Concurrency in Go

CS 240 – Fall 2018
Rec. 2

Housekeeping

- Should have a working doMap() in Assignment 1

We Should Probably Teach you Map Reduce

The *Hello World* of Map Reduce:
Word Count

If we have time:
Let’s Make, a very basic, Google Maps from Raw Data

(A Solution to the Final Project for CS 245 – Databases)
You’re welcome

Abstract Map Reduce

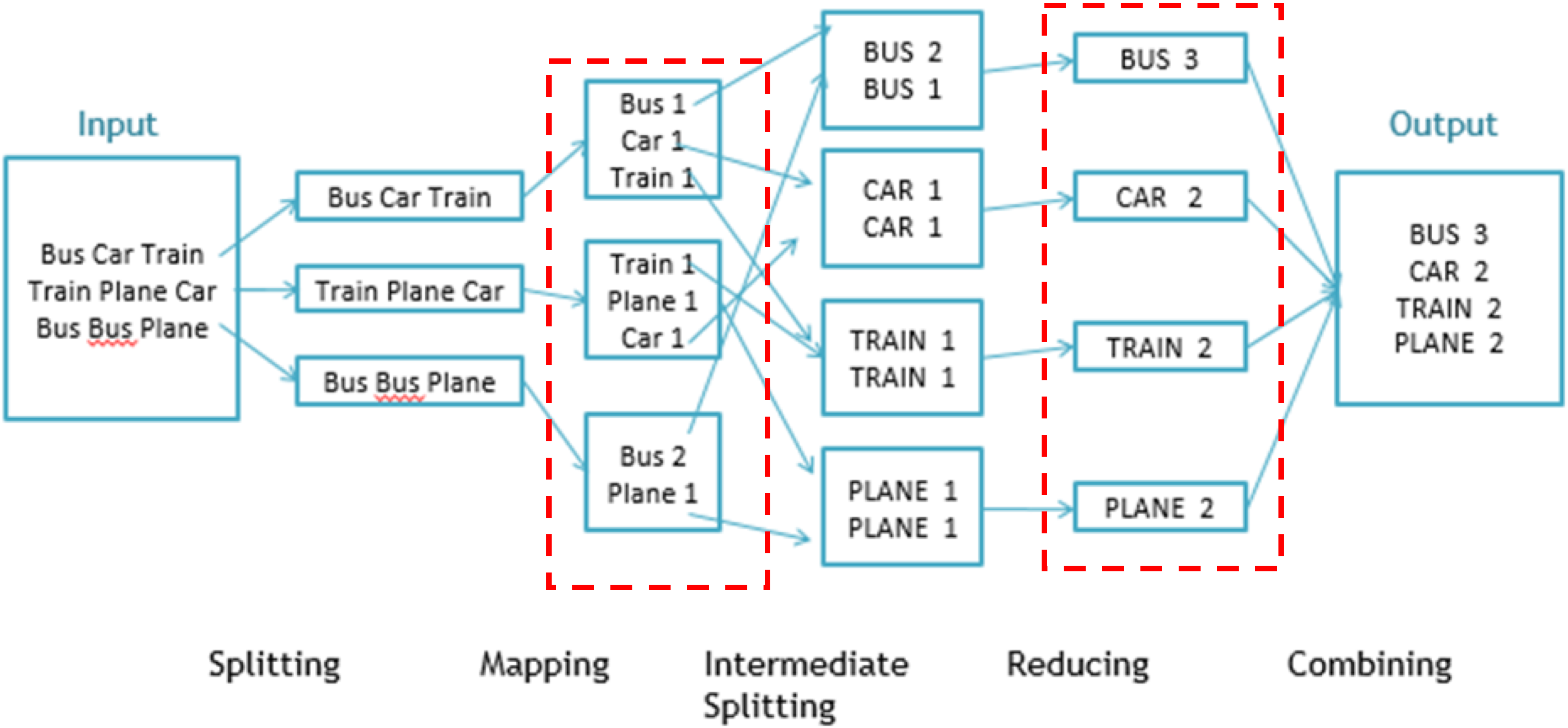
```
map(key, value) -> list(<k’, v’>)
```

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

```
reduce(key, list<value>) -> <k’, v’>
```

- Applies aggregation function to values
- Outputs result

Word Count – The *Hello World* of Map Reduce



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
[22.2,    39.1]                 KAUST Library
[35.7,    139.7]                Starbucks
...                             ...
```

In KAUST

In Tokyo, Japan

“It’s ok, I didn’t want to enjoy my weekend anyway”

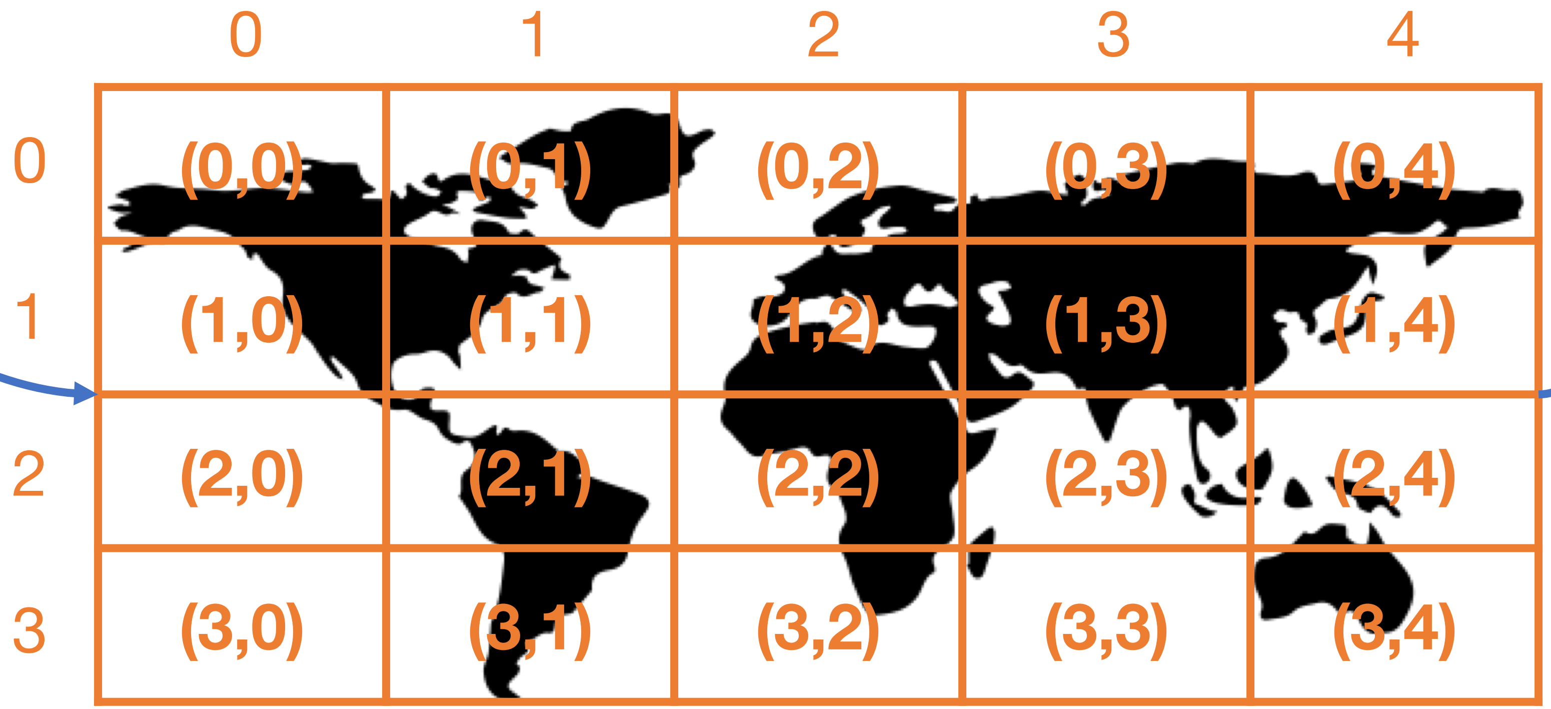
A Motivating Problem for Map Reduce

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

TXT ✓ 0_0.txt TXT ✓ 0_1.txt TXT ✓ 0_2.txt

TXT ✓ 1_0.txt TXT ✓ 1_1.txt TXT ✓ 1_2.txt

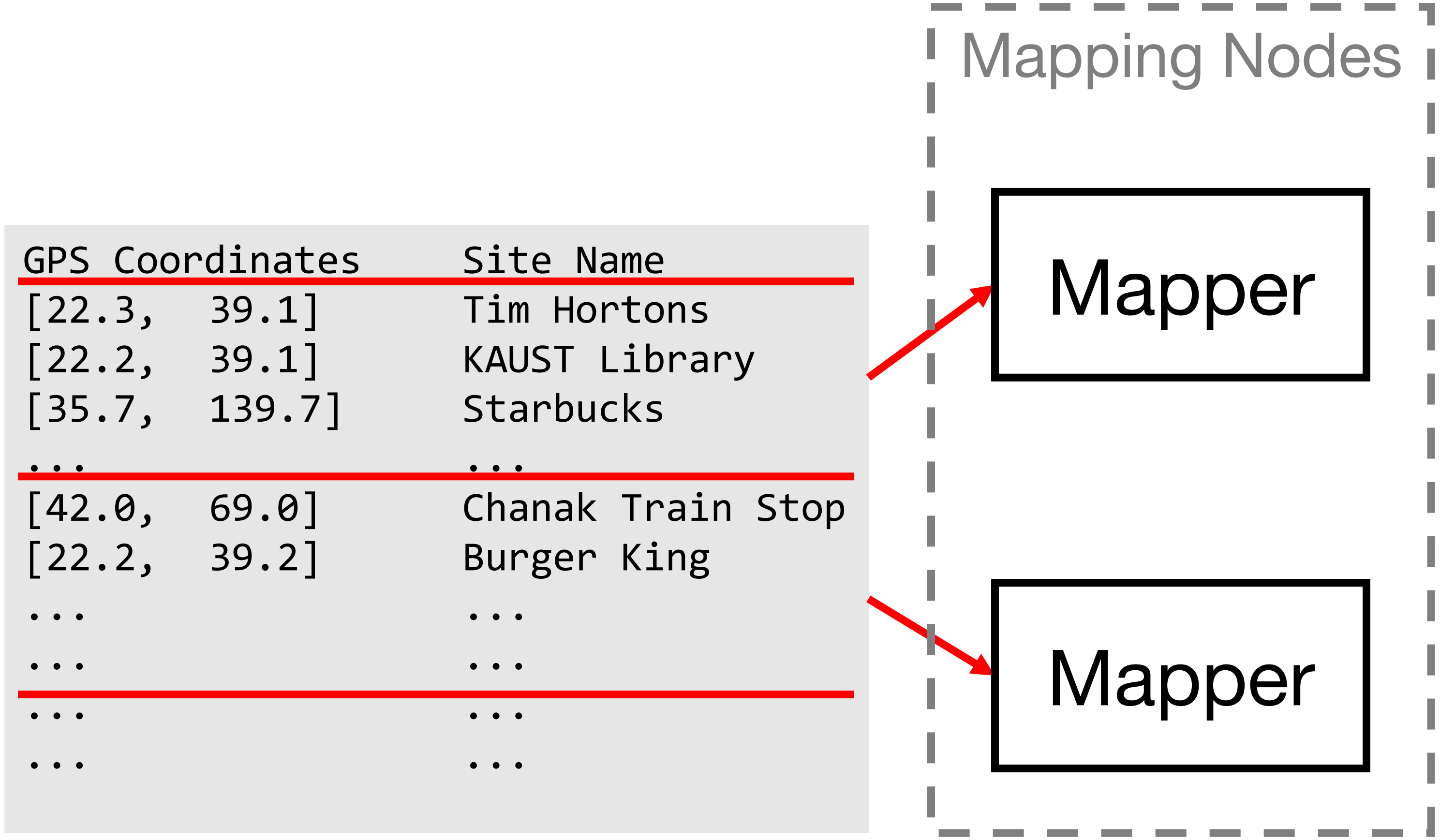
Map to grids



	0	1	2	3	4
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)

Reduce to single files

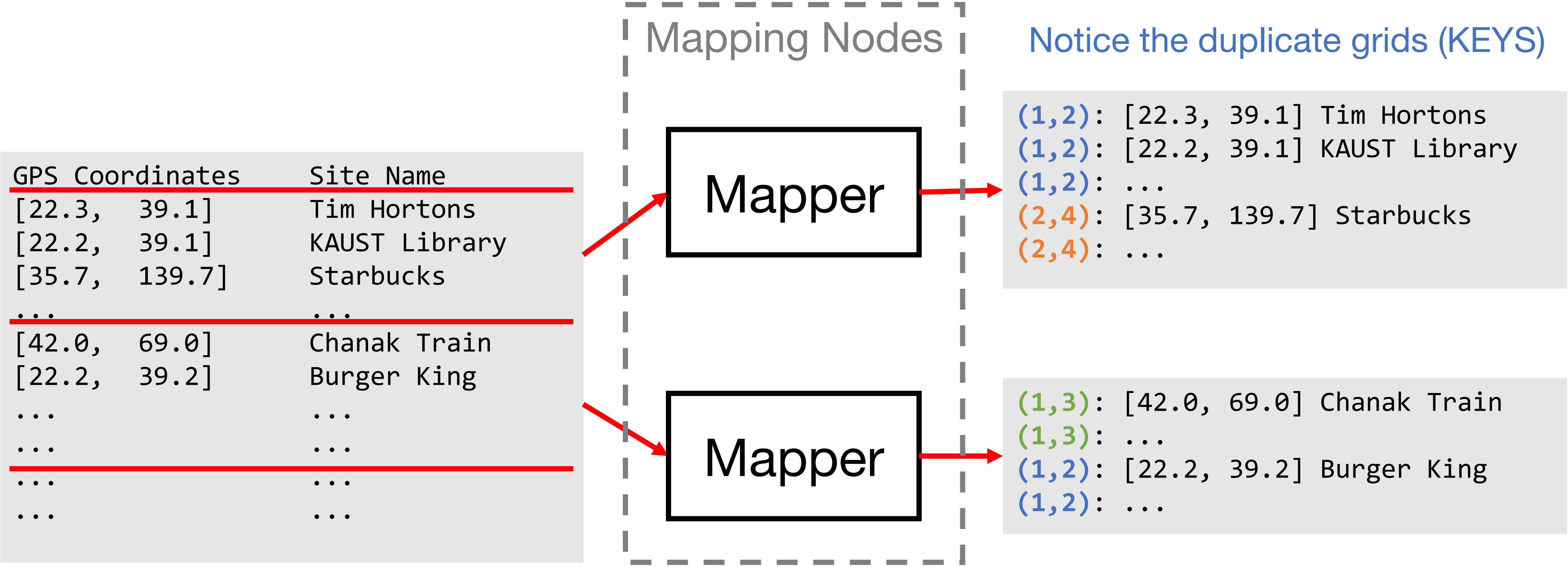
Split the File and Map Each Chunk Independently (1/2)



