

# Concurrency in Go

CS 240 – Fall 2019

Rec. 2

# Housekeeping

Name's Arnaud Dethise, PhD student. Nice to meet you.

Second TA for this course

Will also have office hours (TBA)

# Housekeeping

- Assignment 1 deadline is coming soon.
- Should have read and started the assignment.
- If progressing correctly, should have working `mapF ()`

# Concurrency in Go

## Part 1 - MapReduce

CS 240 – Fall 2019  
Rec. 2

# Map Reduce

Wikipedia:

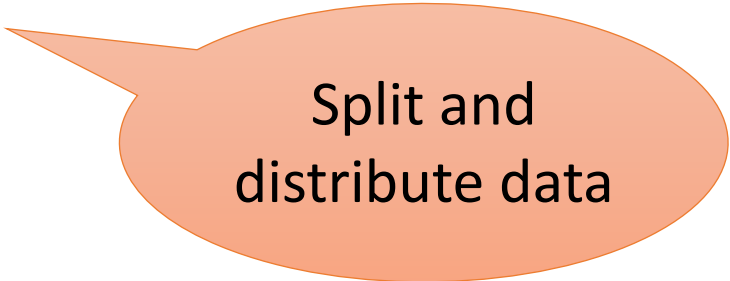
*“MapReduce is a **programming model** and an associated implementation for processing and generating **big data** sets with a **parallel, distributed** algorithm on a **cluster**.”*

In other words, a general and scalable solution to deal with big data computation on multiple machines.

# Abstract Map Reduce

**map(key, value) -> list(<k', v'>)**

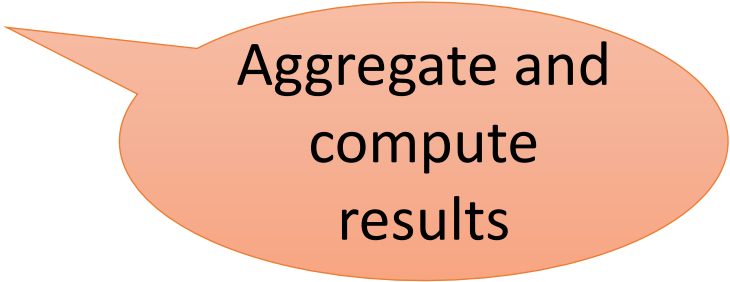
- Apply function to (key, value) pair
- Outputs set of intermediate pairs



Split and  
distribute data

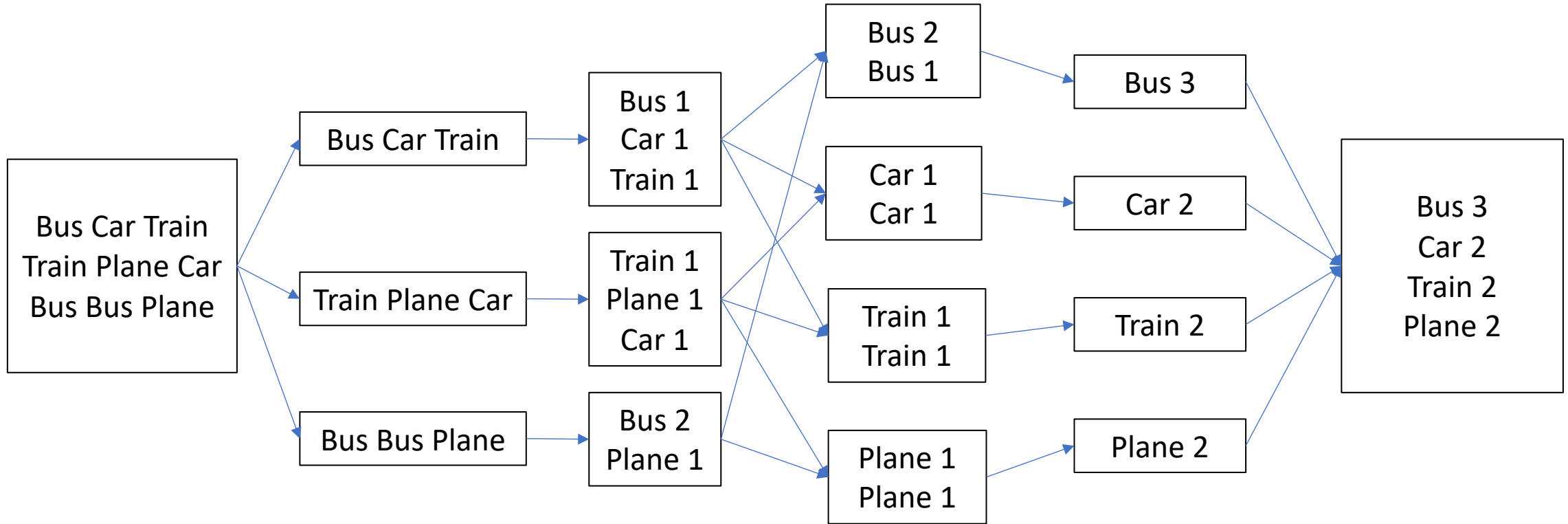
**reduce(key, list<value>) -> <k', v'>**

- Applies aggregation function to values
- Outputs result



Aggregate and  
compute  
results

# Word Count – The *Hello World* of Map Reduce



Splitting

Mapping

Intermediate  
Splitting

Reducing

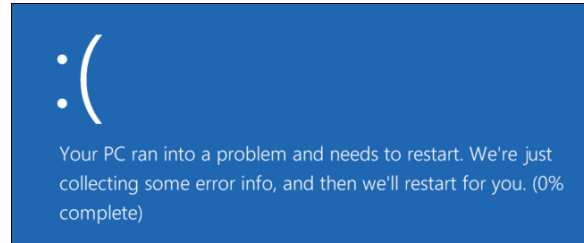
Combining

`doMap()`

`doReduce()`

# A Motivating Problem for Map Reduce

“Find me the closest Starbucks to KAUST.  
Actually, I’ll give you a place and something to look for,  
and you find me the closest one.  
Here’s a 1 TB text file ... good luck”



GPS Coordinates	Site Name	
[22.3, 39.1]	Tim Hortons	} In KAUST
[22.2, 39.1]	KAUST Library	
[35.7, 139.7]	Starbucks	} In Tokyo, Japan
...	...	



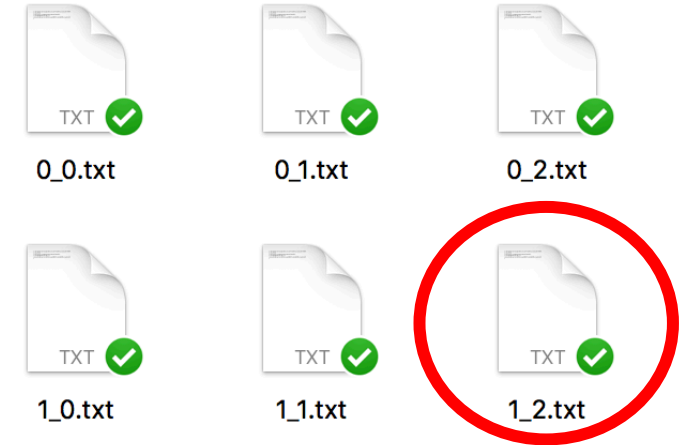
# A Motivating Problem for Map Reduce

## GPS Coordinates

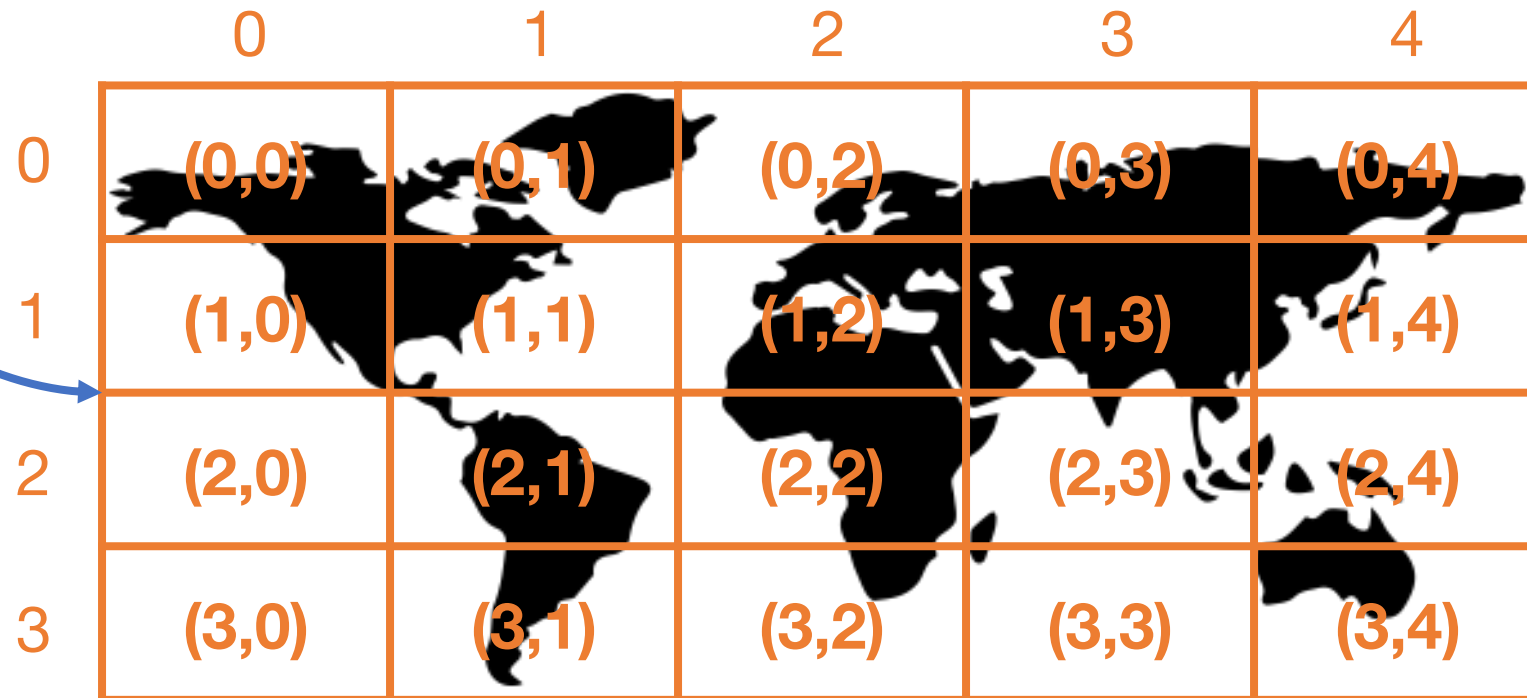
[22.3, 39.1]  
[22.2, 39.1]  
[35.7, 139.7]  
...

