Large Scale Systems
CS 410 / 510

Suyash Gupta

Assistant Professor

Distopia Labs and ORNG

Dept. of Computer Science

(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

UNIVERSITY OF OREGON

Welcome!

Course Webpage:

https://gupta-suyash.github.io/cs410-spring26.html

Course Canvas:

https://canvas.uoregon.edu/courses/265485

Instructors

- **Suyash Gupta**
  - Office Hours: Tuesday, 10-11am, Deschutes 334

TextBooks

- **Main Course Book:**
  - Principles of Distributed Database Systems (4/e by Tamer Ozsu).
  - Fault-Tolerant Distributed Transactions on Blockchains (1/e by Gupta).

Grading Scheme

	Percentage
Project	60%
Presentations	20%
Class Participation	10%
Attendance	10%

No Exams!!!

Rubric

- **97% - A+**
- **92% - A**
- **87% - A-**
- **82% - B+**
- **77% - B**
- **72% - B-**
- **67% - C+**

Attendance

- Every student can **miss up to 2 classes** without getting penalized for attendance.

Project

- Projects should be completed in **groups of 3 or 4**.
- Every student in the group needs to **state their contribution**.

Project

- The goal of the project is to build a **MiniSpanner**, like **Google's Spanner**.
- Thus, the project is divided into **4 incremental Assignments**.
- Each assignment carries **15%** of the credit.
- Each assignment submission is a **Latex report (no code)!**
- **Assignments for CS 410 and CS 510 students will differ a little**.

Project Assignment Schedule

	Date Release	Date Due
Assignment 1	04/02/2025	04/16/2025
Assignment 2	04/16/2025	04/30/2025
Assignment 3	04/30/2025	05/18/2025
Assignment 4	05/18/2025	06/02/2025

- **All assignments are due at 11:59pm. Post that it will be considered as late.**

Late Days

- Each group has **2 Late Days**.
- You can use these late days in any assignment.
- Late Days allow you to miss the deadline.

Plagiarism

- **Please don’t plagiarize.**
- **Just don’t!**
- **We will apply the harshest UO policy available to us to deal with plagiarism!**
- **More details on the webpage.**

Presentations

- In the week of **6/1 – 6/12**, we will have presentations.

- Each group will get **30 minutes** to present the design of their **MiniSpanner**.
  - **15 min** presentation through slides (**5 min** per student)
  - **5 min** for live demo.
  - **10 min** Q/A.

Large Scale Systems
CS 410 / 510

Lecture 1:

Overview: Why Large Scale Systems

Suyash Gupta

Assistant Professor

Distopia Labs and ORNG

Dept. of Computer Science

(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

UNIVERSITY OF OREGON

Meet Alice

This is Alice.

Meet Alice

This is Alice.

She makes designer pots and wants to sell them online.

What Should Alice Do?

This is Alice.

She makes designer pots and wants to sell them online.

Step 1: Creates an HTML Page

Alice creates an HTML/CSS page that lists all the pots and their price.

What Next? Is Alice done?

Alice creates an HTML/CSS page that lists all the pots and their price.

Step 2: Upload the Page on the Web

How to upload a page on the web?

Step 3: Buy a Domain and Set up a Server

- **Alice buys a domain: PotsForAll.com**
- **Alice buys a computer and sets it up as a server for this website.**

Why the term Server?

Why the term Server?

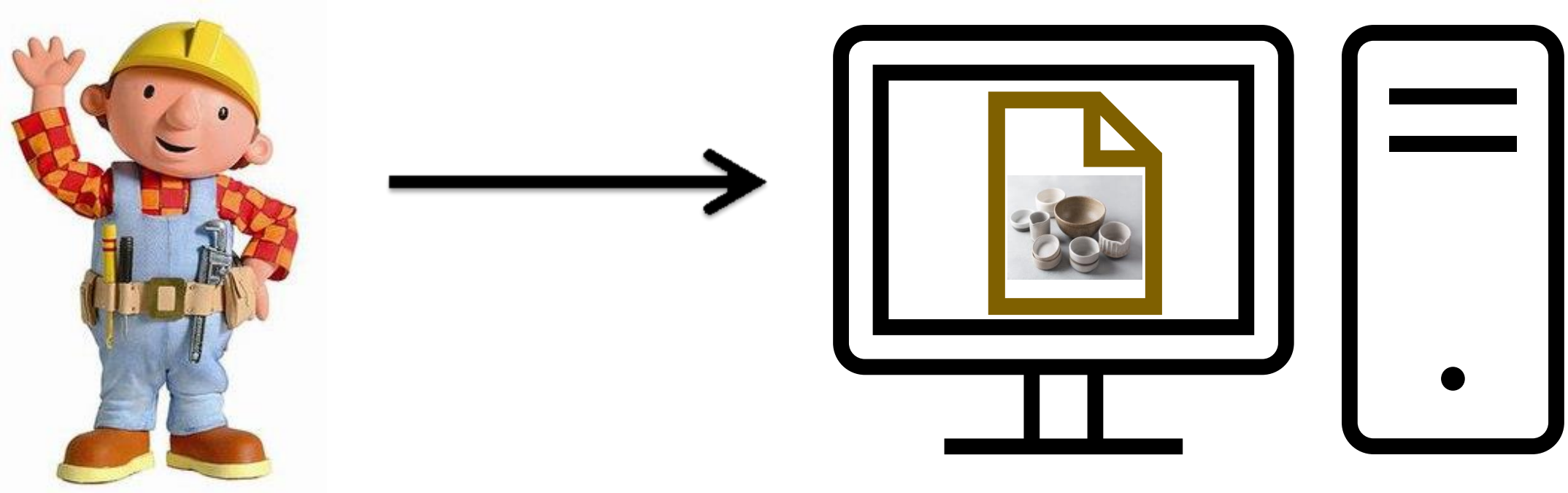
It serves all client requests.

Why the term Server?

This is Bob.

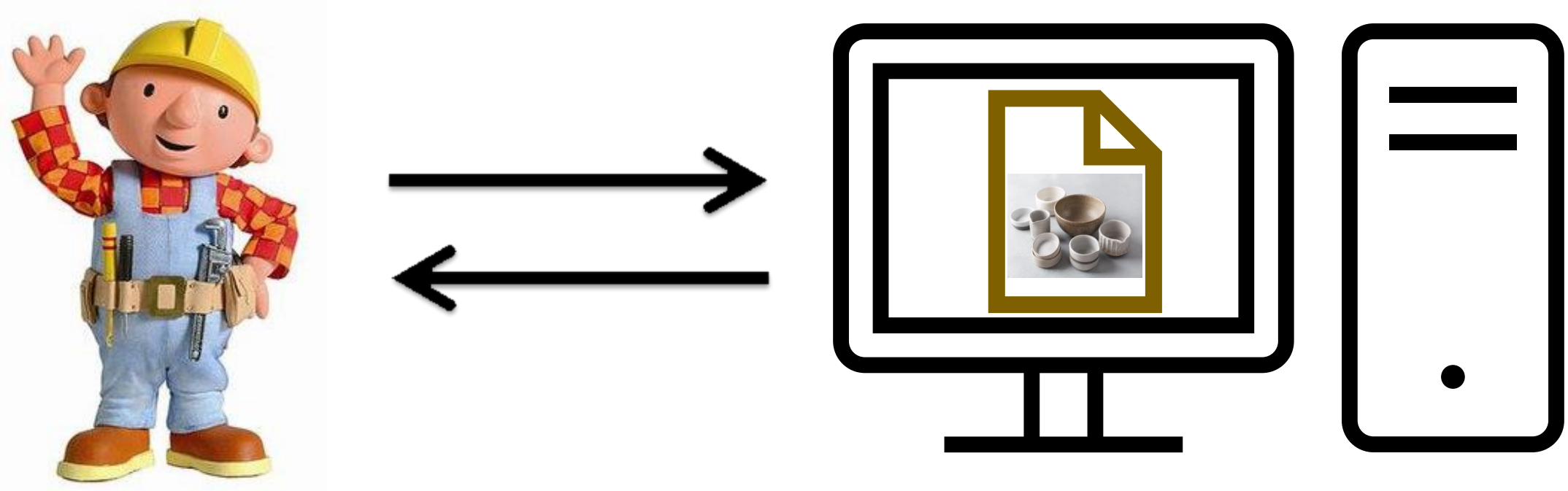
He likes the pots that Alice makes and wants to purchase one online.

Why the term Server?



Bob sends server a request to buy a brown pot.

Why the term Server?



Bob sends server a request to buy a brown pot.

Once Bob's payment succeeds, the server sends Bob an order confirmation receipt.

Challenge 1: Failures

What if Alice's computer gets infected with a virus?

Or what if Alice's computer crashes?