```
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 Stop
[22.2,   39.2]          Burger King
...                     ...
...                     ...
...                     ...
...                     ...
```

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

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



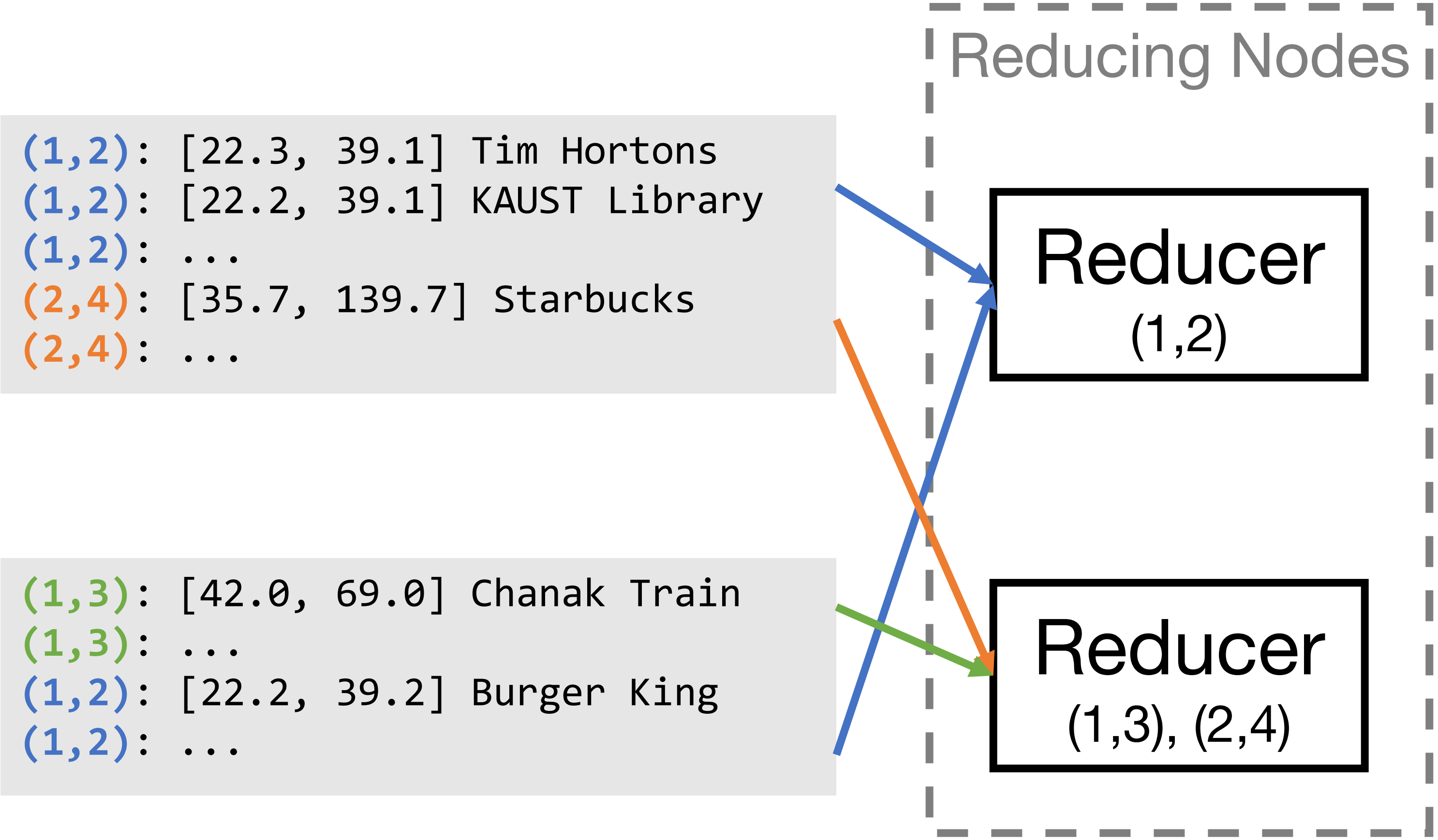
GPS Coordinates	Site Name
[22.3, 39.1]	Tim Hortons
[22.2, 39.1]	KAUST Library
[35.7, 139.7]	Starbucks
...	...
[42.0, 69.0]	Chanak Train
[22.2, 39.2]	Burger King
...	...
...	...
...	...
...	...

Notice the duplicate grids (KEYS)

```
(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): ...
```

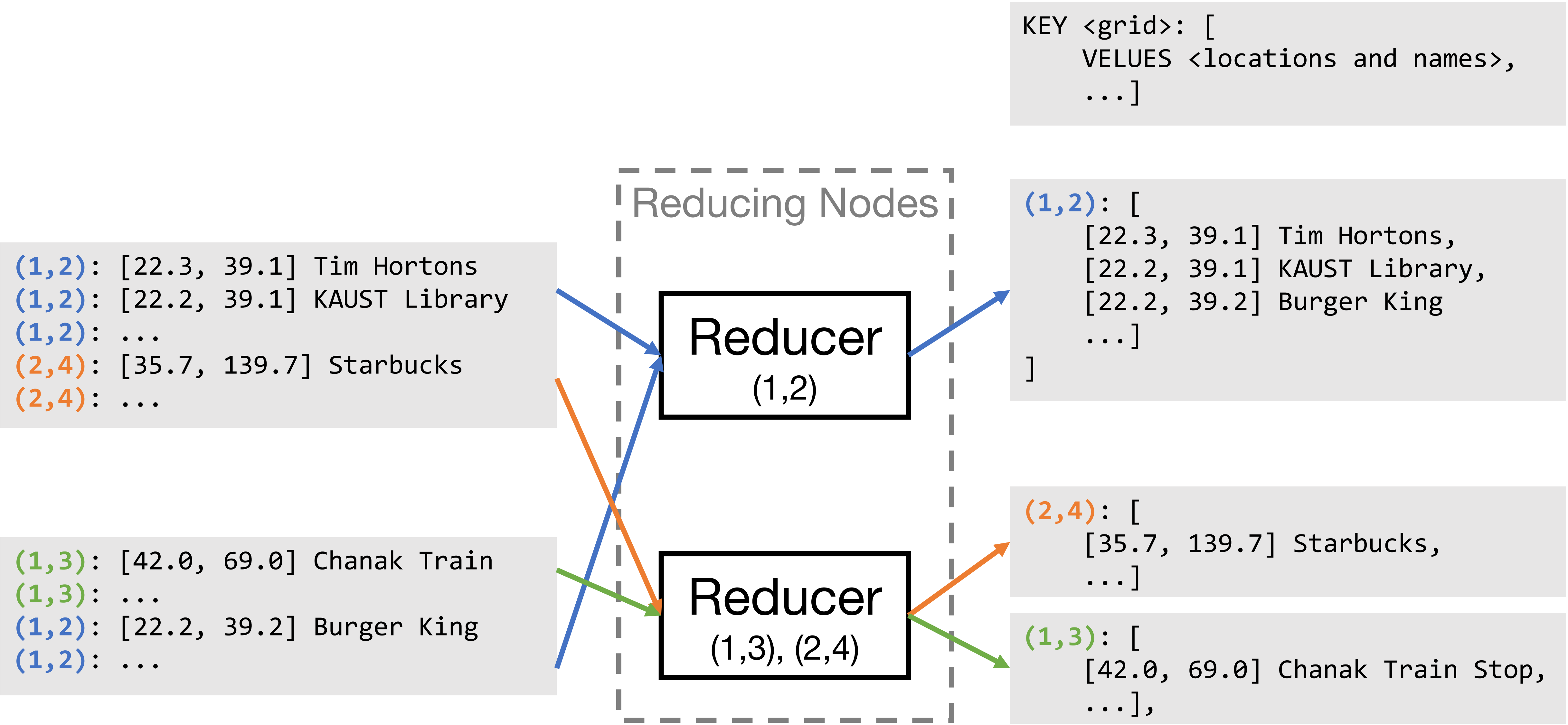
Collect the Mapper Results and Reduce to Single Files (1/2)