## Site Name

Tim Hortons  
KAUST Library  
Starbucks  
...



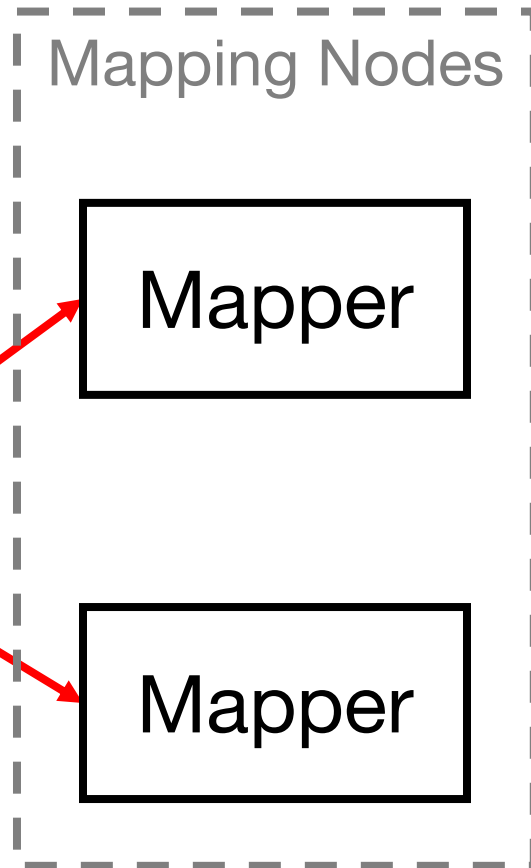
Map to  
grids



Reduce to  
single files

# Split the File and Map Each Chunk Independently (1/2)

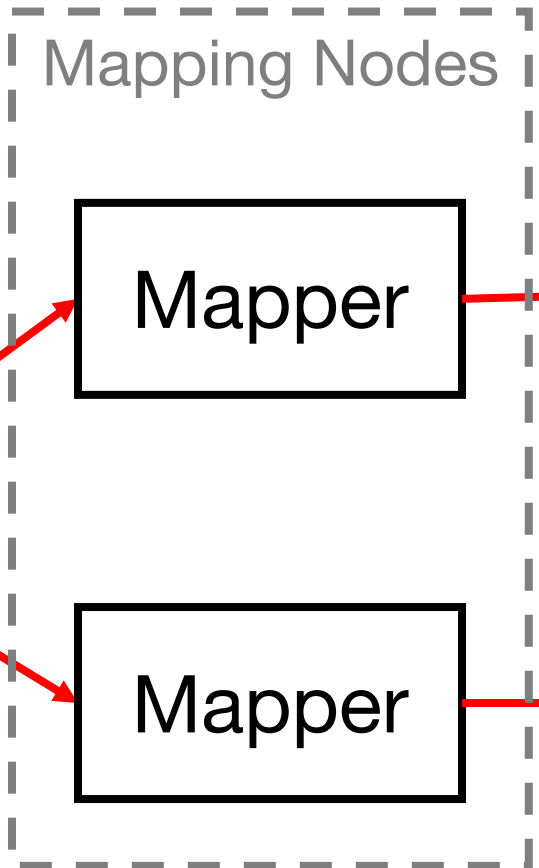
<u>GPS Coordinates</u>	<u>Site Name</u>
[22.3, 39.1]	Tim Hortons
[22.2, 39.1]	KAUST Library
[35.7, 139.7]	Starbucks
...	...
[42.0, 69.0]	Chanak Train Stop
[22.2, 39.2]	Burger King
...	...
...	...
...	...
...	...



# Split the File and Map Each Chunk Independently (2/2)

```
KEY <grid>: VALUE <locations and name>
...
```

GPS Coordinates	Site Name
[22.3, 39.1]	Tim Hortons
[22.2, 39.1]	KAUST Library
[35.7, 139.7]	Starbucks
...	...
[42.0, 69.0]	Chanak Train
[22.2, 39.2]	Burger King
...	...
...	...
...	...
...	...



```
(1,2): [22.3, 39.1] Tim Hortons
(1,2): [22.2, 39.1] KAUST Library
(1,2): ...
(2,4): [35.7, 139.7] Starbucks
(2,4): ...
```

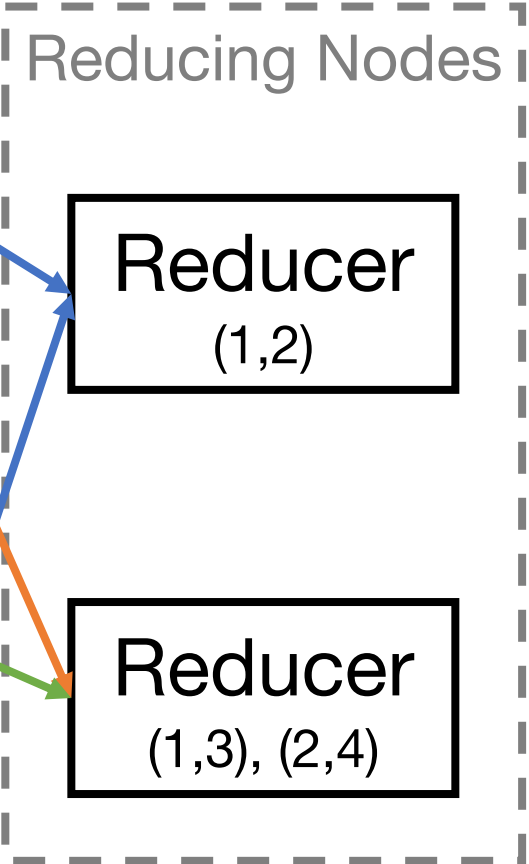
```
(1,3): [42.0, 69.0] Chanak Train
(1,3): ...
(1,2): [22.2, 39.2] Burger King
(1,2): ...
```

(KEY) can appear in multiple mappers

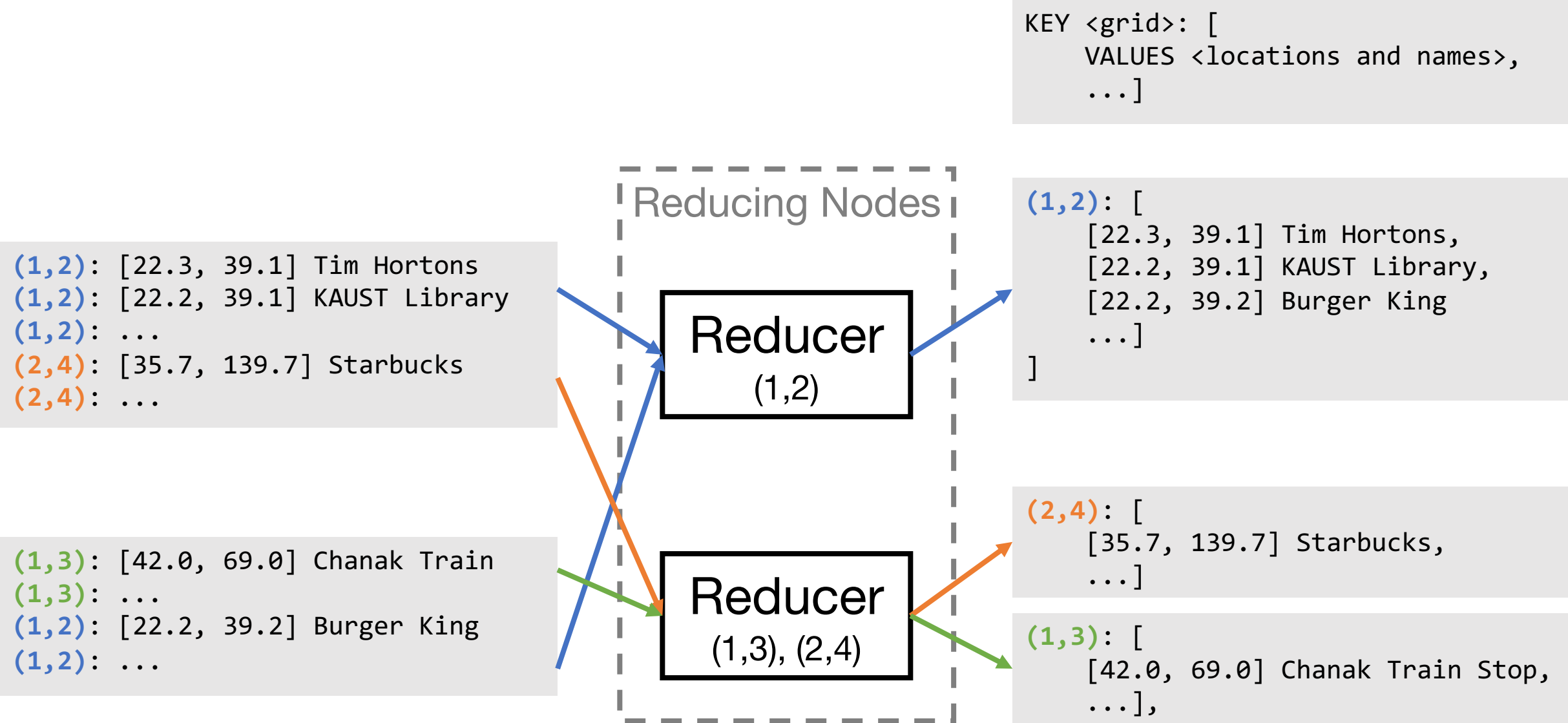
# Collect the Mapper Results and Reduce to Single Files (1/2)

(1,2): [22.3, 39.1] Tim Hortons  
(1,2): [22.2, 39.1] KAUST Library  
(1,2): ...  
(2,4): [35.7, 139.7] Starbucks  
(2,4): ...

