strict 2PL, a transaction needs to borrow all the locks in the beginning.
- As read sets are known, Calvin runs a thread to request locks for local reads.
- Two changes for strict 2PL in Calvin:
  - If transaction A occurs before B in the sequence order and they both need lock for the same variable x, then A must request lock on x before B.

Strict 2PL in Calvin Scheduler

- To run strict 2PL, a transaction needs to borrow all the locks in the beginning.
- As read sets are known, Calvin runs a thread to request locks for local reads.
- Two changes for strict 2PL in Calvin:
  - If transaction A occurs before B in the sequence order and they both need lock for the same variable x, then A must request lock on x before B.
  - Lock manager can only grant locks in the order transactions requested locks.

Execution

- Once a transaction has acquired all the locks, Calvin starts its **5-step execution**.

Execution

- Once a transaction has acquired all the locks, Calvin starts its **5-step execution**.
  1. Read-Write Set Analysis
  2. Perform Local Reads
  3. Serve Remote Reads
  4. Collect Remote Read Results
  5. Execution and Applying Write.

Read-Write Set Analysis

- **Analyze** a transaction's read-write sets.
  - Find out which of the reads and writes can be done locally.

Read-Write Set Analysis

- **Analyze** a transaction's read-write sets.
  - Find out which of the reads and writes can be done locally.

- **Determine** active and passive participants.
  - Active participants → remote nodes that include write sets.
  - Passive participants → remote nodes that include read sets.

Perform Local Reads

- **Fetch** all the records that need to be read and are present locally at the node!

- Essentially, a copy of the record (or pointer) is placed in the buffer.

Serve Remote Reads

- **Send** all the locally read records to all the active participants.

Collect Remote Reads

- Prior to executing the transaction, a node needs **access** to all the records (read sets).

- Need to collect all the remote records from all the participants.

Transaction Execution

- **Execute** the transaction logic.

- **Apply** local writes.

Challenges for Calvin and Deterministic Databases

Challenges for Calvin and Deterministic Databases

- Often, it is not possible to determine all the records a transaction needs to read prior to its execution.
  - For example, due to complex code logic

Challenges for Calvin and Deterministic Databases

- Often, it is not possible to determine all the records a transaction needs to read prior to its execution.
  - For example, due to complex code logic
- Calvin needs to run a **reconnaissance phase** to handle such cases.
- Reconnaissance tries to execute the transactions to discover all the read-write sets.

Can we do something better?