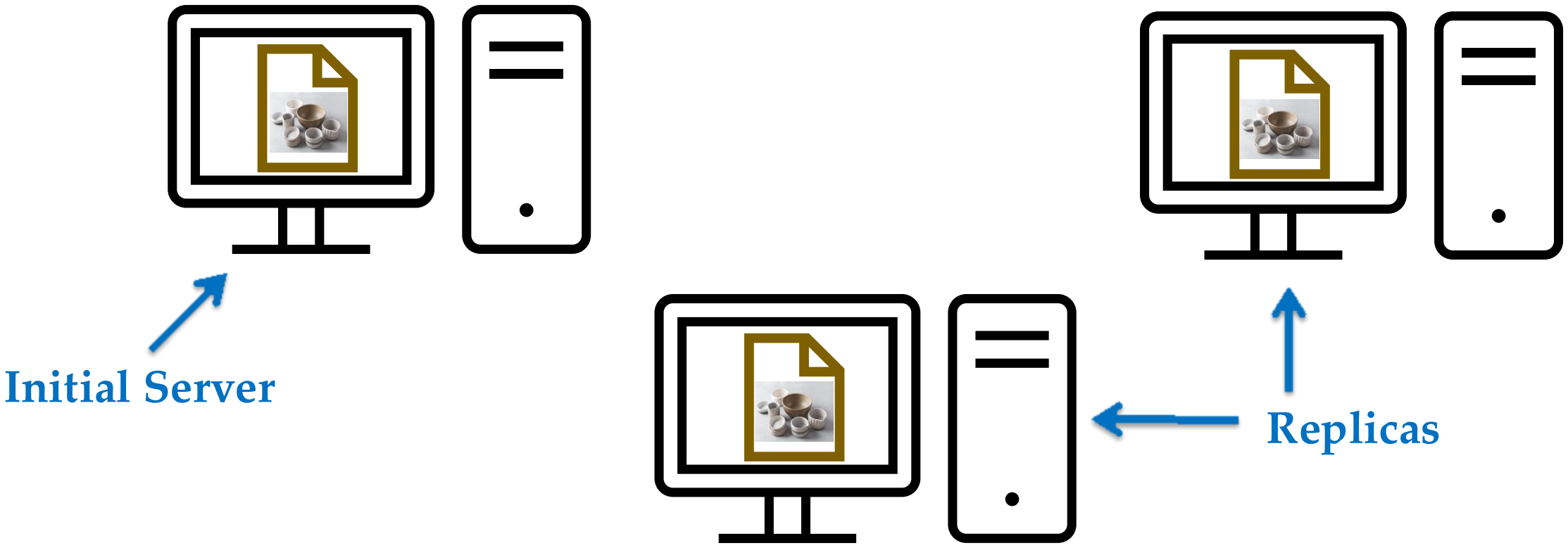
How will she fulfill client requests?

Need for Replication

Alice should replicate her webpage on multiple computers.

Have multiple computers store a copy of her webpage.

How does Replication help?



Alice now has replicas of her webpage.

How does Replication help?

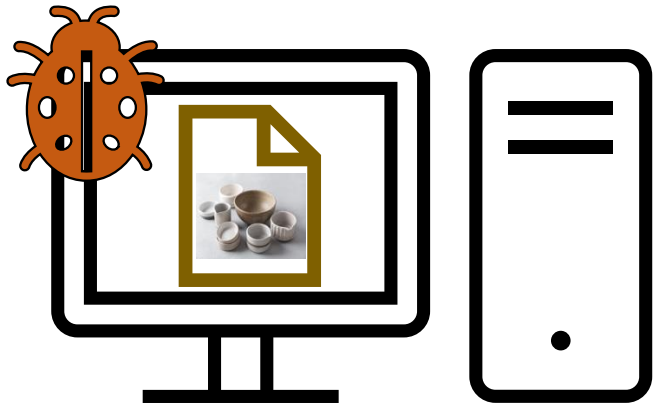
Alice now has replicas of her webpage.

If one computer has a virus, she can quickly make another computer as server.

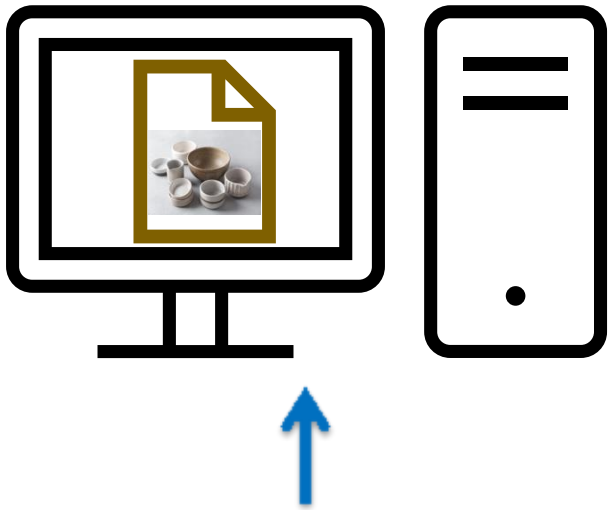
How does Replication help?

All future client requests will be re-directed to new server.

What Next? Is Alice done?



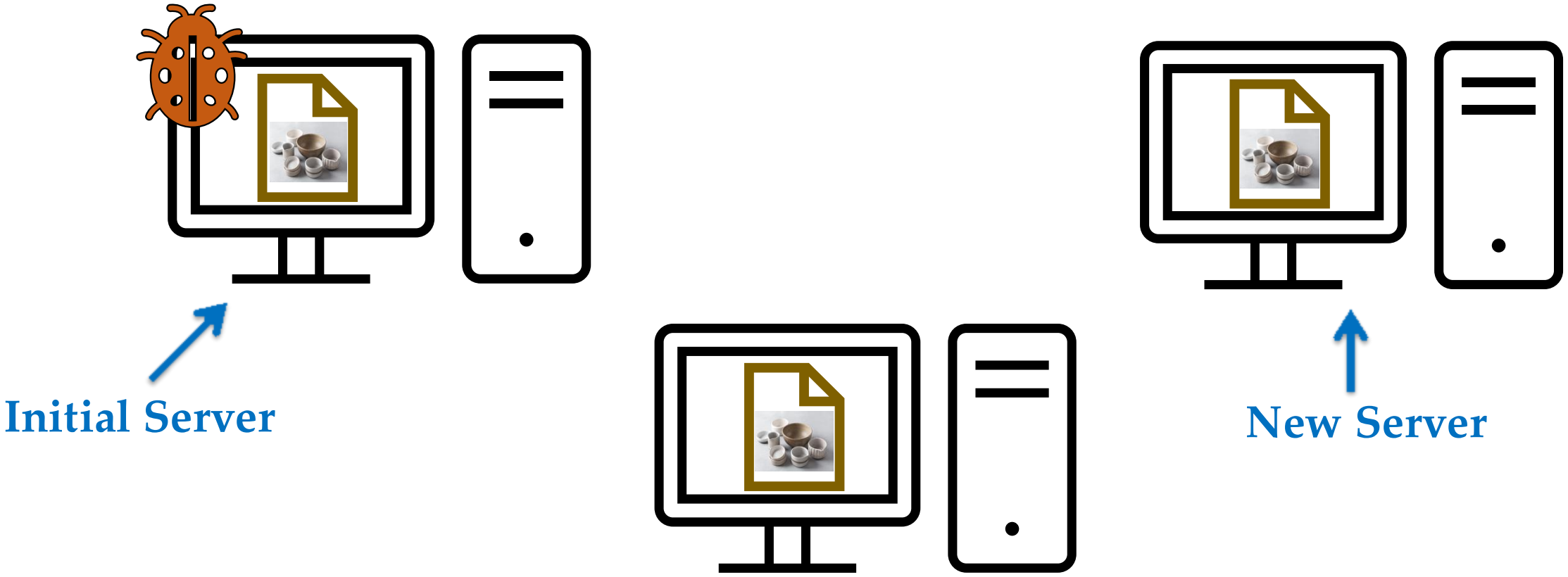
Initial Server



New Server

Any other challenges?