```
(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)



```
KEY <grid>: [
    VELUES <locations and names>,
    ...]
```

```
(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): ...
```

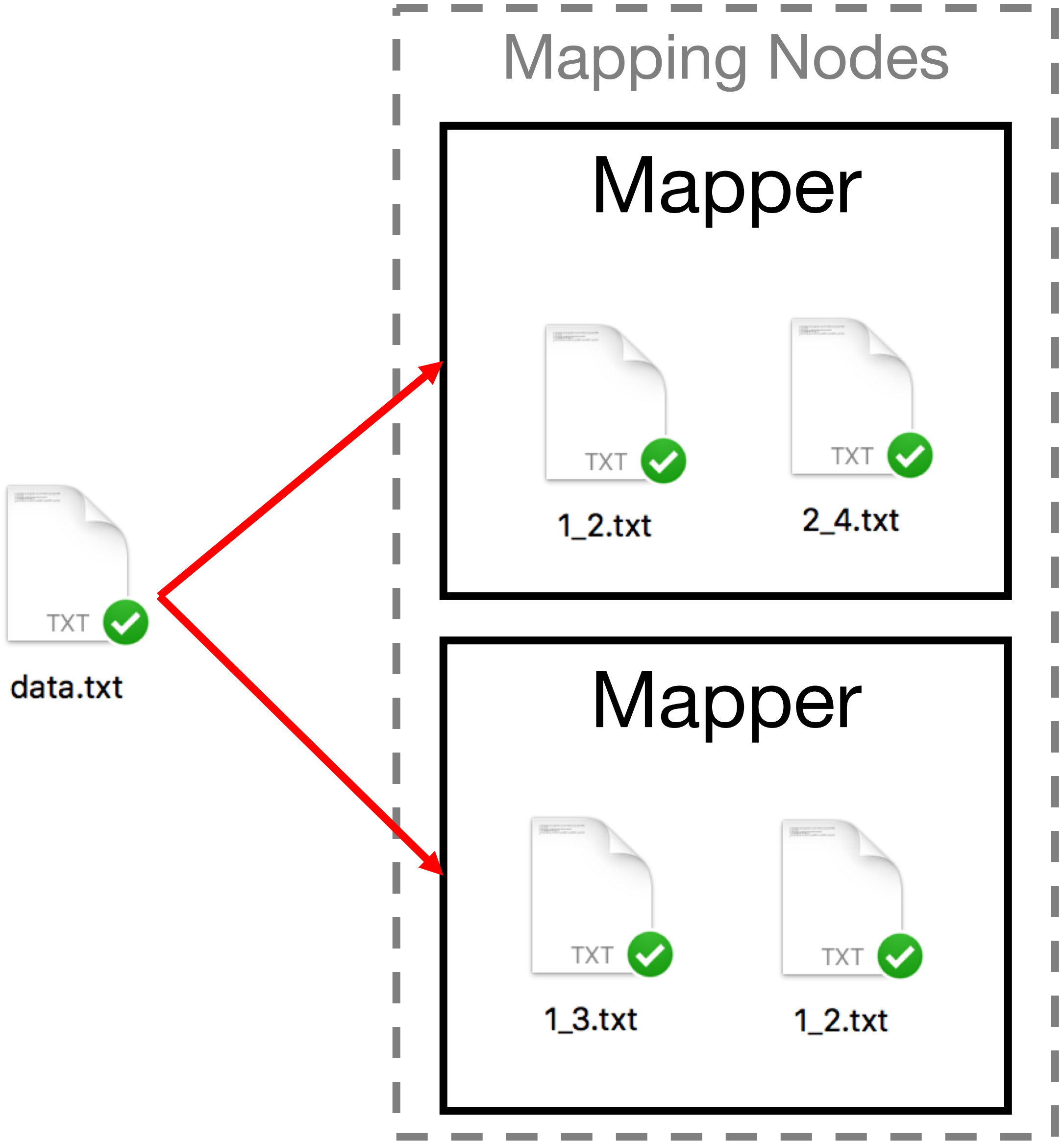
```
(1,3): [42.0, 69.0] Chanak Train
(1,3): ...
(1,2): [22.2, 39.2] Burger King
(1,2): ...
```

```
(1,2): [
    [22.3, 39.1] Tim Hortons,
    [22.2, 39.1] KAUST Library,
    [22.2, 39.2] Burger King
    ...]
]
```

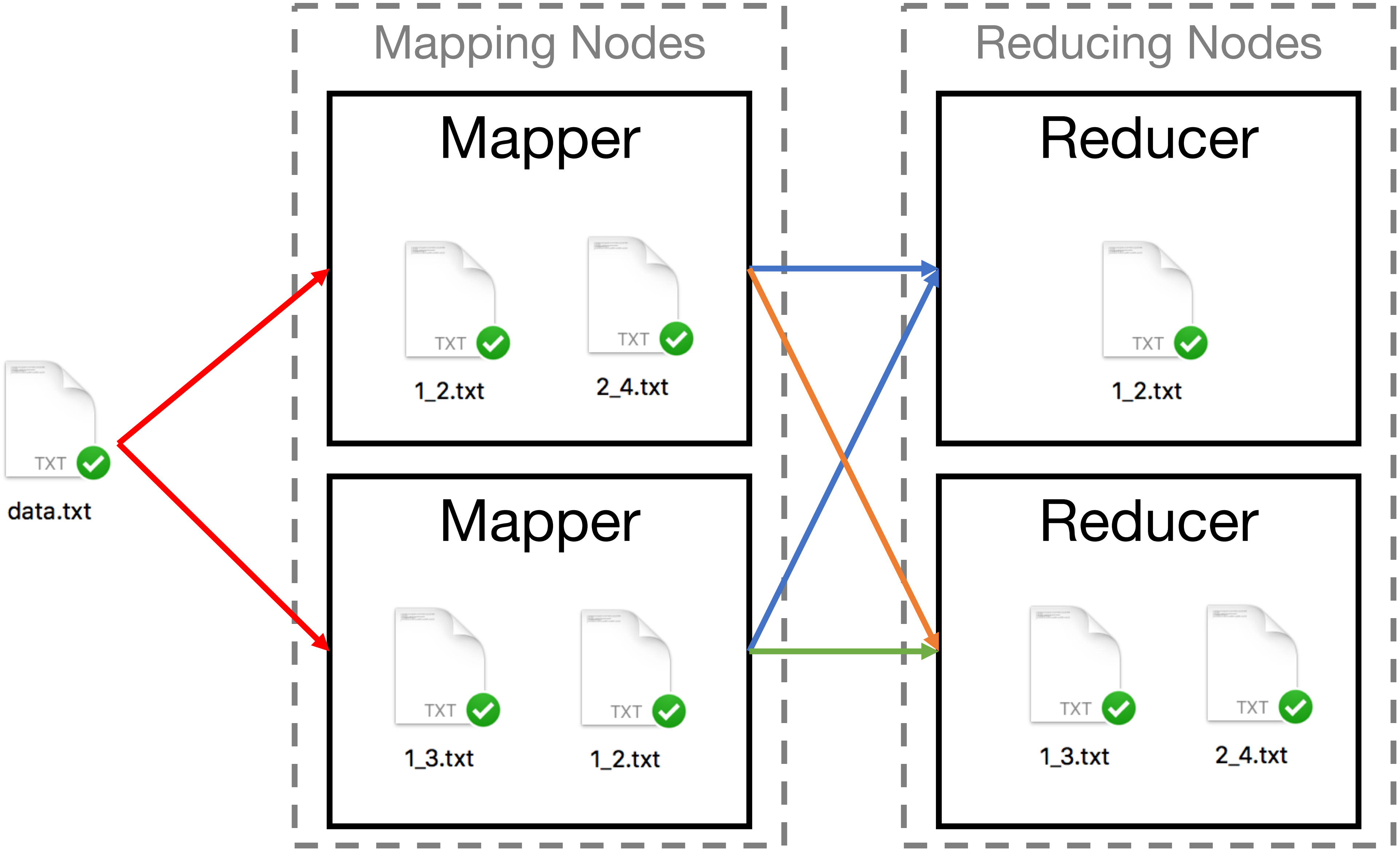
```
(2,4): [
    [35.7, 139.7] Starbucks,
    ...]
```

```
(1,3): [
    [42.0, 69.0] Chanak Train Stop,
    ...],
```

How Hadoop Does it (1/2)