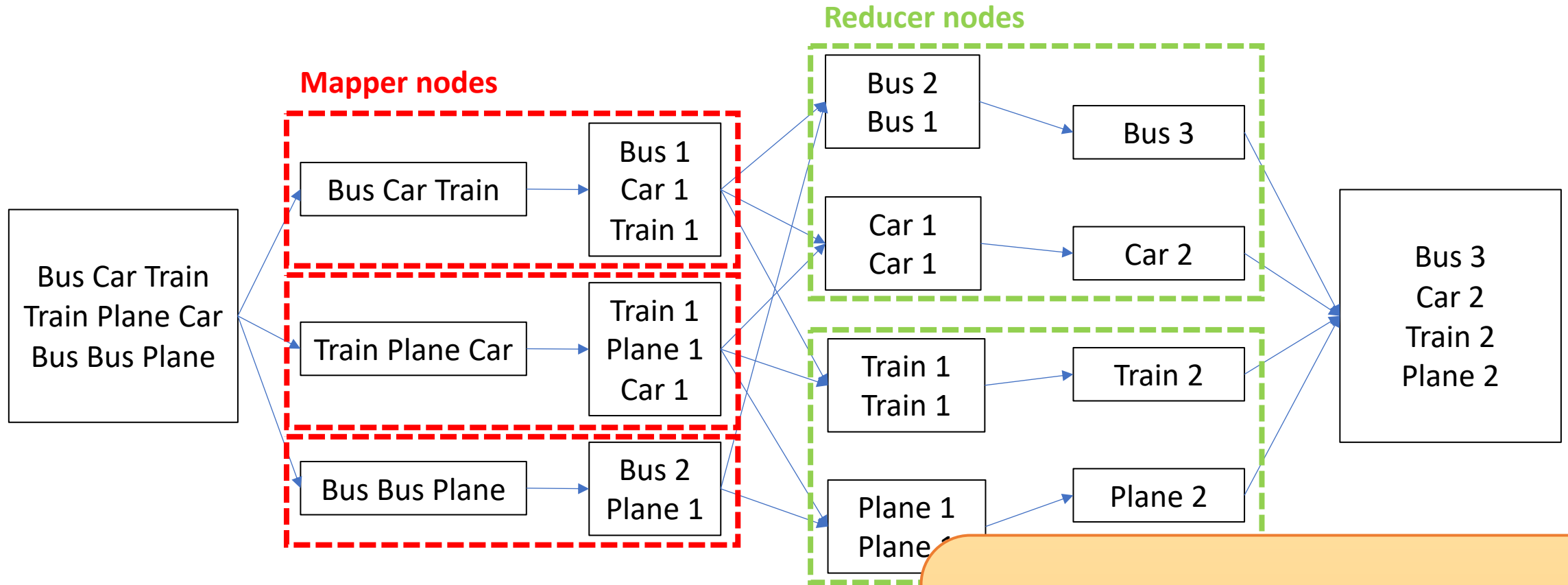
(1,3): [42.0, 69.0] Chanak Train  
(1,3): ...  
(1,2): [22.2, 39.2] Burger King  
(1,2): ...



# Collect the Mapper Results and Reduce to Single Files (2/2)



# Word Count – The *Hello World* of Map Reduce



Splitting

Mapping

Intermediate  
Splitting

**Task is automatically  
distributed across five  
different nodes**

# Hadoop: An open-source implementation



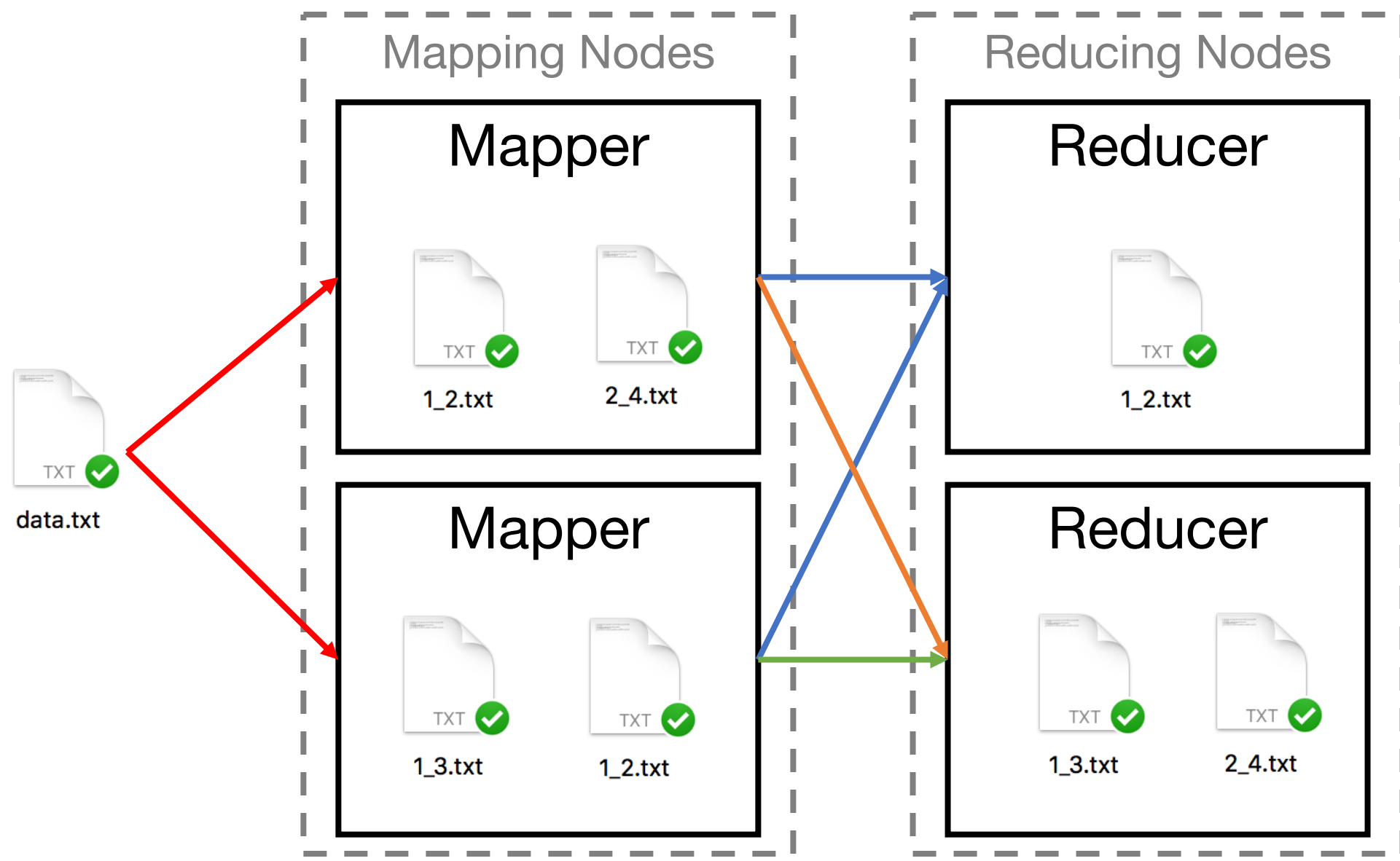
Apache Hadoop is the most popular open-source implementation of MapReduce

Runs on top of a distributed filesystem (HDFS)

Try their MapReduce tutorial:

[https://hadoop.apache.org/docs/r1.2.1/mapred\\_tutorial.html](https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html)