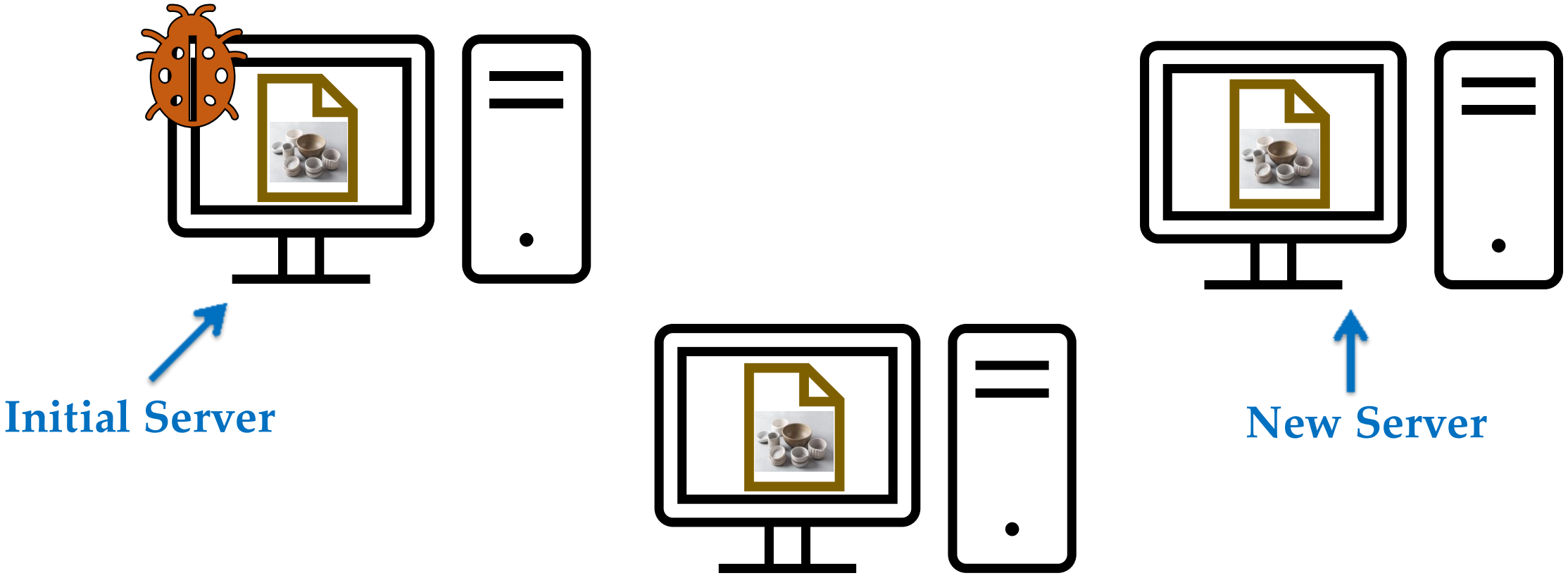
Challenge 2: State Update



What state does the new server starts from?

Loss of client data?

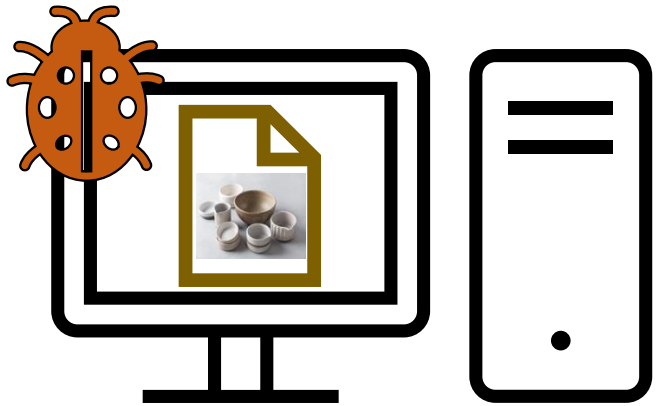
Need for Consistency



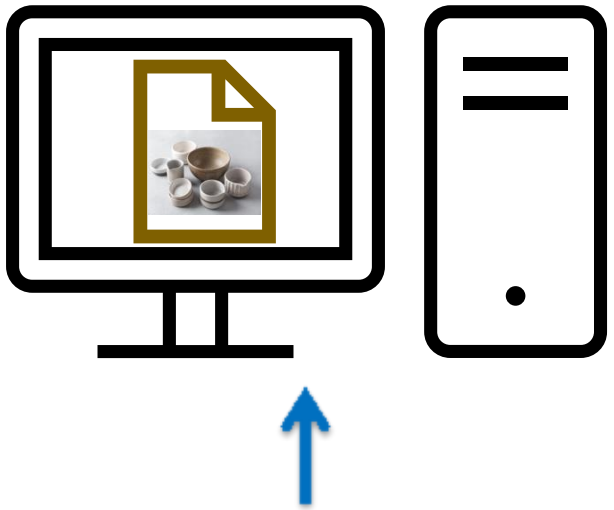
New Server should include all the state and account updates of the old server.

Need a protocol to continuously updates all replicas → keep replicas consistent.

What Next? Is Alice done?



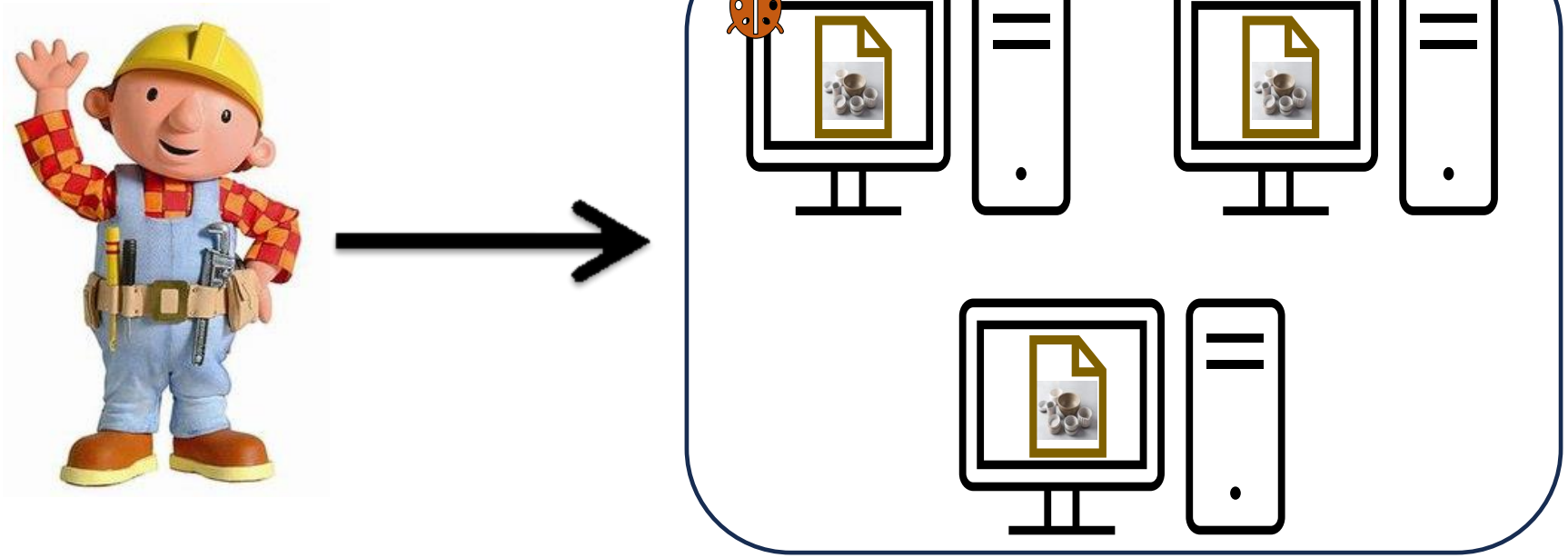
Initial Server



New Server

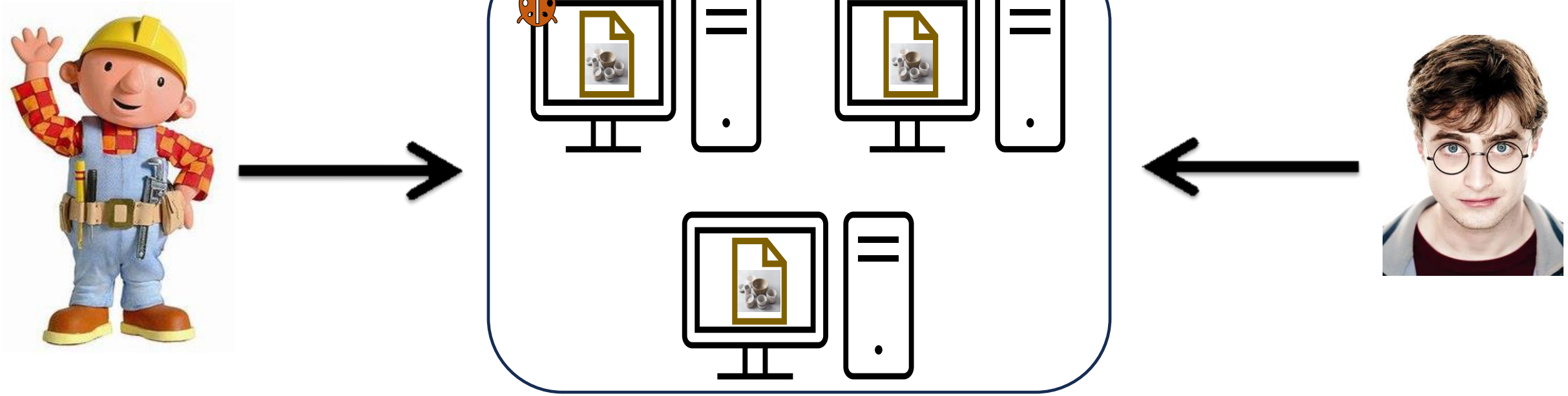
Any other challenges?