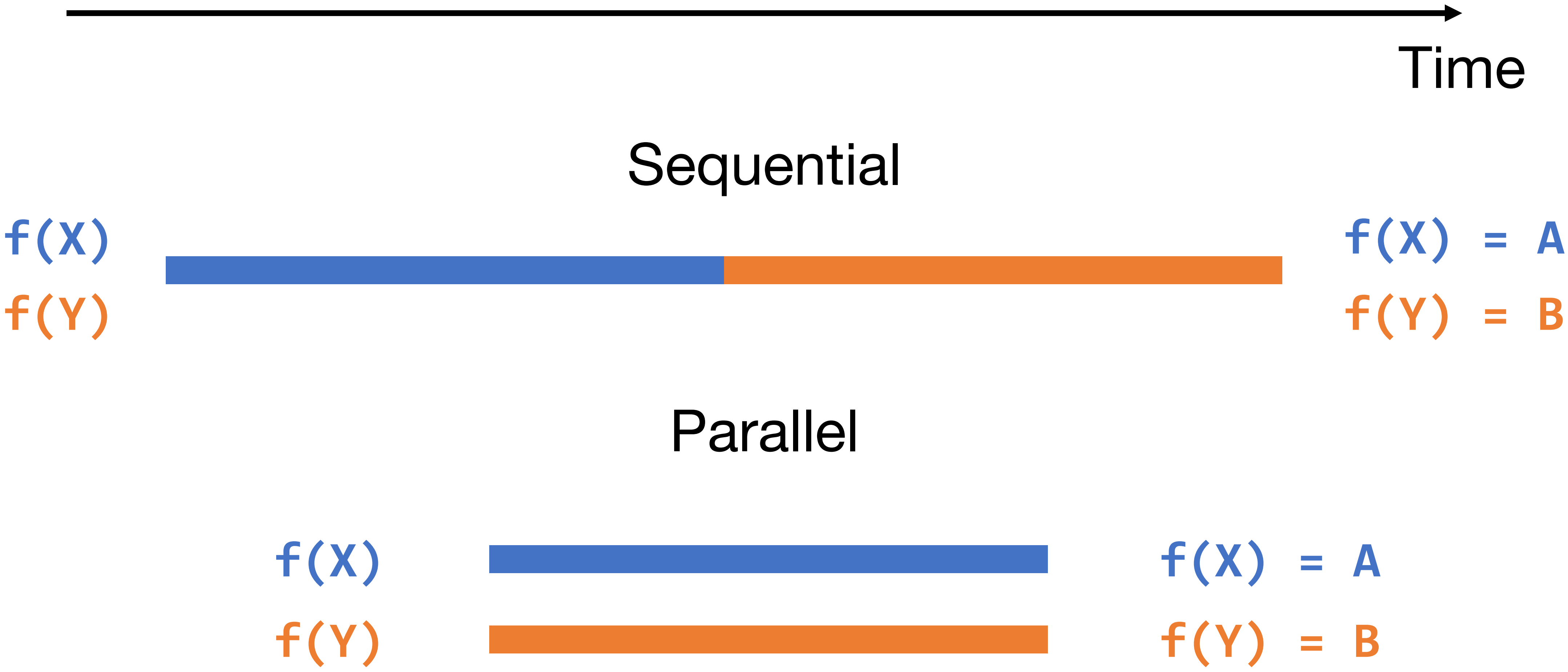
How Hadoop Does it (2/2)



What is Concurrency?

It’s like parallel that’s not in parallel

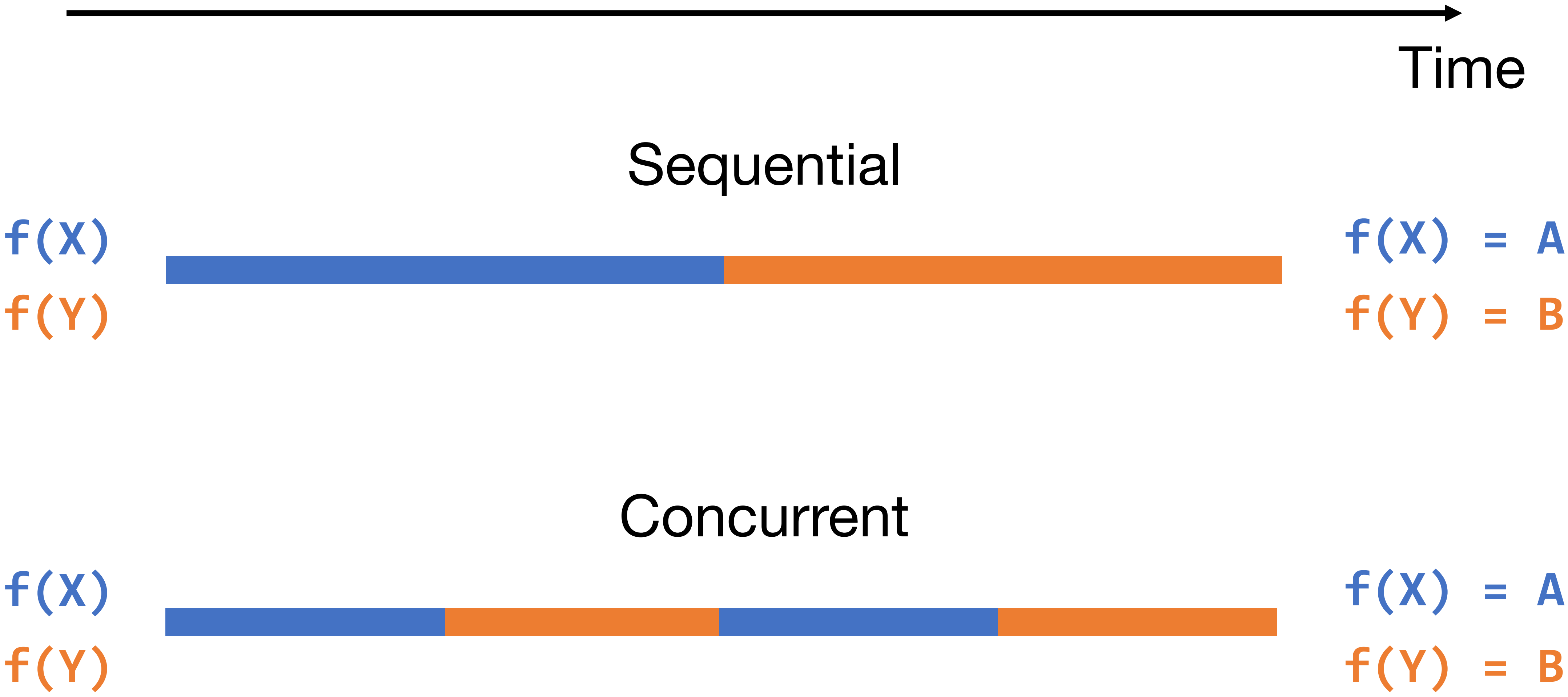
What is Parallelism?



Parallelism in Go

Demo: parallel.go

What is Concurrency?



Time

Sequential

f(X)

f(Y)

f(X) = A

f(Y) = B

Concurrent

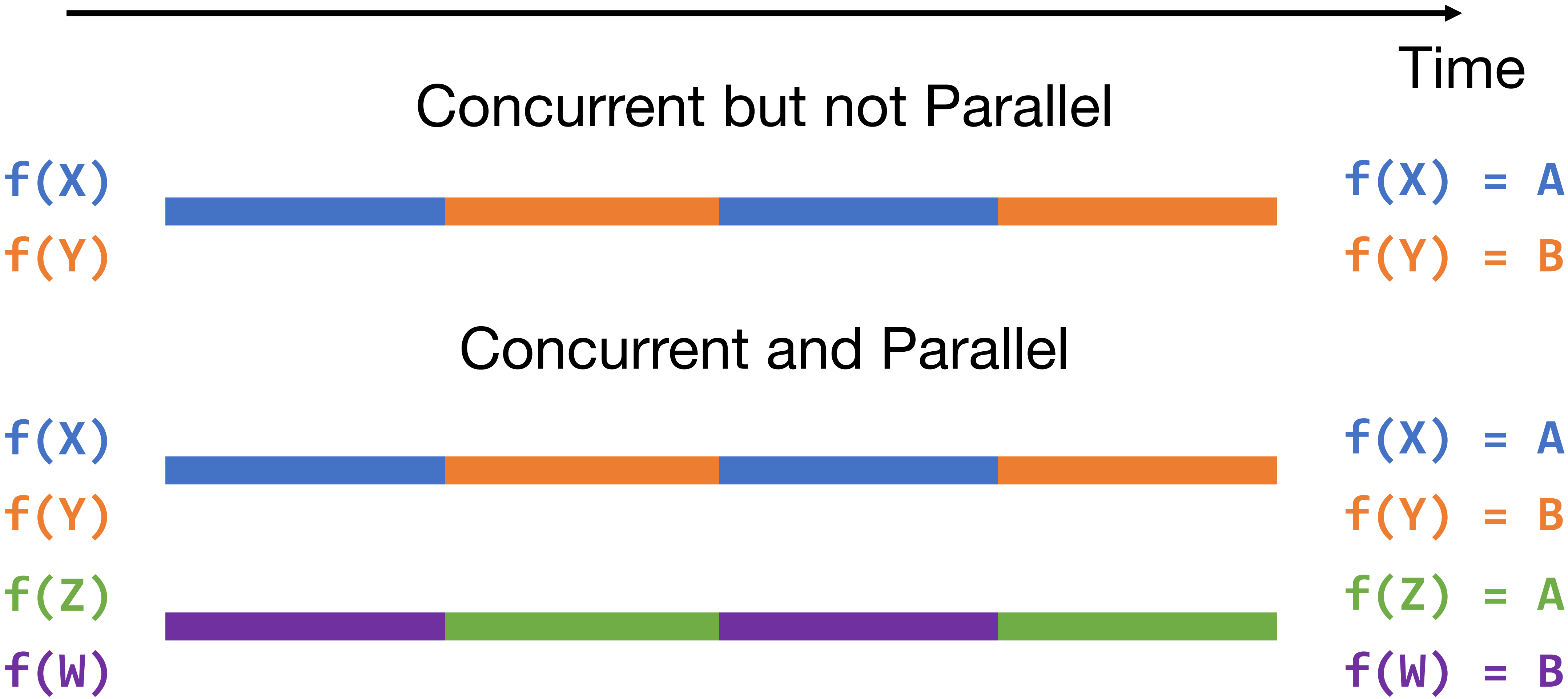
f(X)

f(Y)

f(X) = A

f(Y) = B

Concurrency Could be Parallel but not Always



Time

Concurrent but not Parallel

f(X)

f(Y)

f(X) = A

f(Y) = B

Concurrent and Parallel

f(X)

f(Y)

f(Z)

f(W)

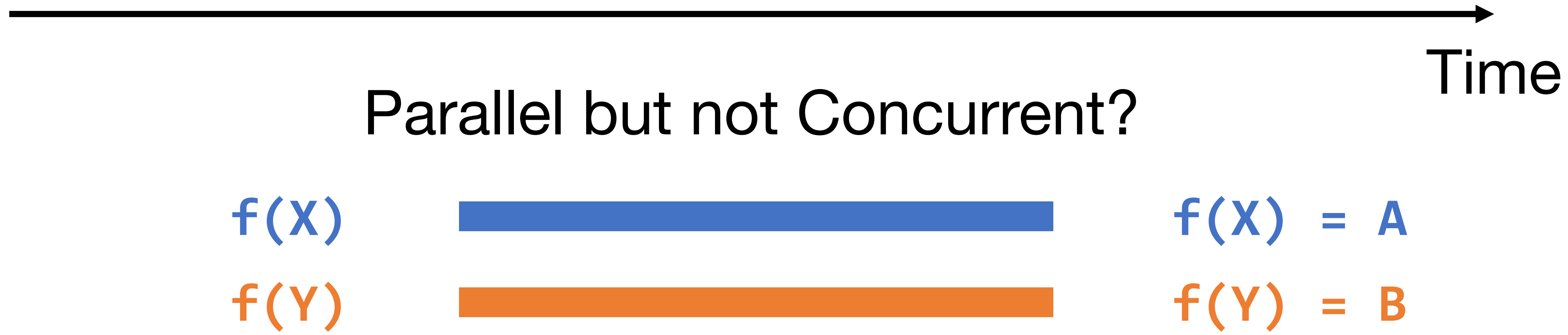
f(X) = A

f(Y) = B

f(Z) = A

f(W) = B

Parallel is Always Concurrent



Time

Parallel but not Concurrent?