# How Hadoop Does it





# Concurrency in Go

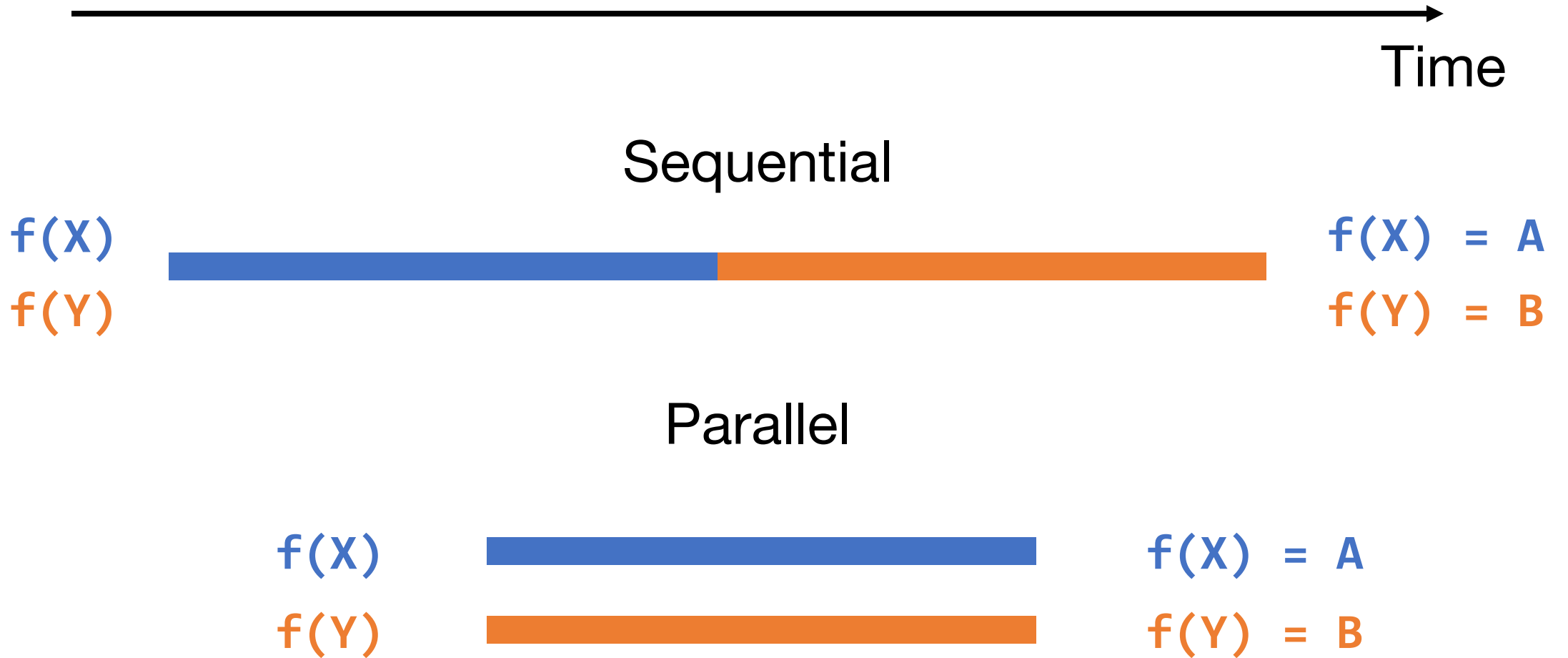
## Part 2 - Concurrency

CS 240 – Fall 2019  
Rec. 2

# What is Concurrency?

It's like parallel that's not in parallel

# What is Parallelism?



# What is Concurrency?



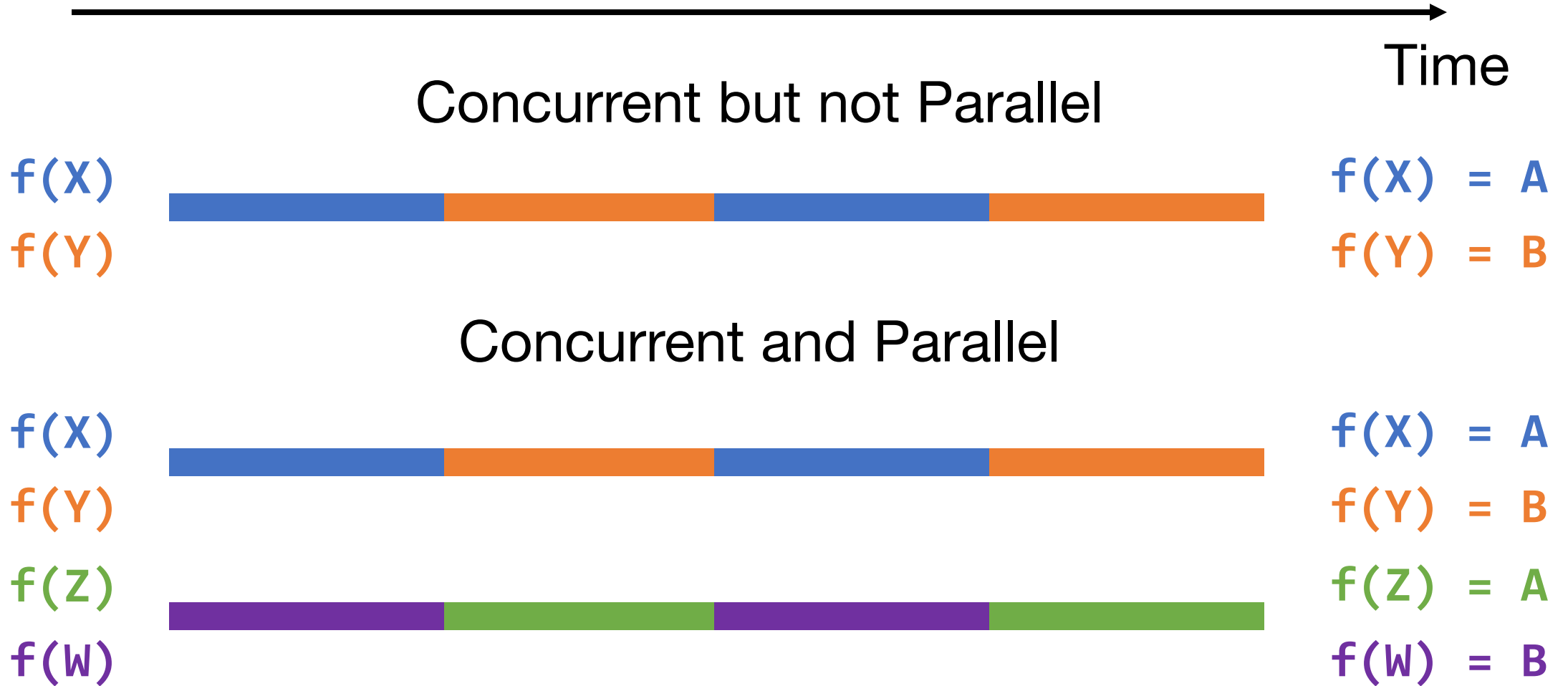
## Sequential



## Concurrent



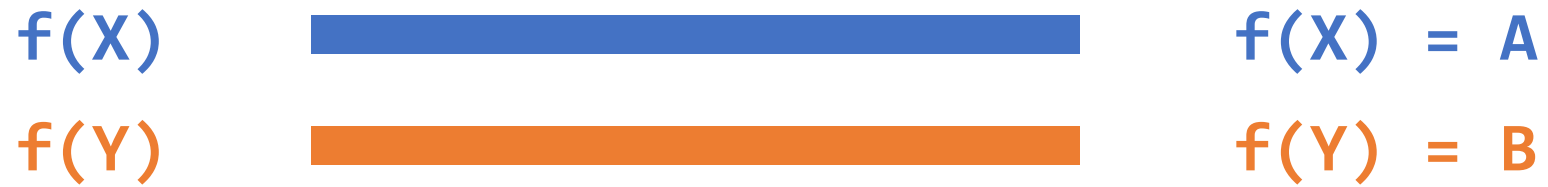
# Concurrency Could be Parallel but not Always



# Parallel is Always Concurrent



Parallel but not Concurrent?

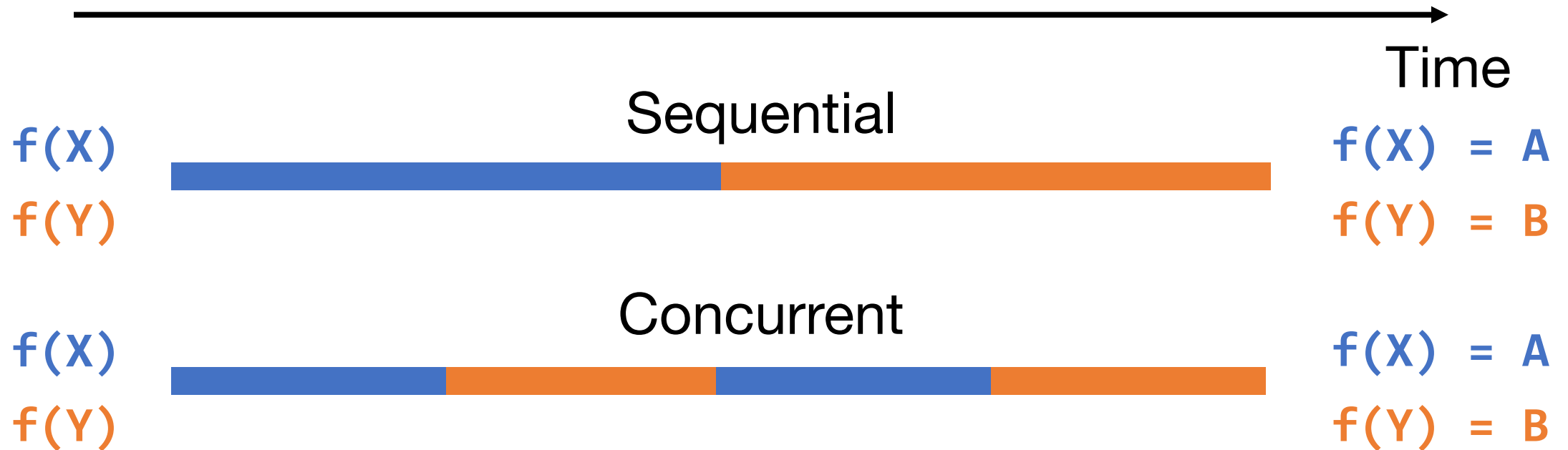


Nope ... still concurrent

Parallel	→	Concurrent
Concurrent	↗	Parallel

# Why Care about Concurrency

If something concurrent but not parallel takes as much time as something sequential, why make it concurrent?



# Concurrency is a *Design* Pattern

“Concurrency is about dealing with lots of things at once.  
Parallelism is about doing lots of things at once.”