Challenge 4: Throughput



Bob wants to buy a cup.

Challenge 4: Throughput



Bob wants to buy a cup.

Harry wants to buy a cup.

Need for High Throughput

The system should be able to process multiple client requests.

Should complete a large number of client requests per unit of time.

Challenge 5: Ordering

Say there is only one golden cup and both Bob and Harry send a request to buy the cup.

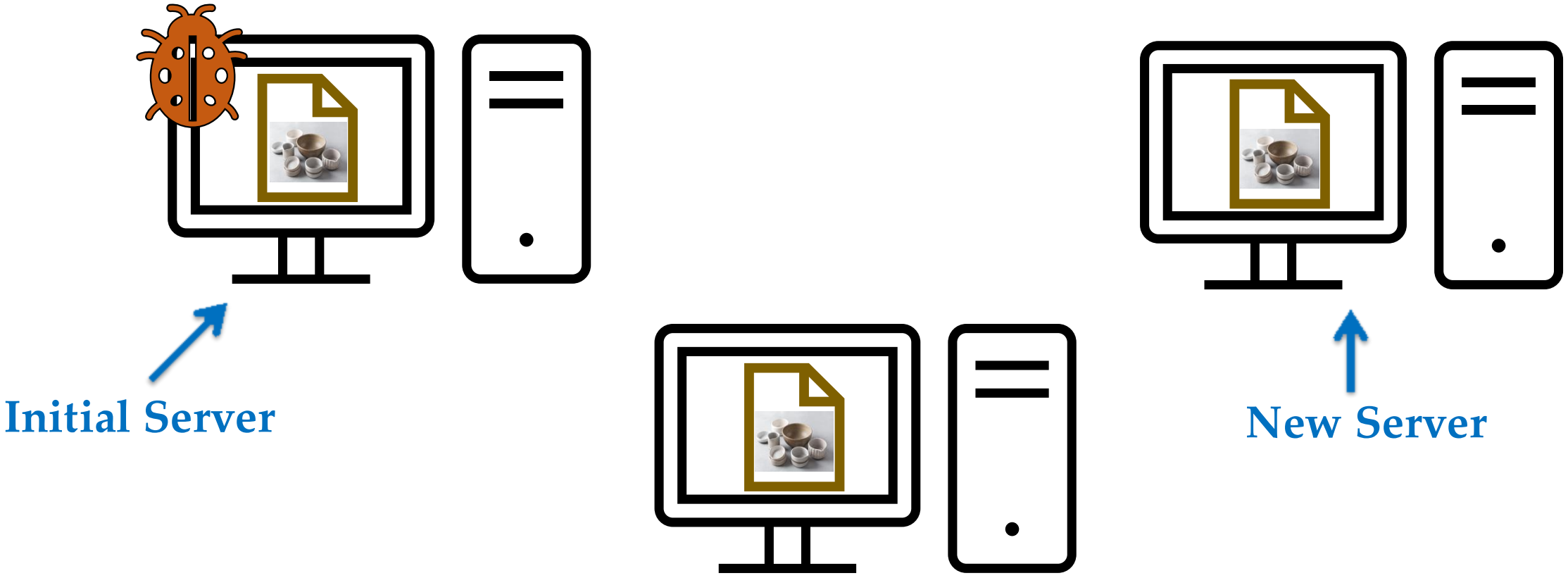
Who gets the cup?

Need for Transaction Ordering

The system should fairly order client requests/transactions.

Need to decide who gets the golden cup based on various factors, such as time of arrival of client transactions.

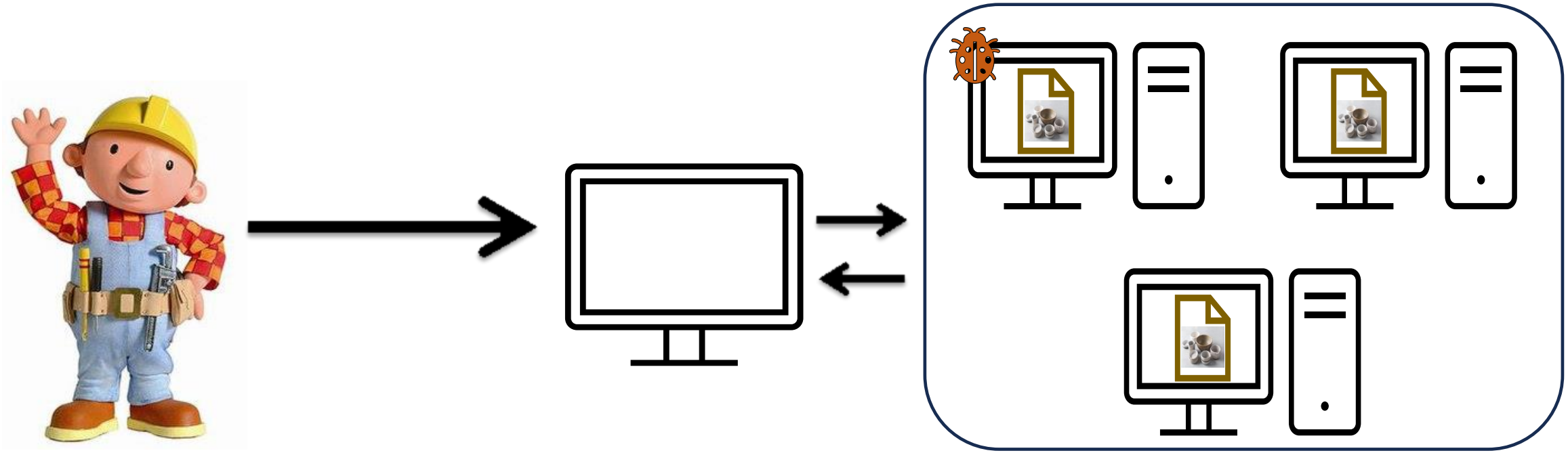
Challenge 6: Client View



How does client know which server to connect to?

How does client know that the old server failed and it needs to connect with the new server?

Need for a Proxy



Client

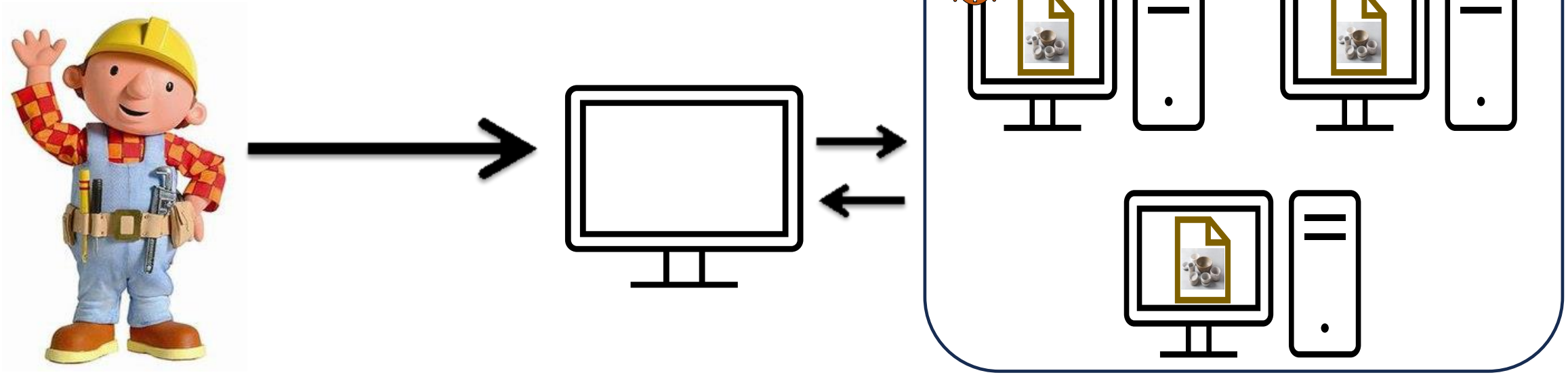
Proxy/ Frontend

Replicated Server/ Backend

Client only talks with the proxy.

Proxy knows the current server and delivers all client requests to server.

What Next? Is Alice done?