`f(X)` `f(X) = A`

`f(Y)` `f(Y) = B`

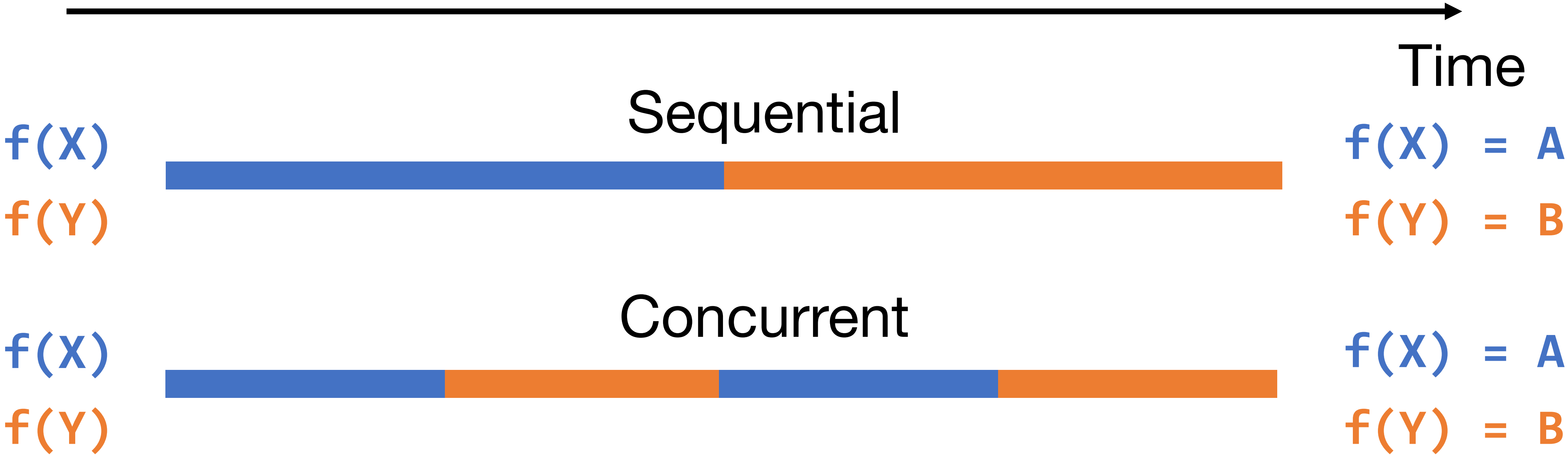
Nope … still concurrent

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

”Concurrency is not Parallelism” by Rob Pike : https://talks.golang.org/2012/waza.slide#1

Distributed Systems are Unpredictable

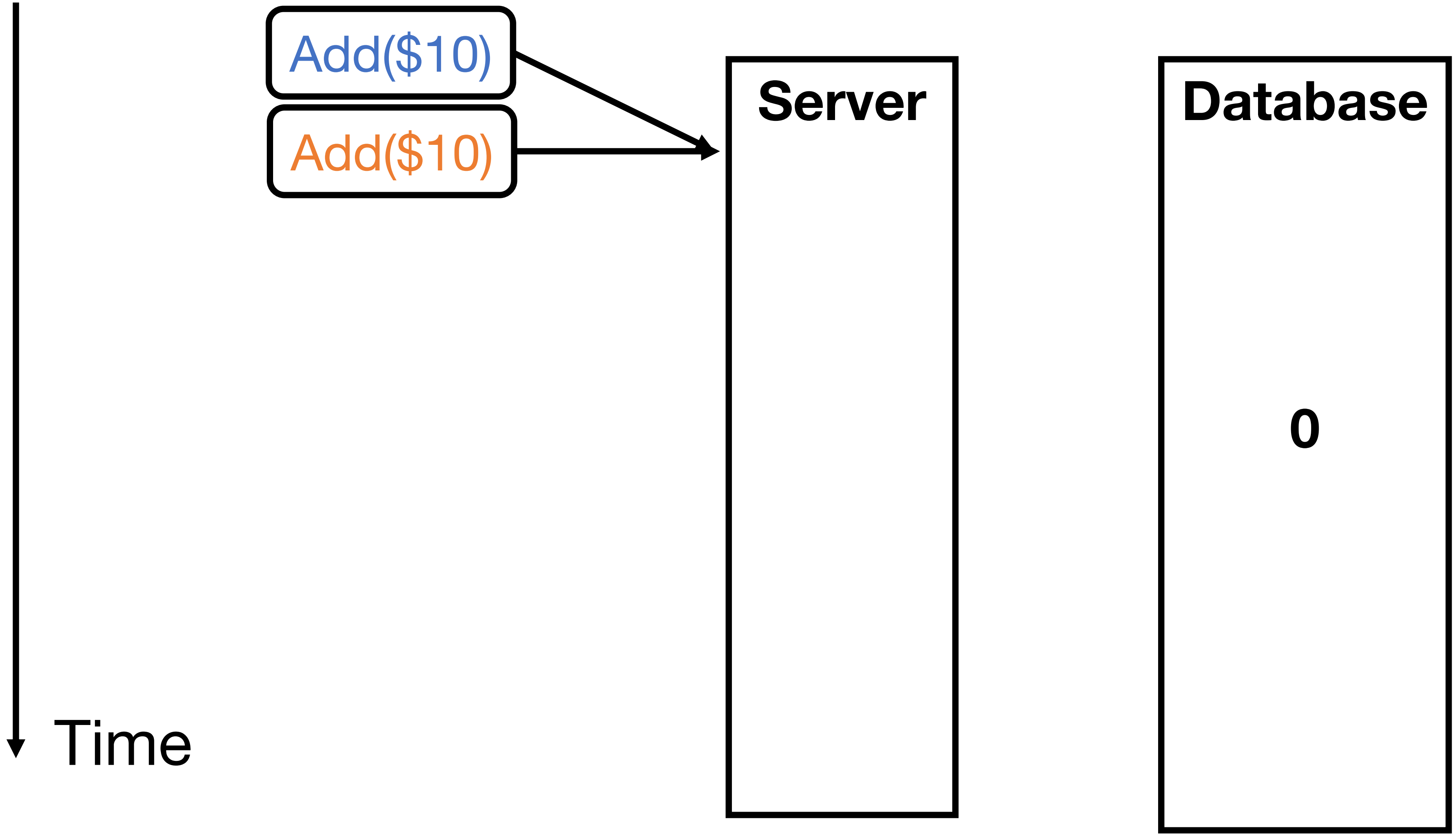
Servers need to react to:

- Others servers
- Crashes
- Users
- …

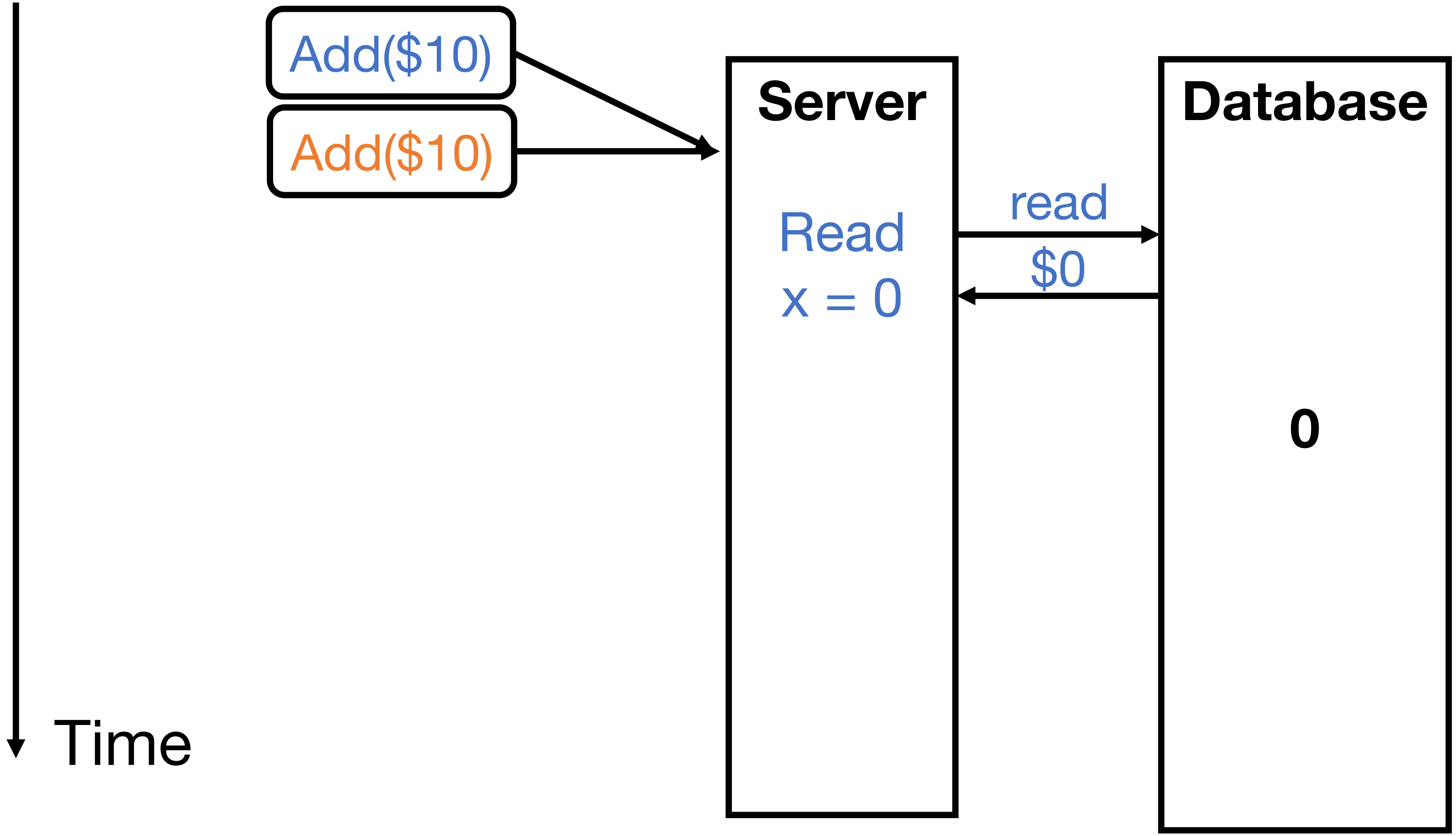
The Design Problem Concurrency Solves

Demo: concurrent.go

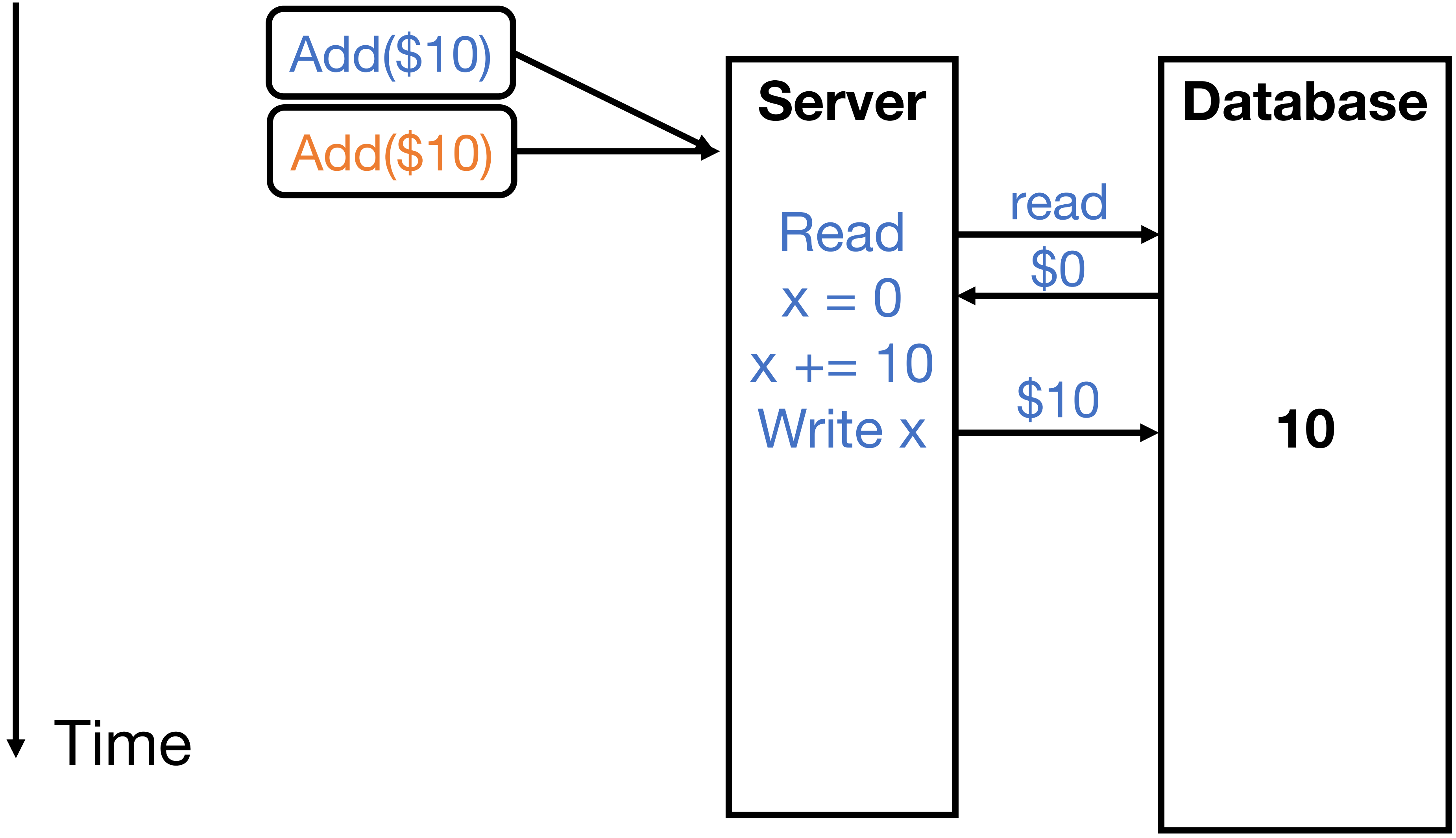
Making Bank Deposits Concurrent (1/5)



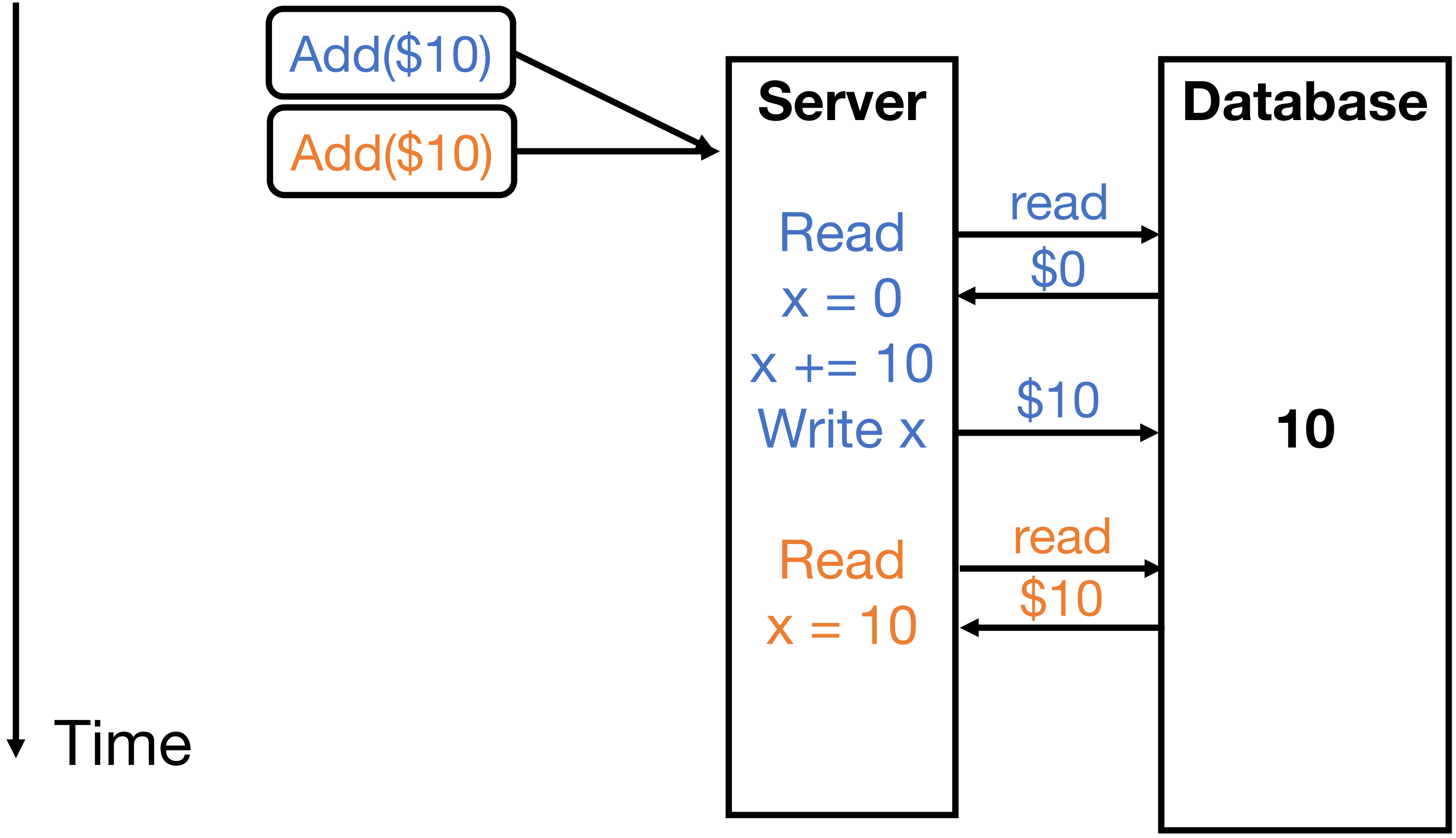
Making Bank Deposits Concurrent (2/5)



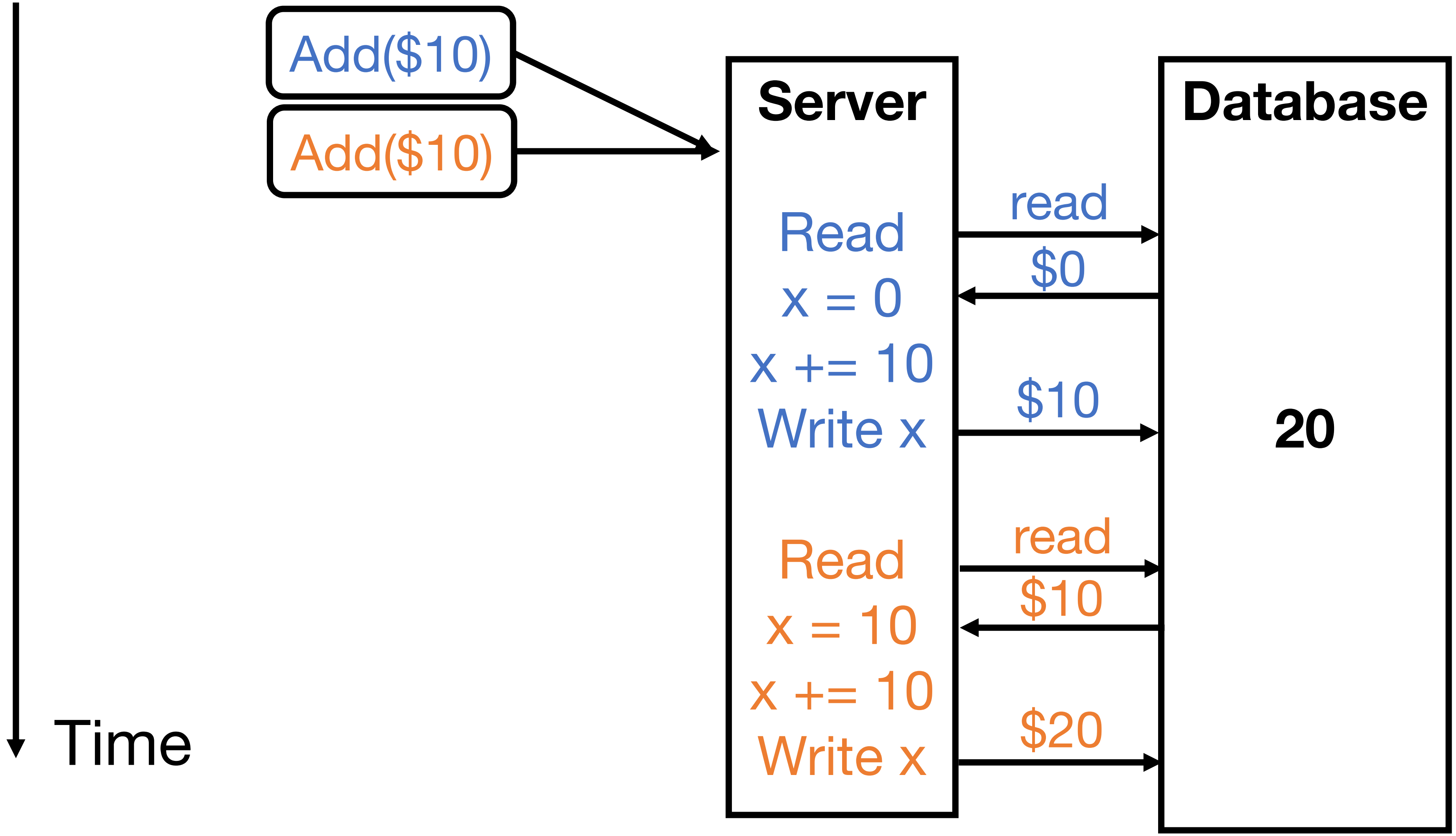
Making Bank Deposits Concurrent (3/5)