- Rob Pike

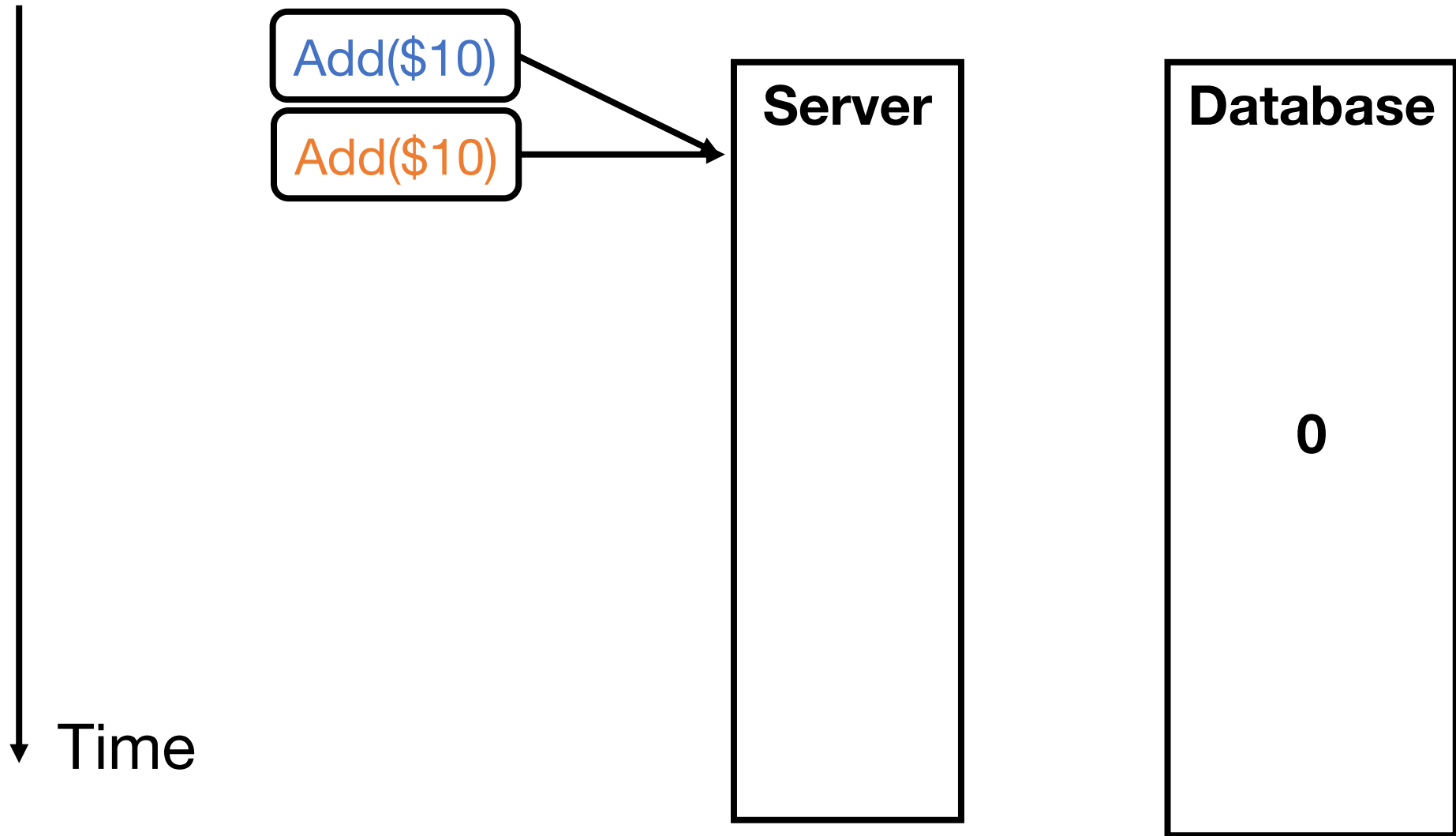


# Distributed Systems are Unpredictable

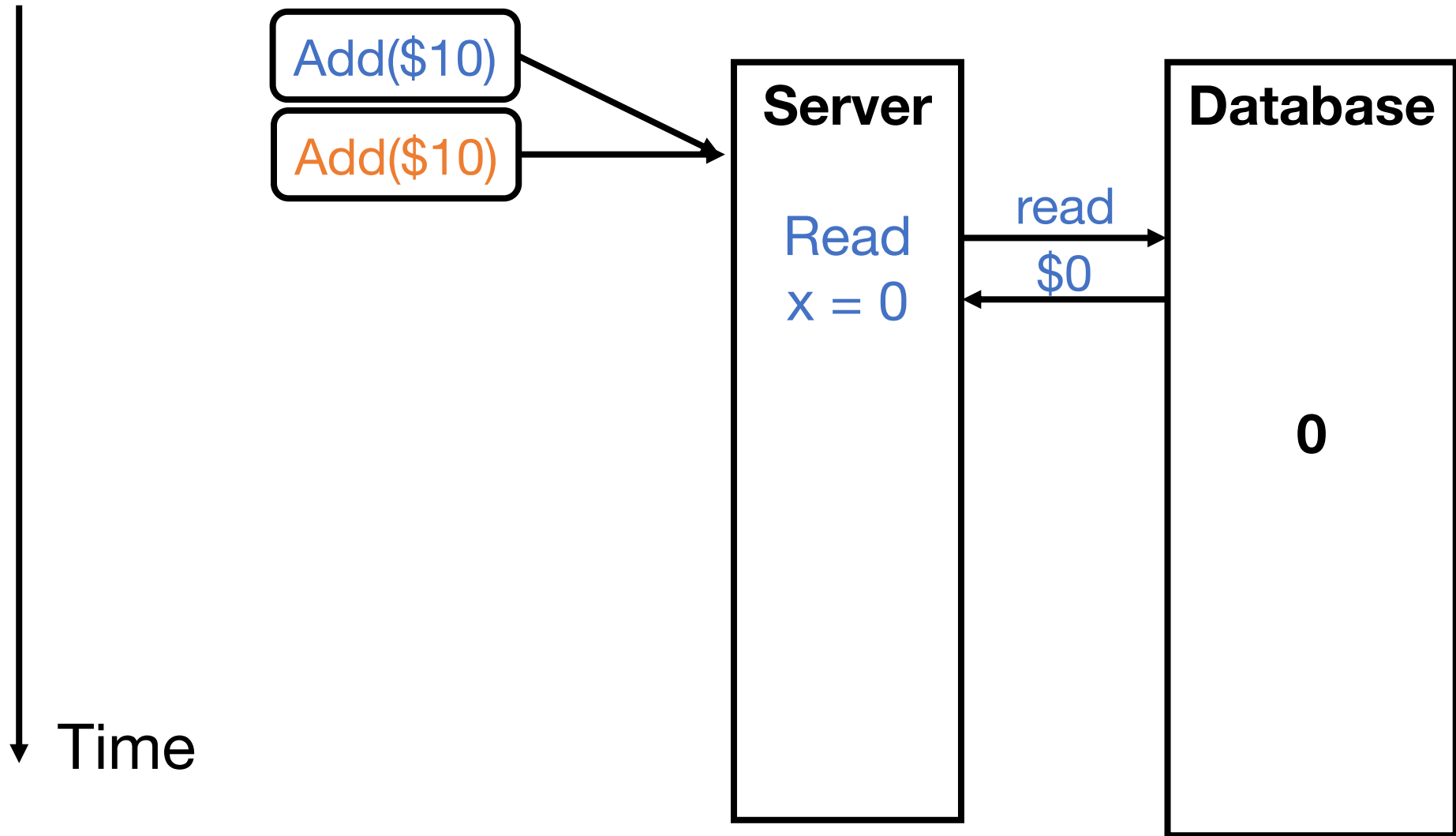
Servers need to react to:

- Others servers
  - Crashes
  - Users
  - ...

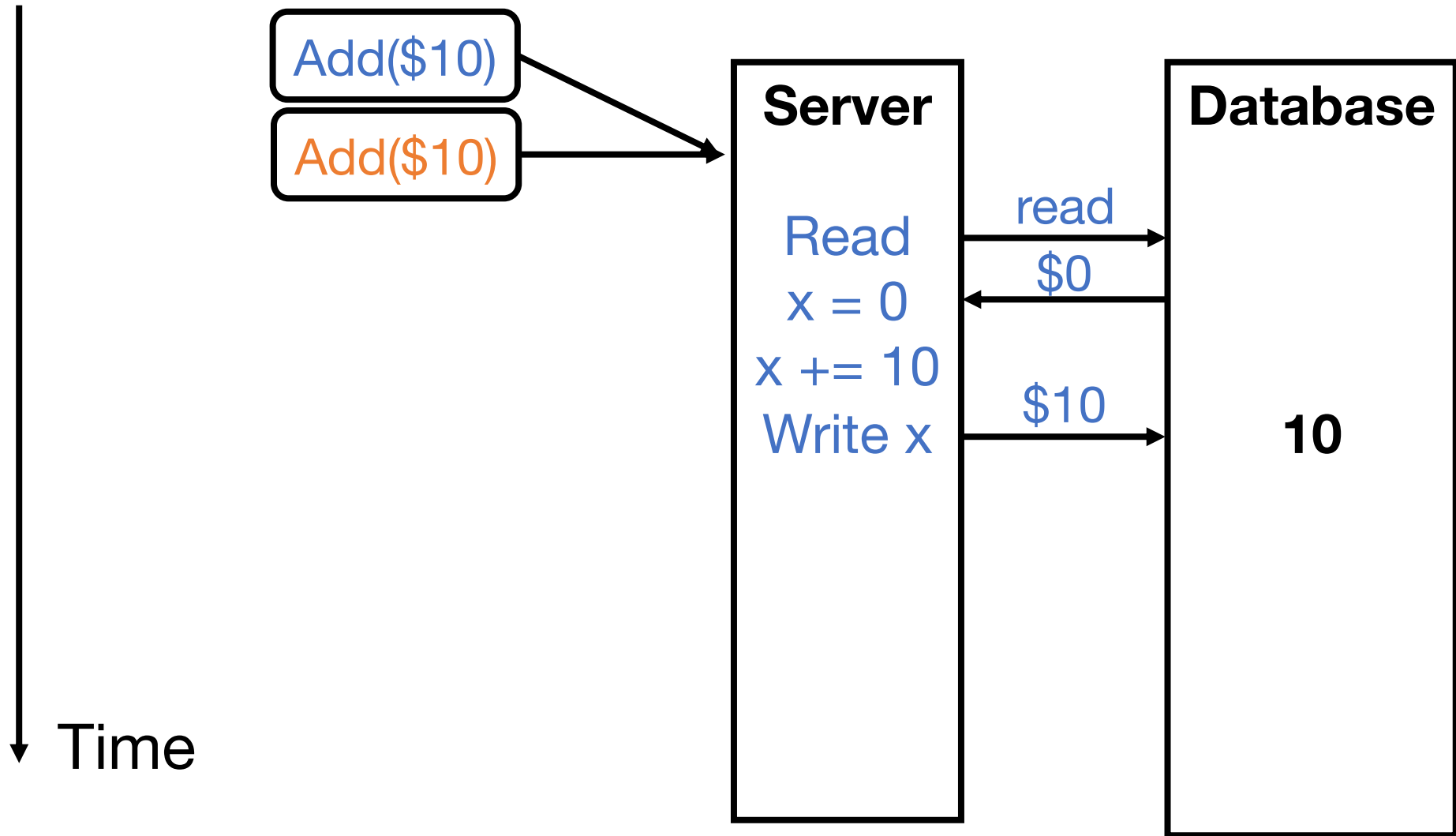
# Making Bank Deposits Concurrent (1/5)