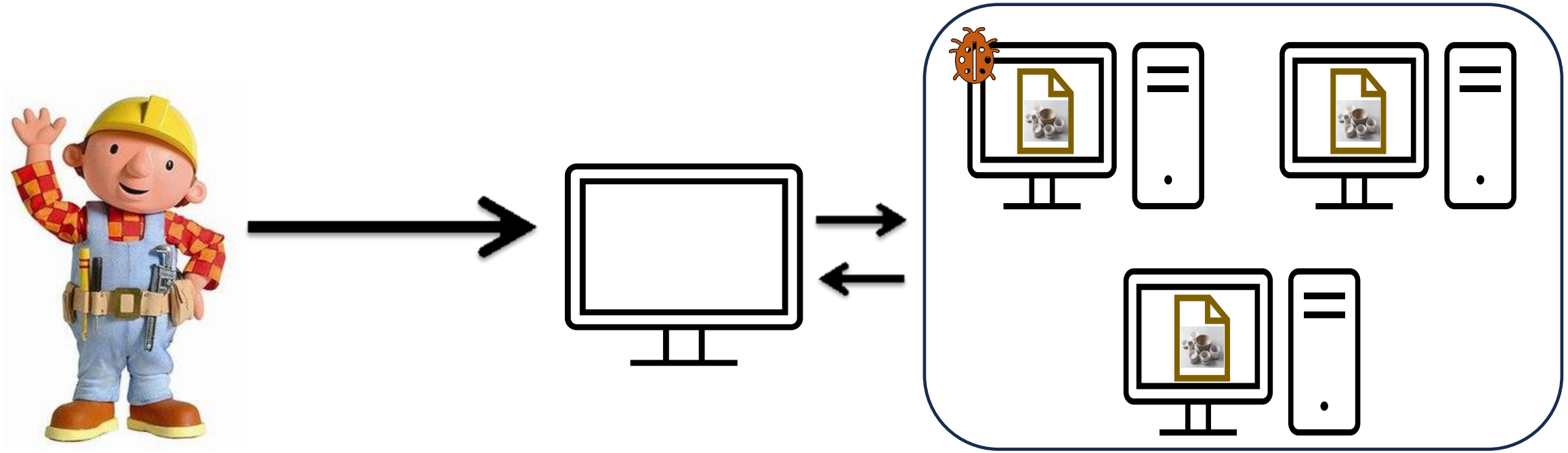
Client

Proxy/ Frontend

Replicated Server/ Backend

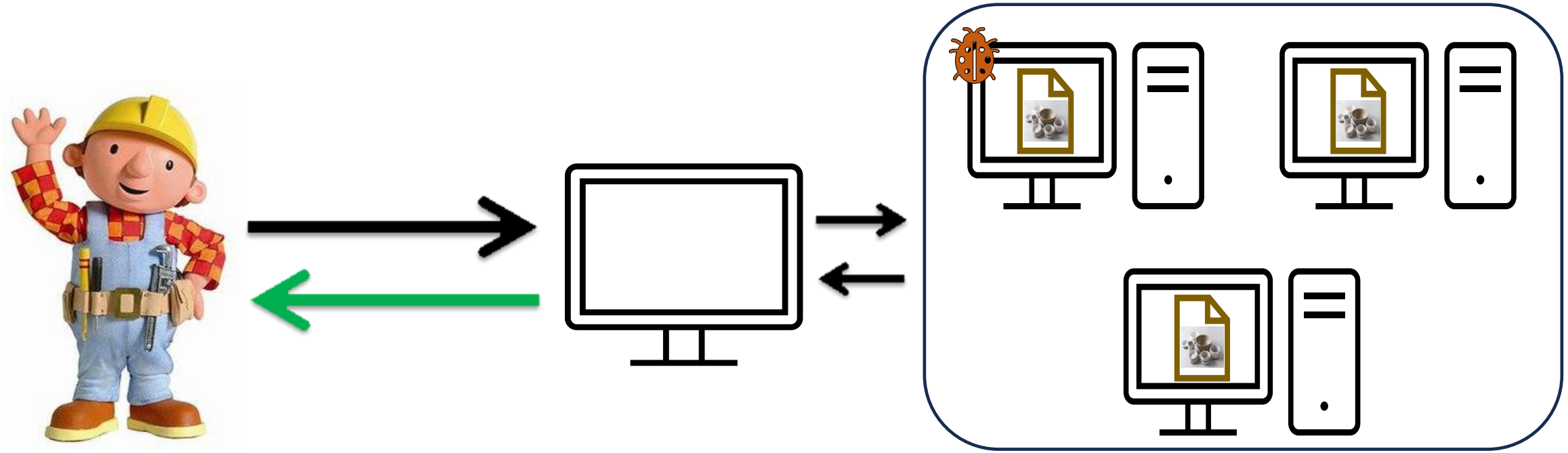
Any other challenges?

Challenge 7: Client Requirements



What other requirements/ expectations can clients have from the system?

Need for Responsiveness

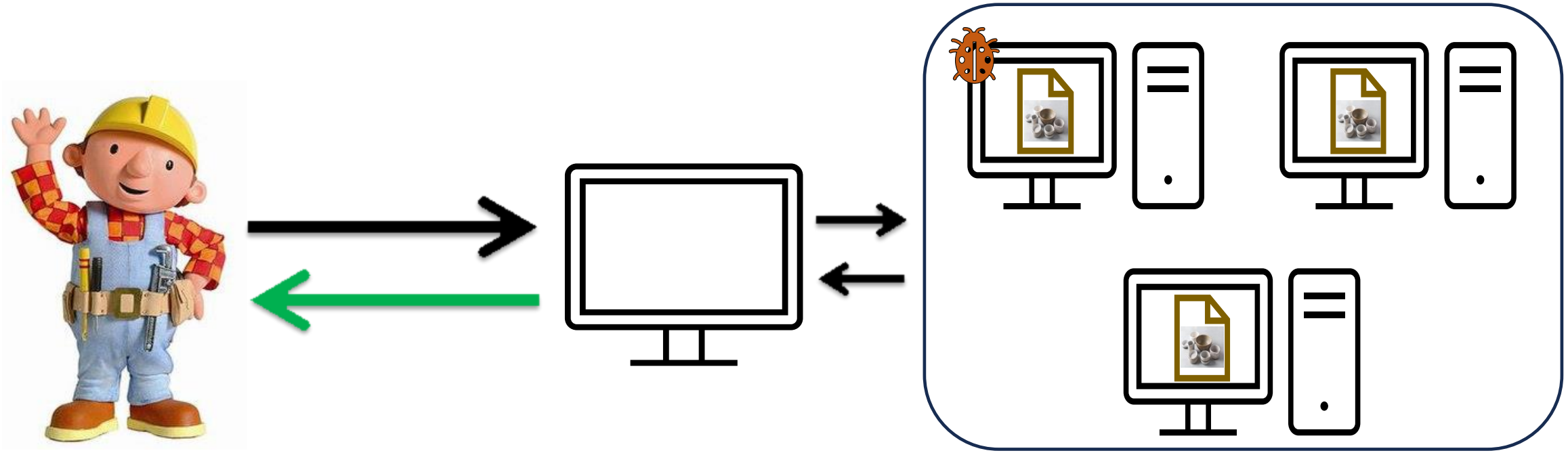


Client Proxy/ Frontend Replicated Server/ Backend

Bob needs a response back from the system to know that his request was processed.

Bob somehow needs to verify that he received a correct response!

Need for Low Latency



Client Proxy/ Frontend Replicated Server/ Backend

Bob needs a response back from the system as fast as possible.

A system should thrive to reduce the time a client sends a request to time when the client receives a response for its request.

System Modeling

When designing a system, a developer needs to lay down the system model, which requires stating various properties.

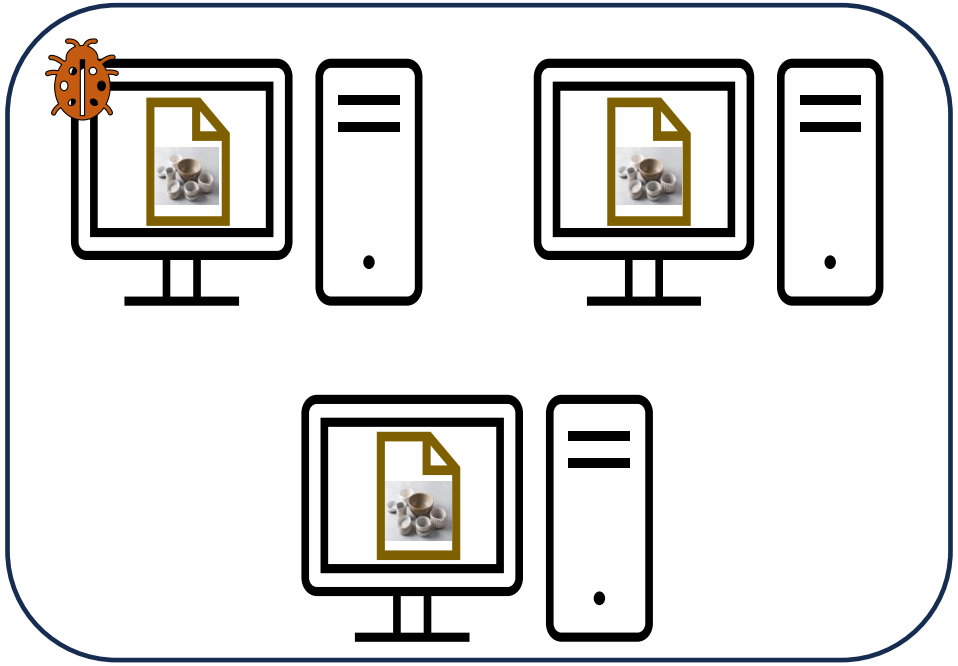
- **System Properties**
  - **Safety** – A system is safe if it guarantees consistency.
  - **Liveness** – A system is live if it is available despite failures.
  - **Responsiveness** – A system is responsive if it sends a response back to the client.