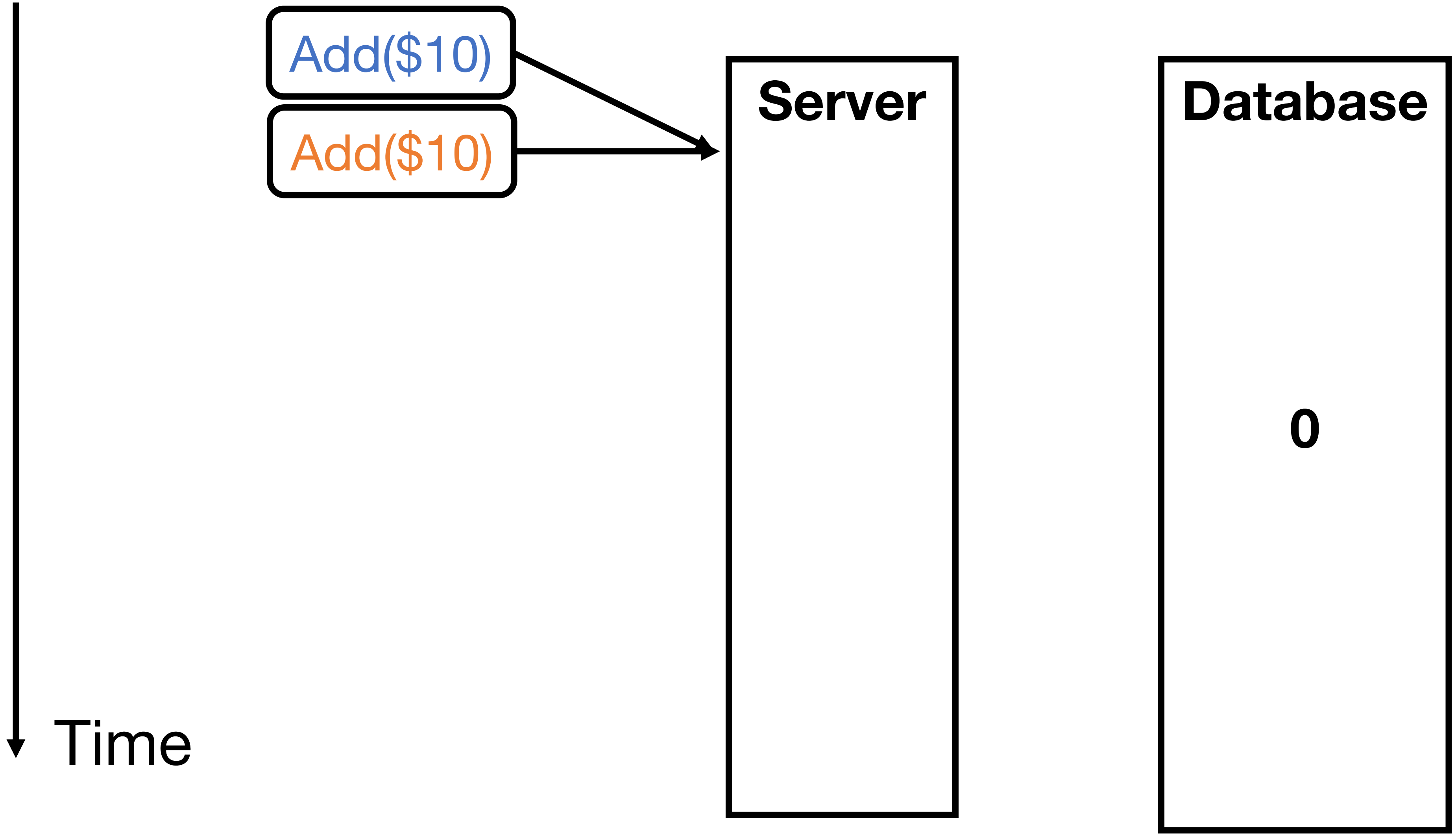
Making Bank Deposits Concurrent (4/5)



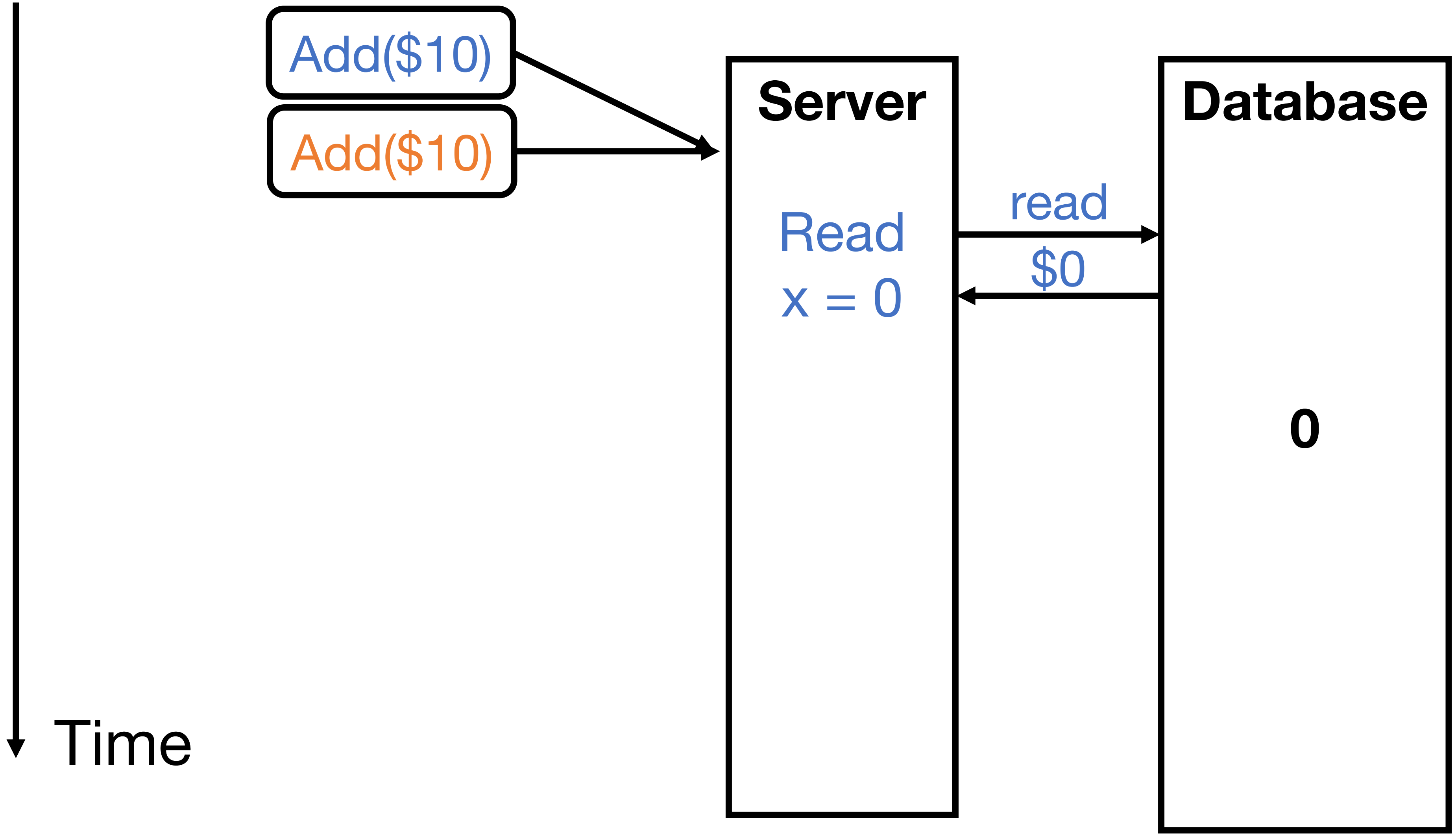
Making Bank Deposits Concurrent (5/5)