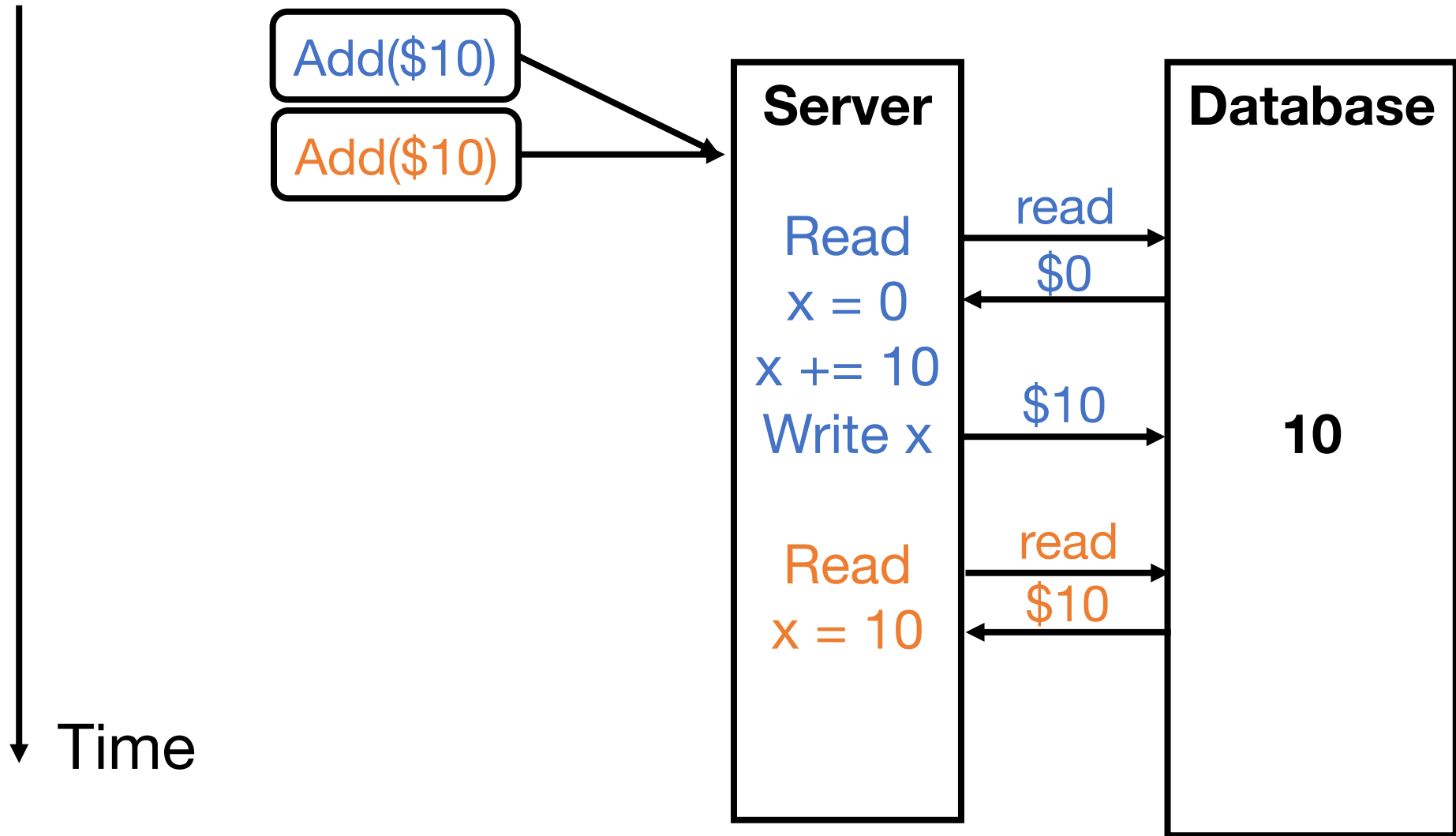
# Making Bank Deposits Concurrent (2/5)



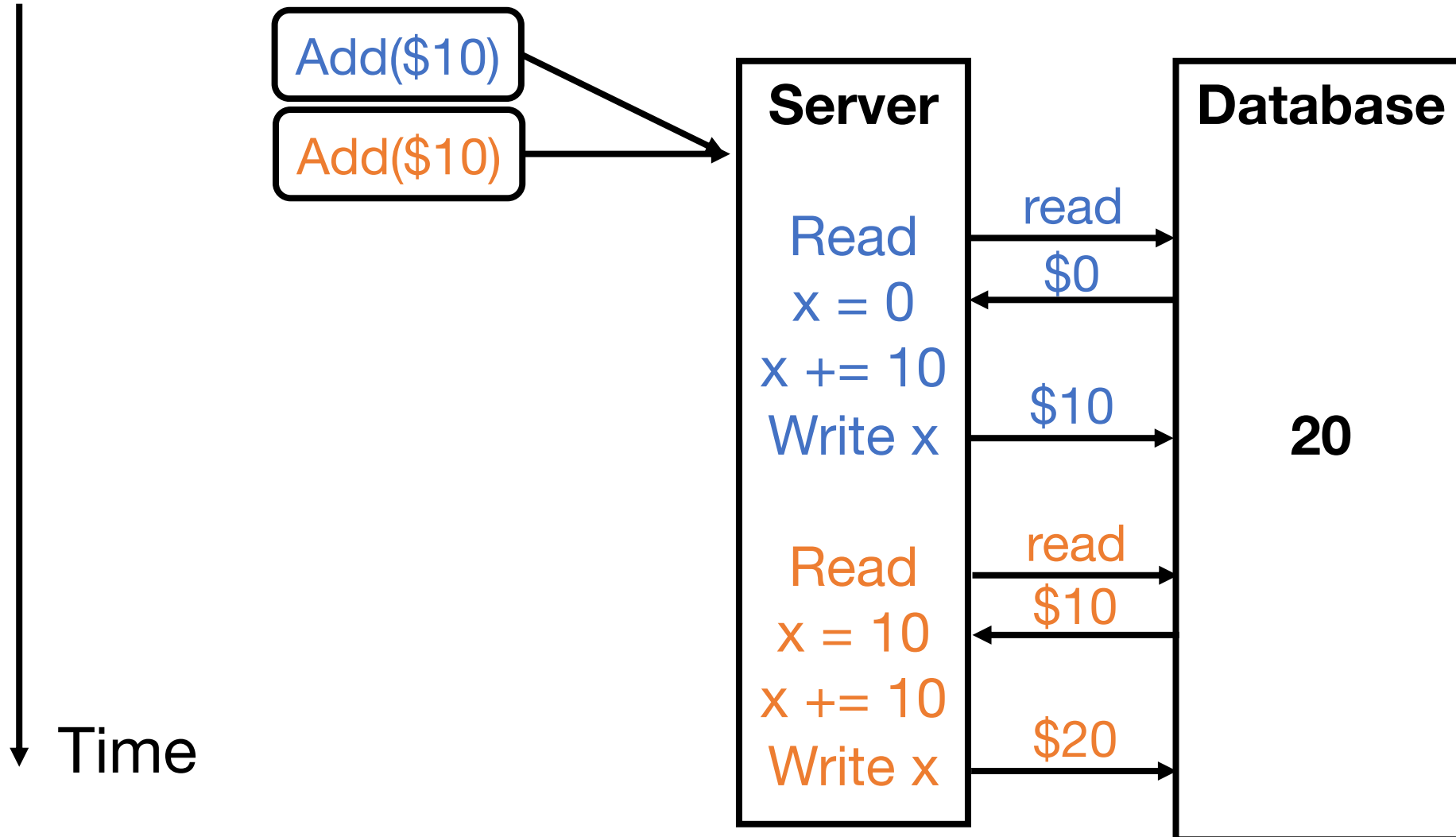
# Making Bank Deposits Concurrent (3/5)