Note: Each modern system guarantees these properties under some network model.

System Modeling

- **System Metrics**
  - **Throughput** – Number of transactions processed per second.
  - **Latency** – Time difference between a client sends a request to the time client receives a response back from the server.

Need for Scalability



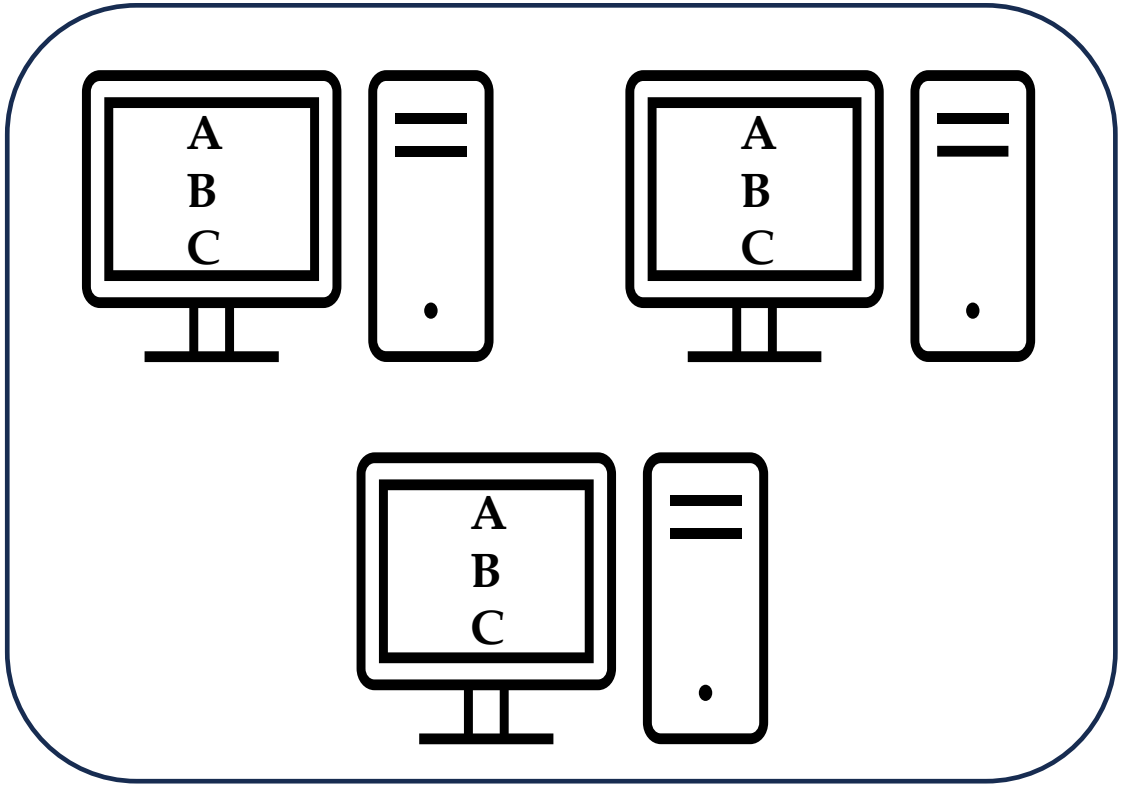
What if Alice's website becomes famous and has to handle billions of client requests daily, like Amazon!

Is continuously replicating billions of requests feasible? What can Alice do?

Replication vs. Sharding

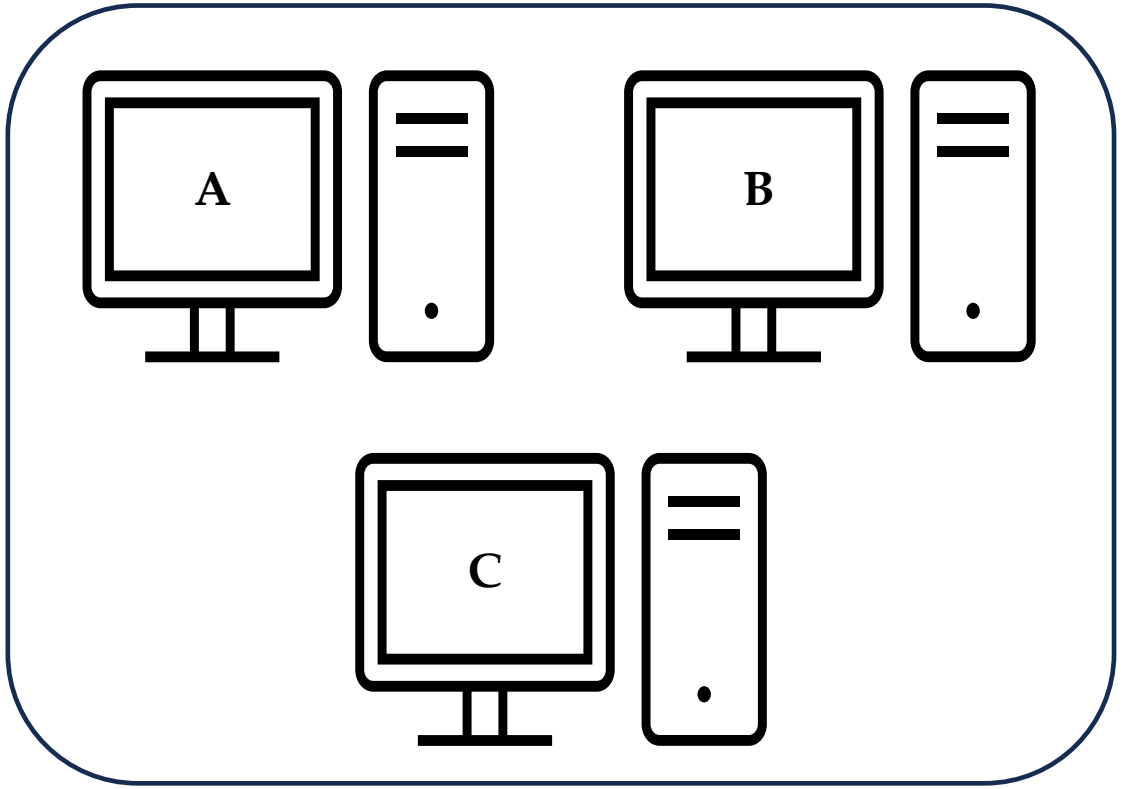
- We saw earlier that replication is necessary.
- But, can we also partition data.
- Sharding allows dividing data into distinct groups.

Need for Scalability



Fully Replicated System

Fully Sharded System



Sharded Replicated System

- As sharding only supports data partitioning.
- We still need availability.
- So, internally each shard can be replicated.
- Google's Spanner is an example of sharded replicated system.

Case Study:
How did Google perform massive compute?

Case Study:
How did Google perform massive compute?

- Even in early 2000's, Internet was large.
- Google wanted to process a large amount of data.
- Often their developers wanted to apply same operations to the data.
- They wanted to use the 1000s of computers they had to quickly perform operations.
- So, what do you think they did?

Case Study:
How did Google perform massive compute?

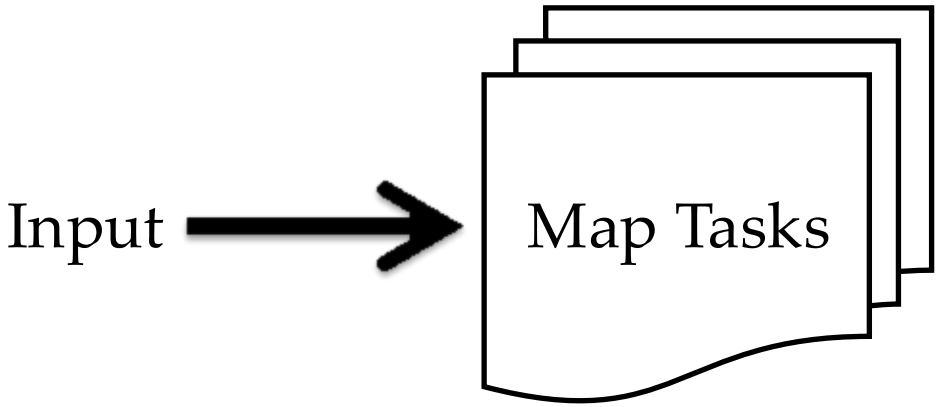
- Example Operations: Sorting data, counting words, etc.
- Solution: **Map Reduce Framework**!
- Google published the seminal paper on Map Reduce.

Case Study: Map Reduce

- Map Reduce architecture is a pipelined architecture.