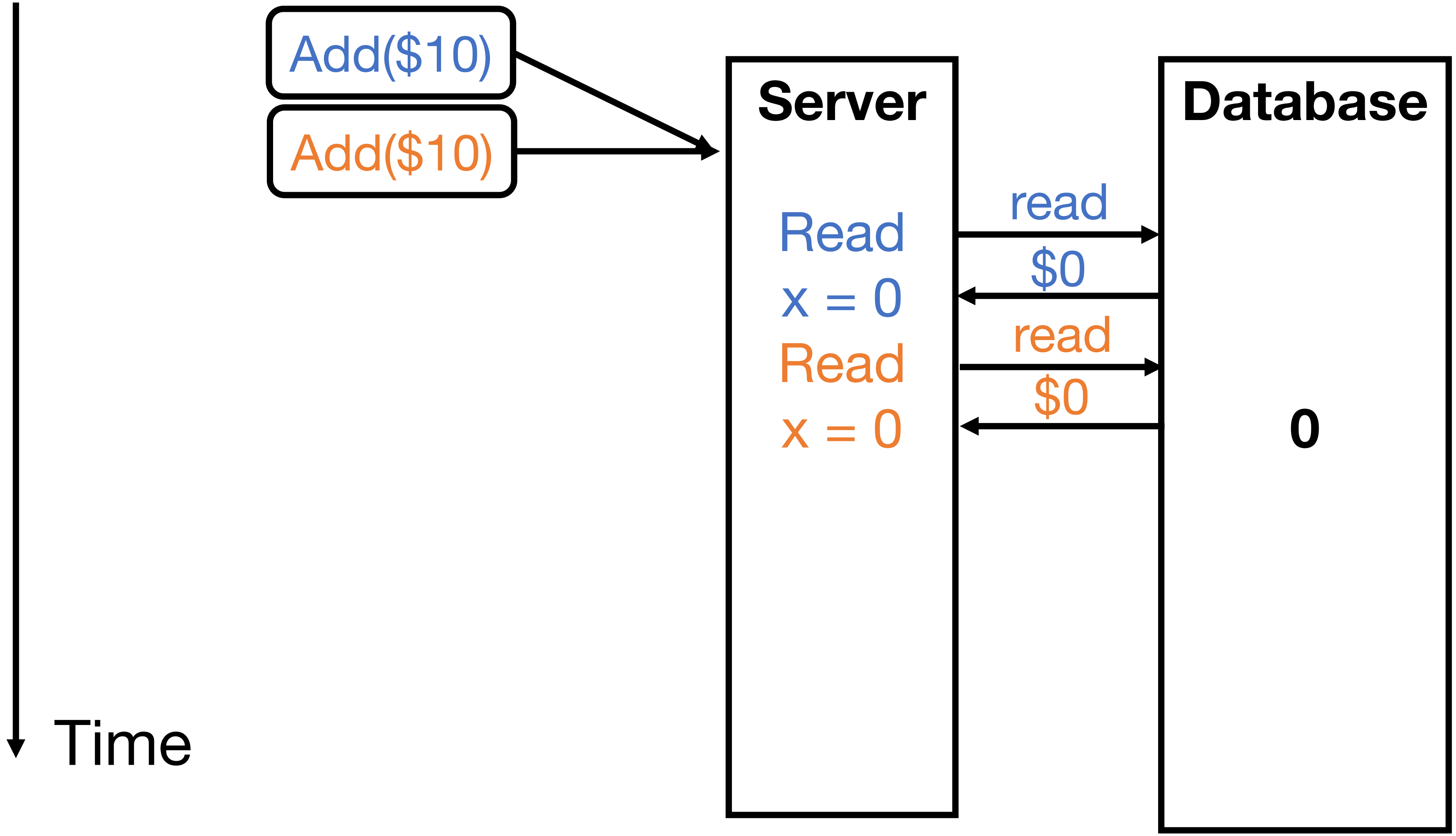
Concurrent Bank Deposits! Yay? (1/5)



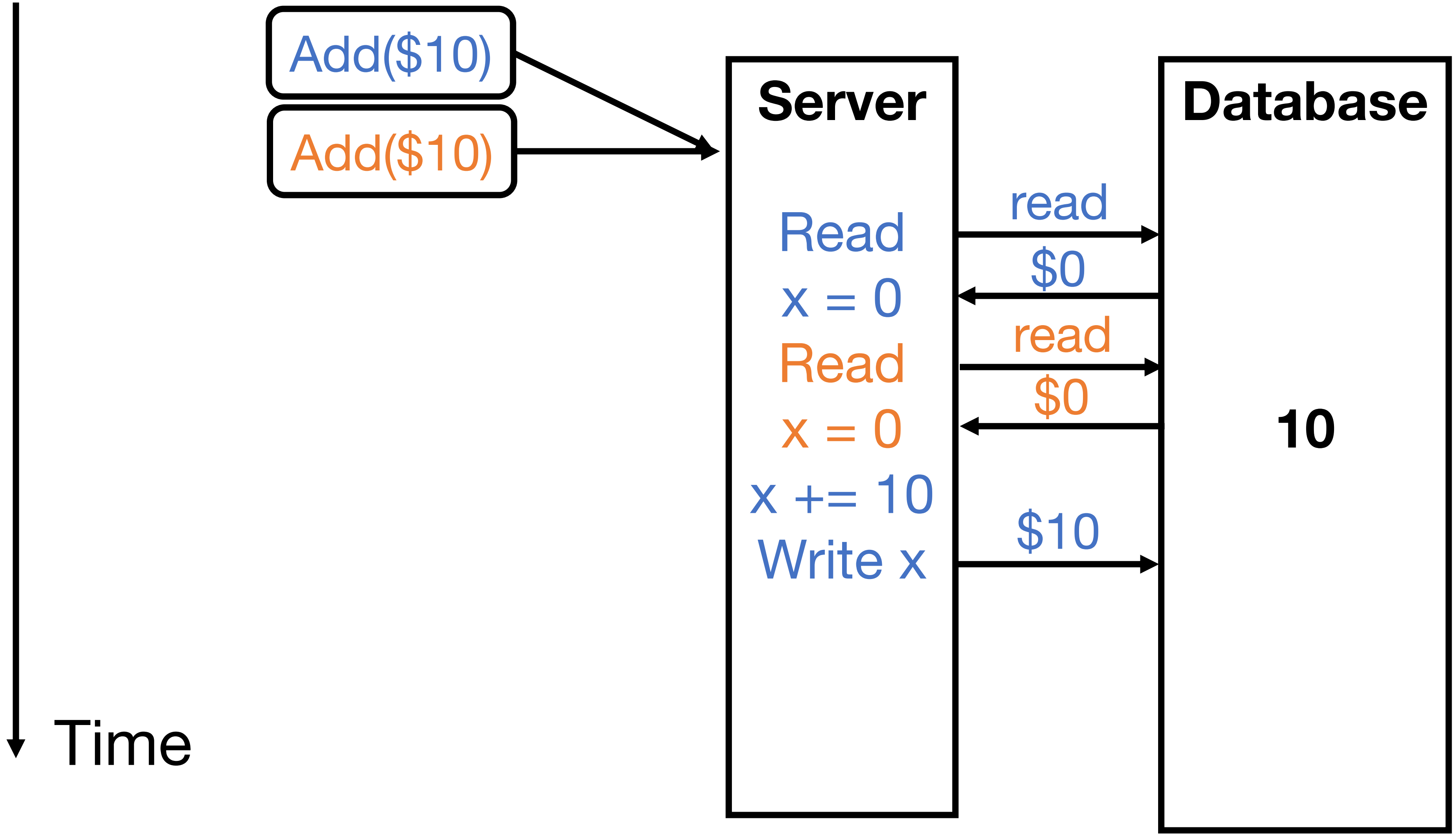
Concurrent Bank Deposits! Yay? (2/5)



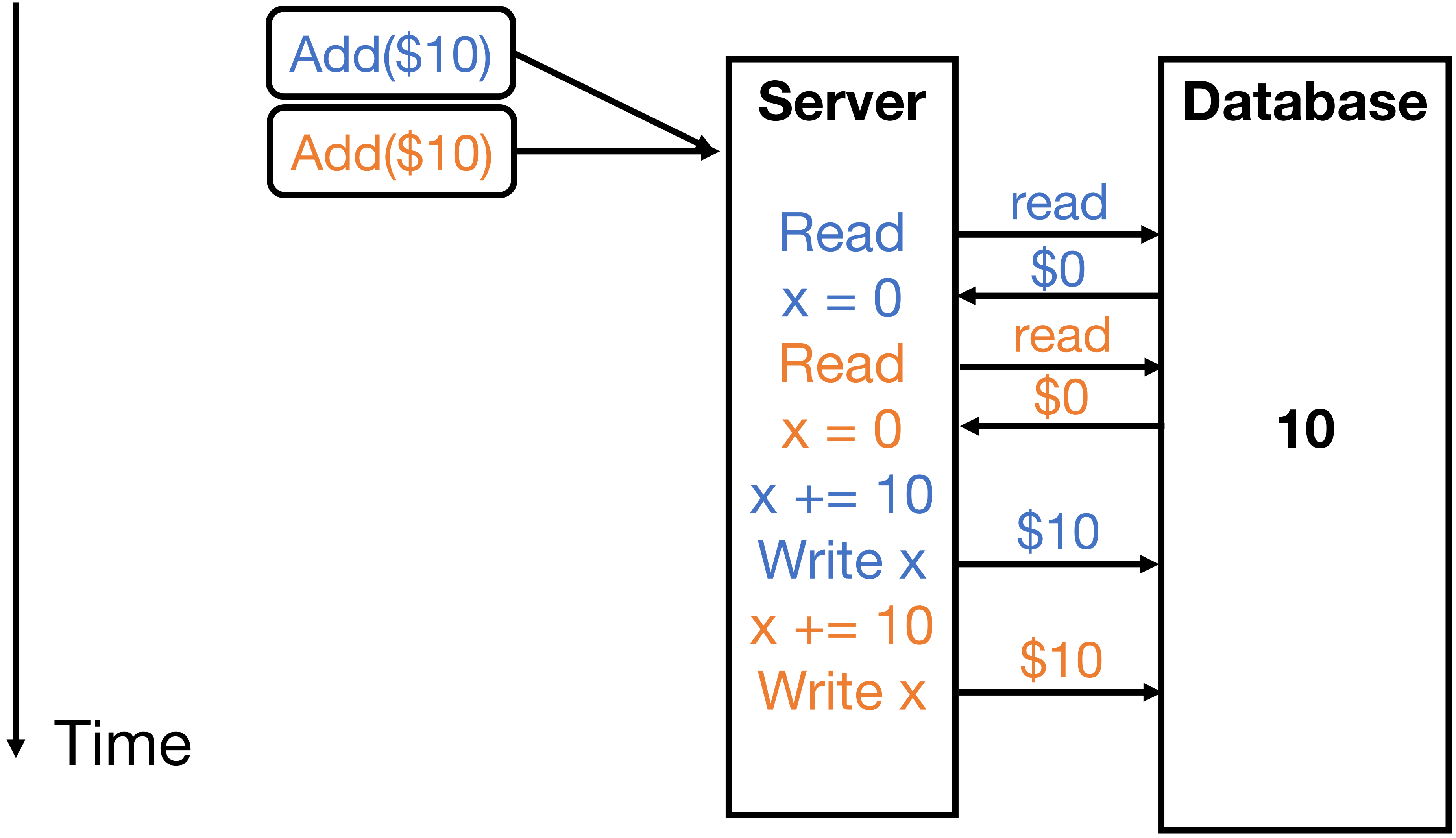
Concurrent Bank Deposits! Yay? (3/5)



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

Channels, Locks and More

Demo: sync.go

Visualize Everything We’ve Learned

And also see many different methods of achieving synchronization: http://divan.github.io/posts/go_concurrency_visualize/