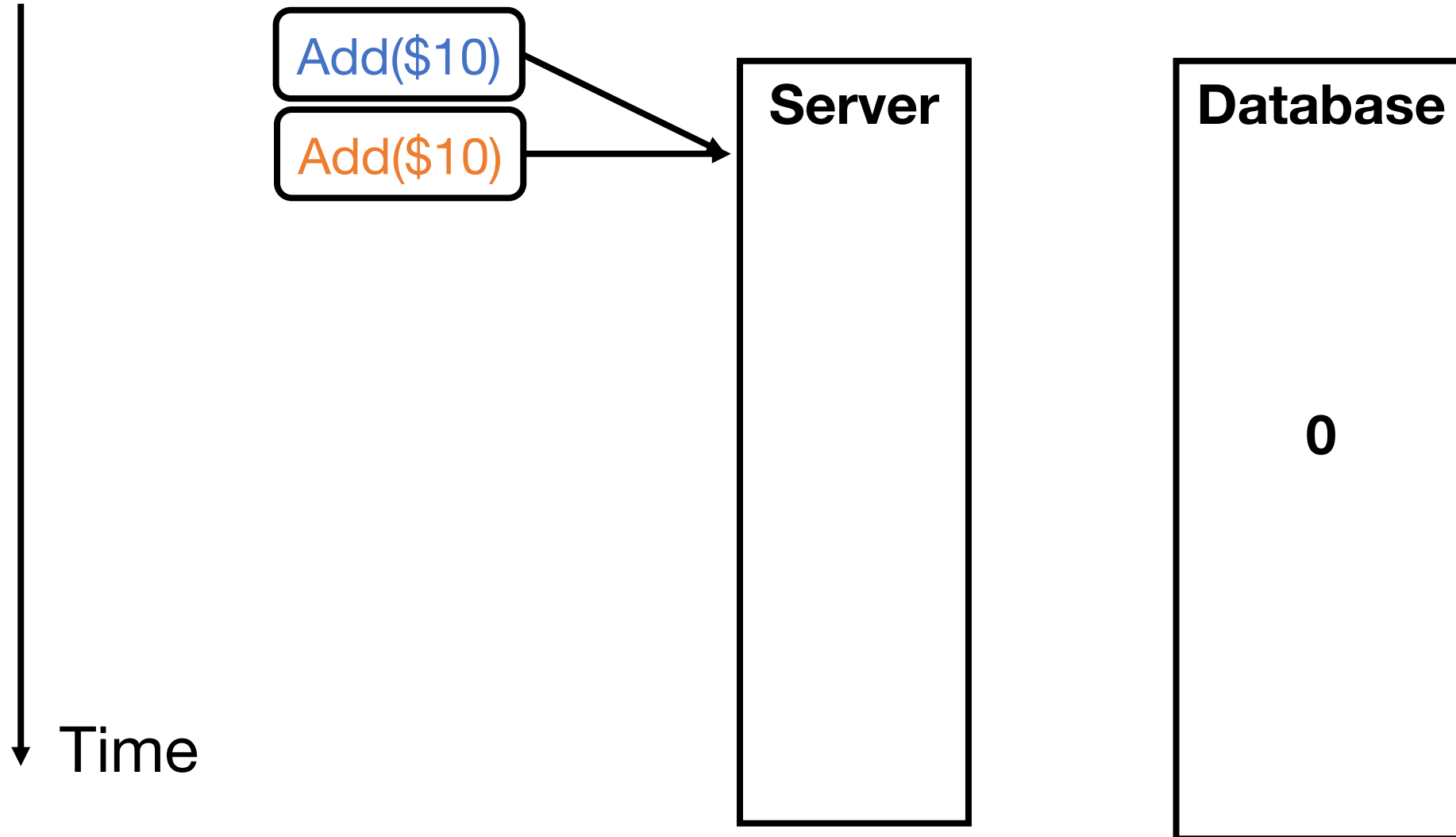
# Making Bank Deposits Concurrent (4/5)



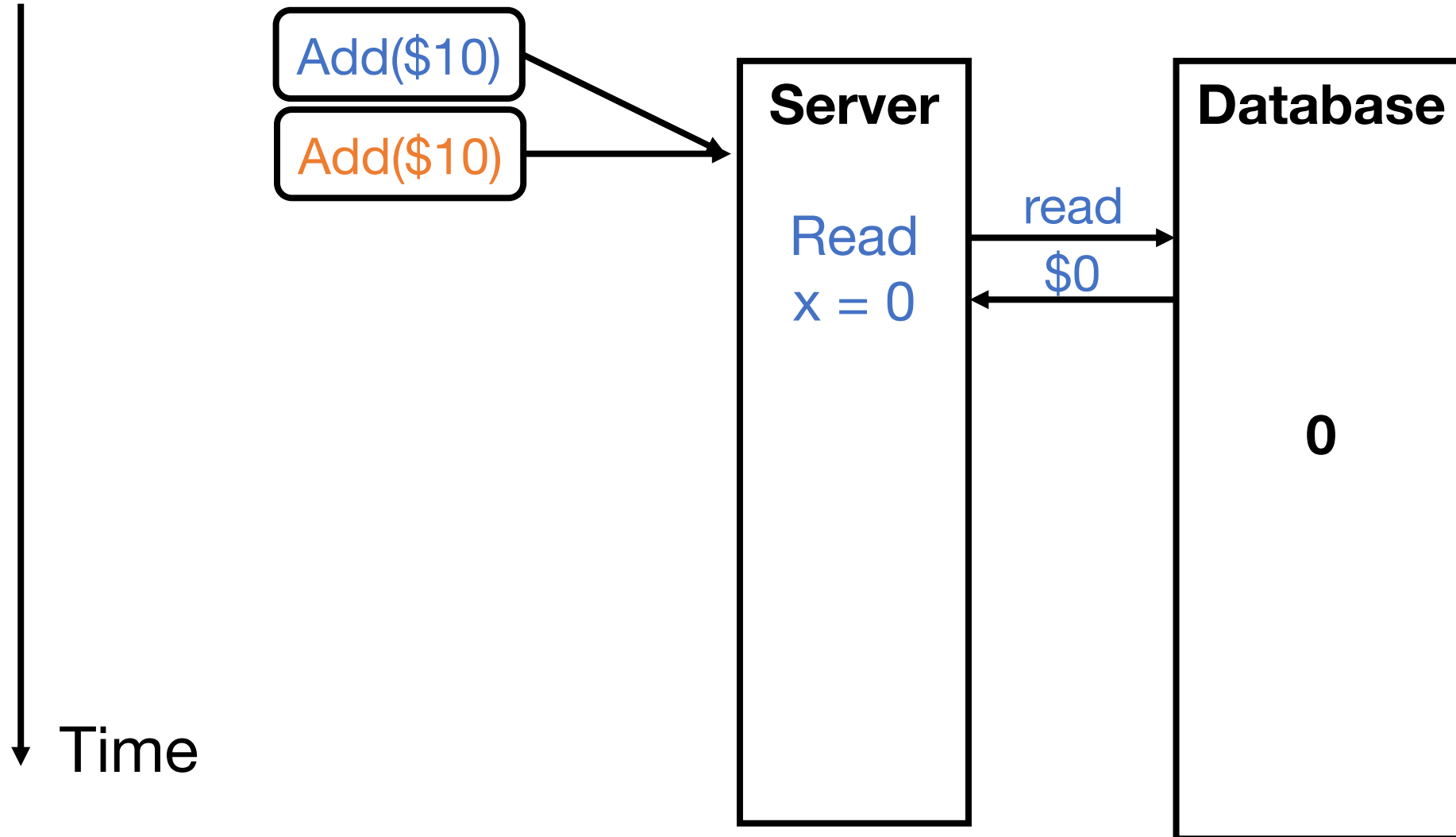
# Making Bank Deposits Concurrent (5/5)