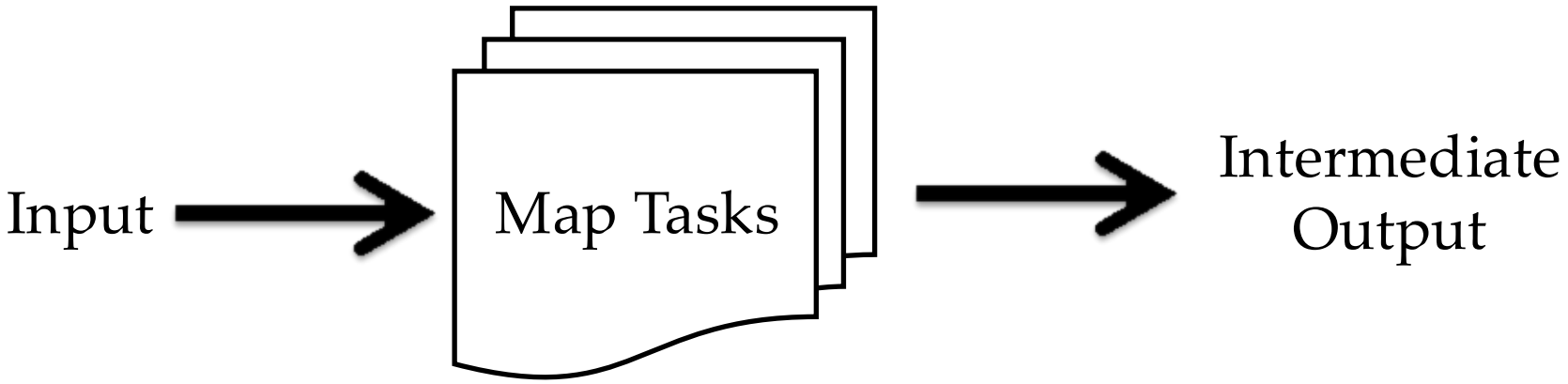
Case Study: Map Reduce

- Map Reduce architecture is a pipelined architecture.



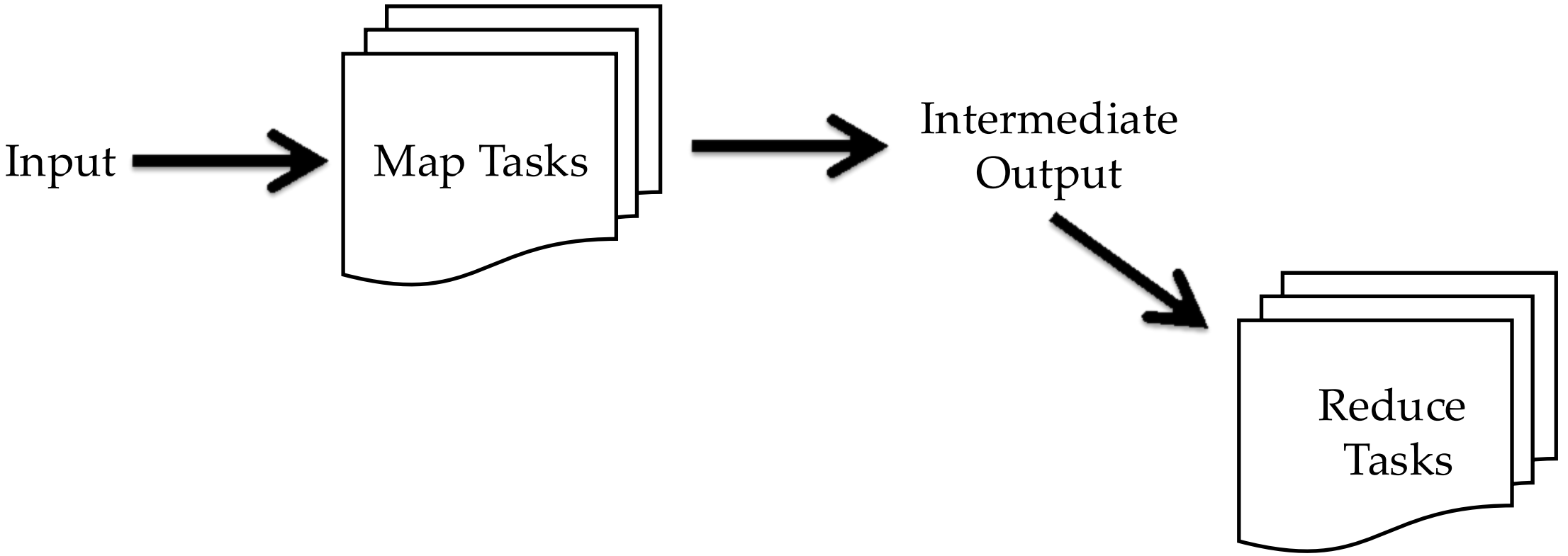
Case Study: Map Reduce

- Map Reduce architecture is a pipelined architecture.



Case Study: Map Reduce

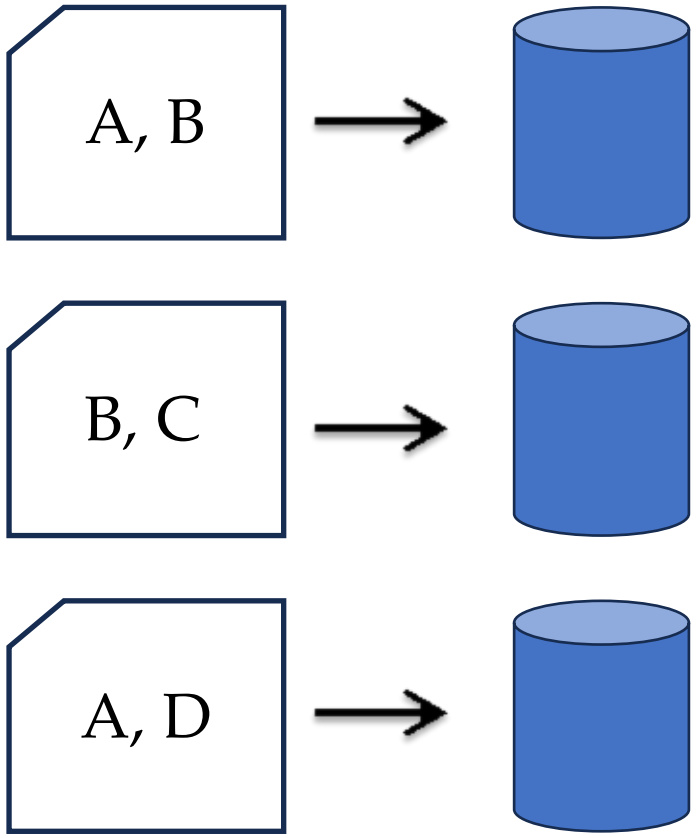
- Map Reduce architecture is a pipelined architecture.



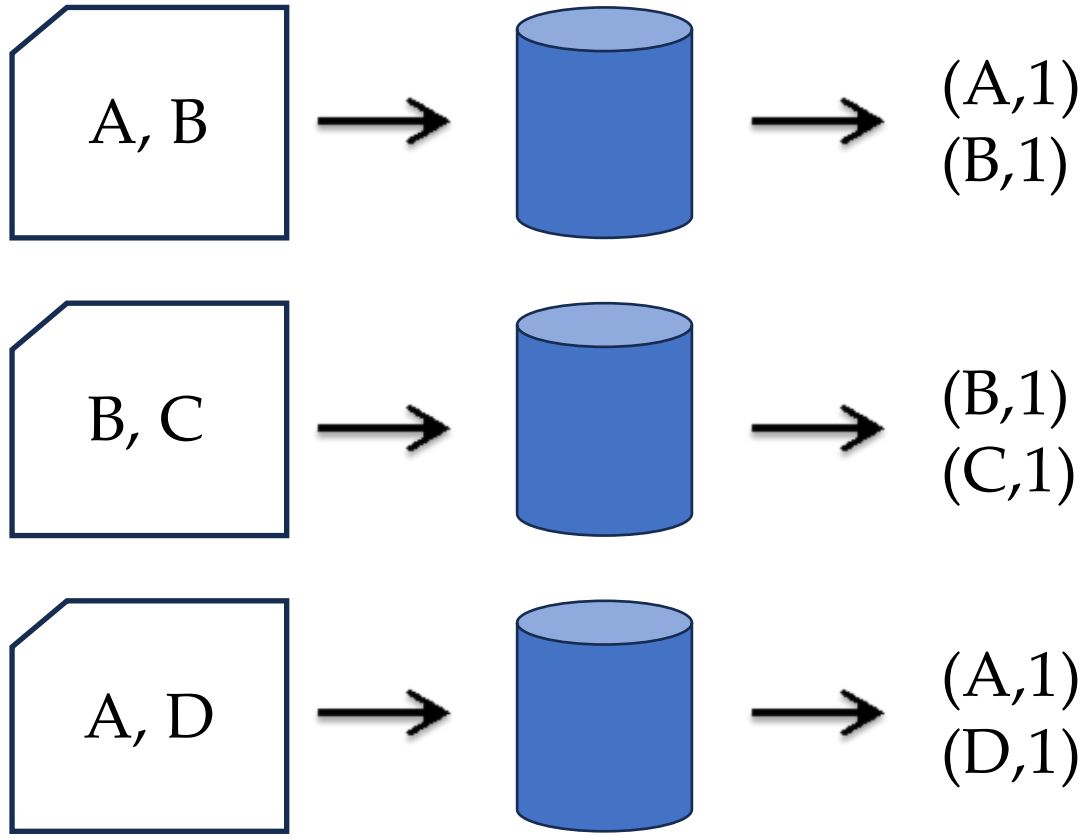
Case Study: Map Reduce

- **Key Requirement**: The tasks should be independent to allow split to happen.
- Lets take an example: Count the total words in a file.
- The map function takes a set of files as inputs and returns key, value pairs that identify each word.
- The reduce function will try to aggregate them.