# Concurrent Bank Deposits! Yay? (1/5)

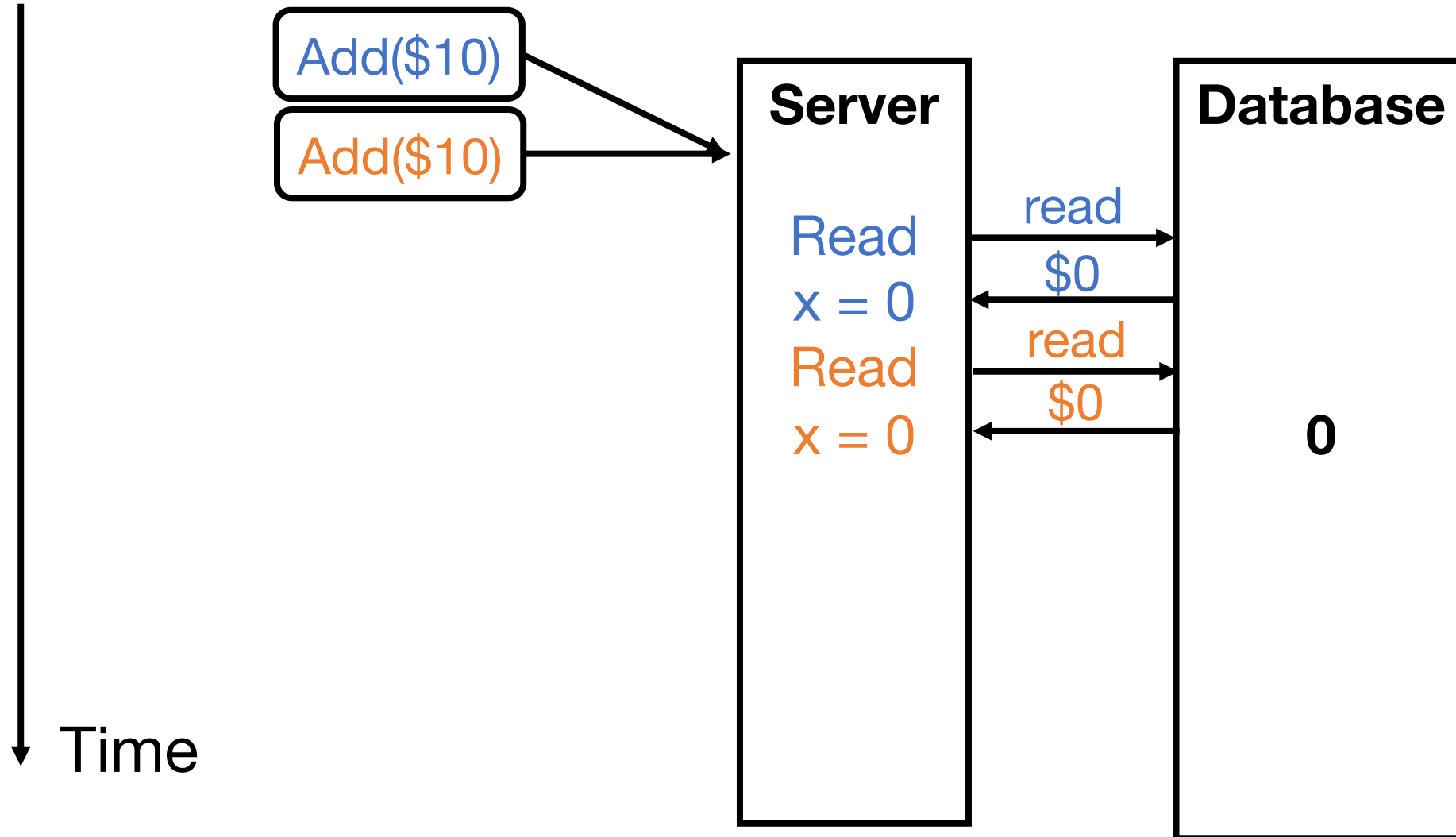


# Concurrent Bank Deposits! Yay? (2/5)

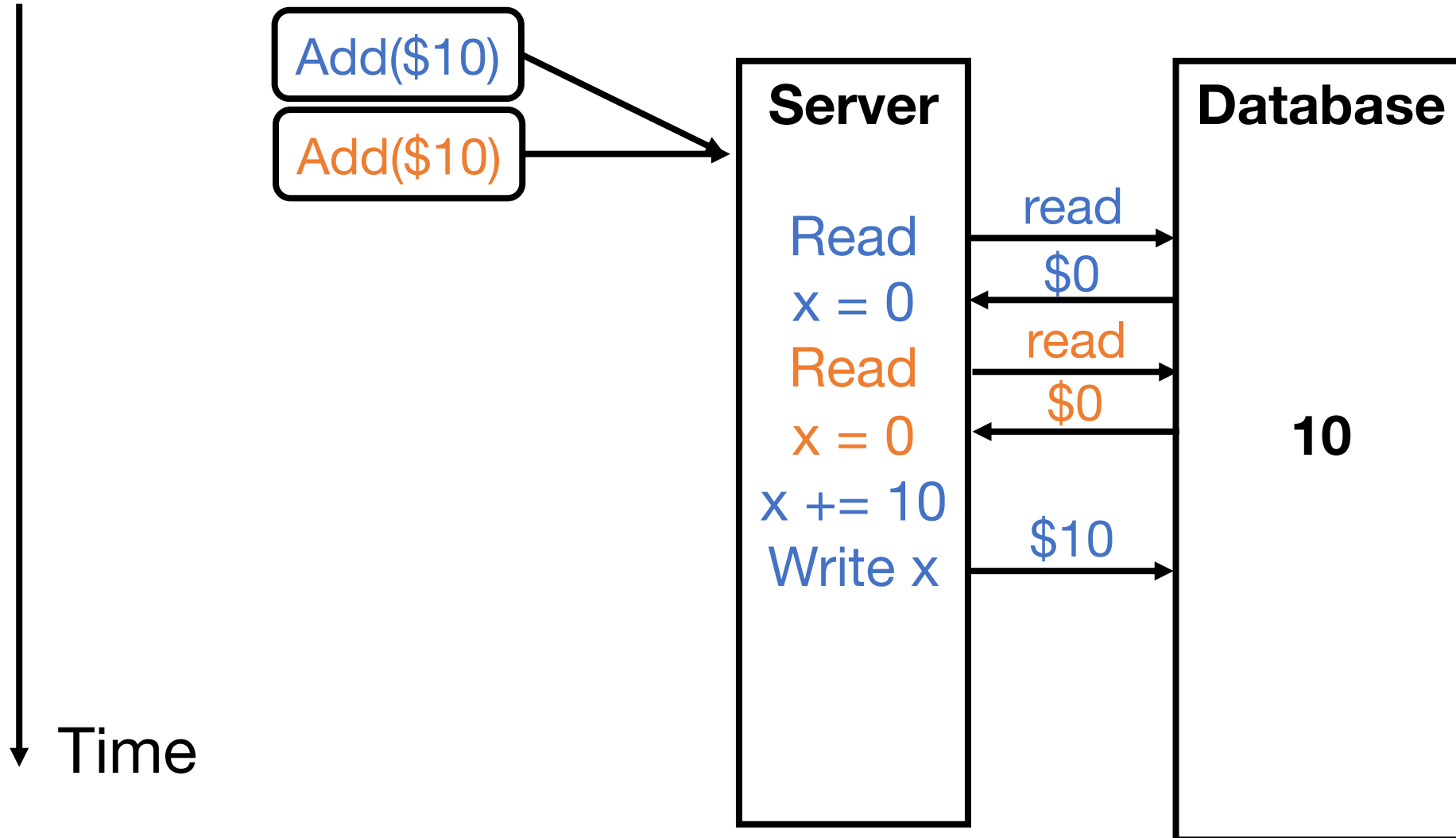




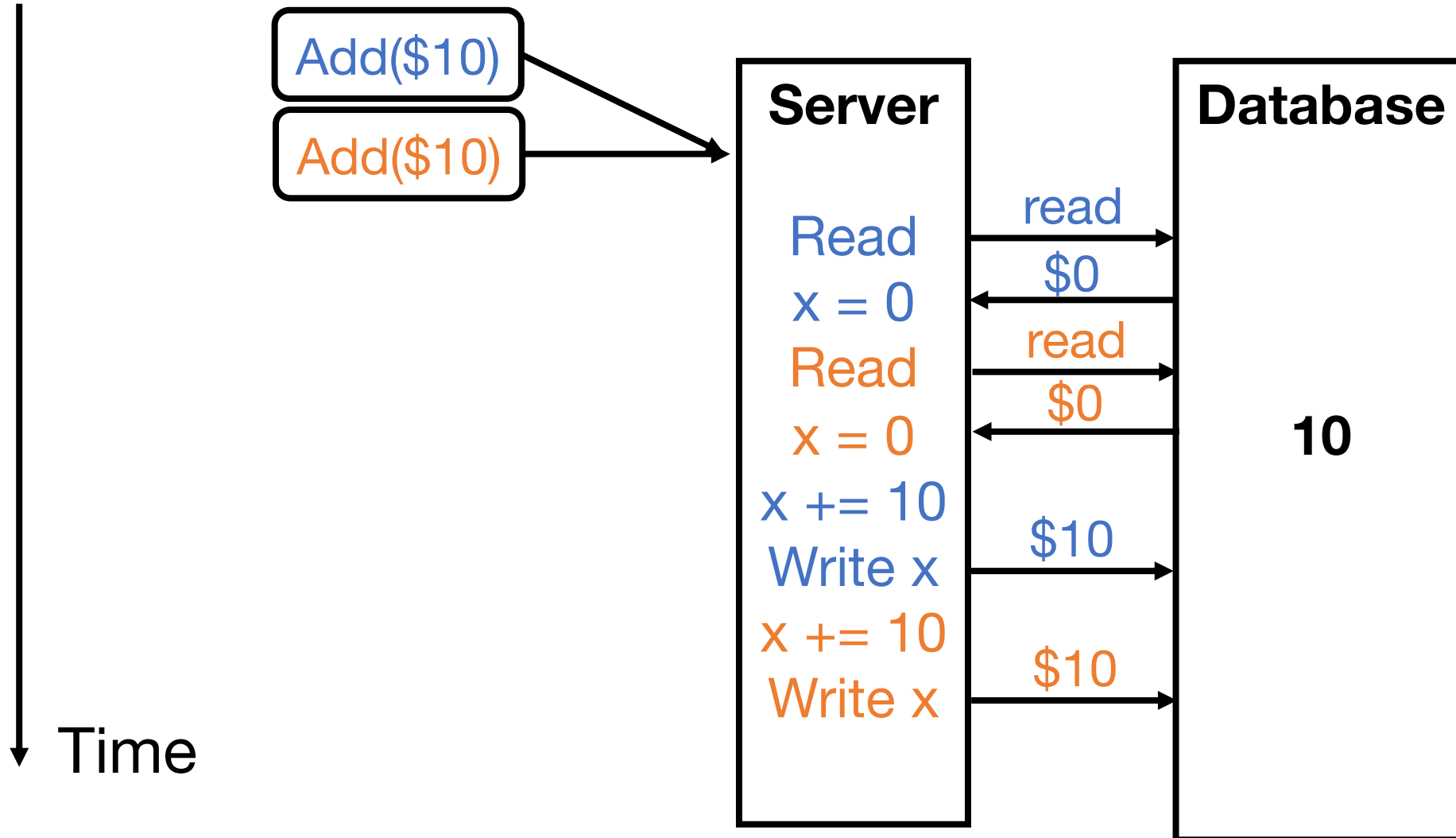
# Concurrent Bank Deposits! Yay? (3/5)



# Concurrent Bank Deposits! Yay? (4/5)



# Concurrent Bank Deposits! Yay? (5/5)



# Concurrency Needs to be Synchronized

**Locks** – limit access using shared memory  
**Channels** – pass information using a queue

# Visualize Everything We've Learned

And also see many different methods of  
achieving synchronization:

[http://divan.github.io/posts/go\\_concurrency\\_visualize/](http://divan.github.io/posts/go_concurrency_visualize/)