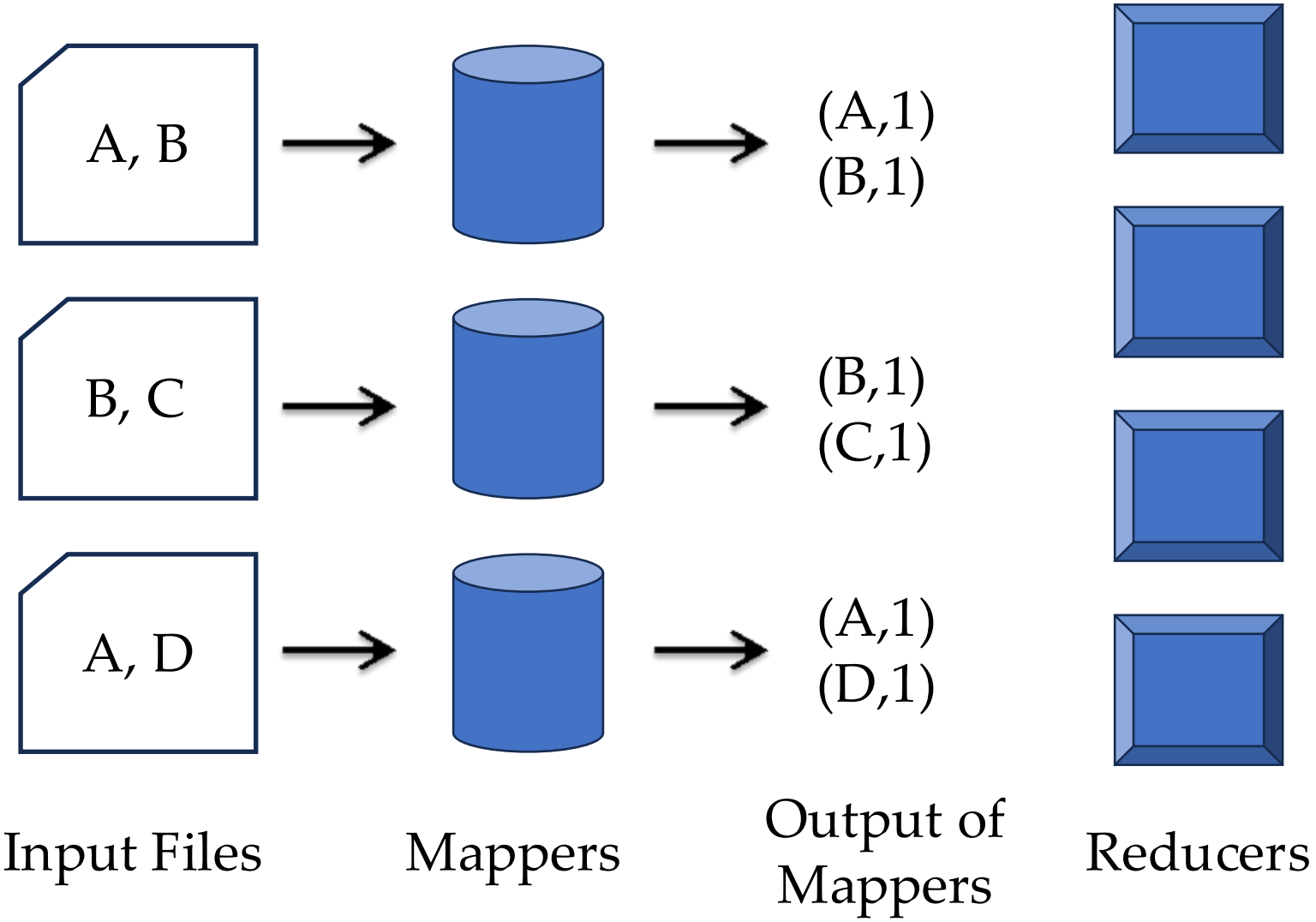
Case Study: Map Reduce



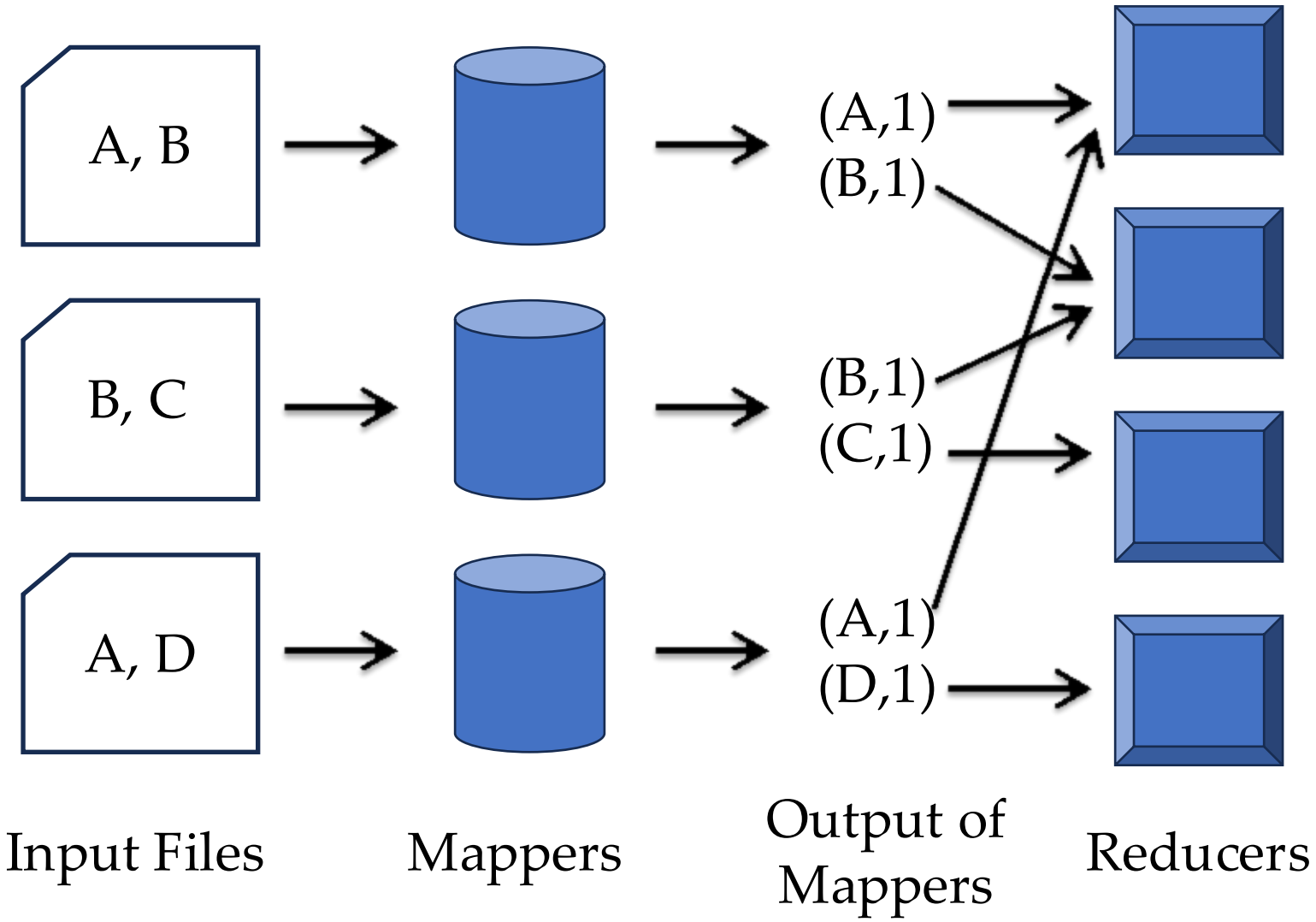
Case Study: Map Reduce



Case Study: Map Reduce



Case Study: Map Reduce



Case Study: Map Reduce