BIG DATA MANAGEMENT

CE/CZ4123

DISTRIBUTED SYSTEMS AND MAP-REDUCE

Siqiang Luo

Assistant Professor

LEARNING OUTCOMES

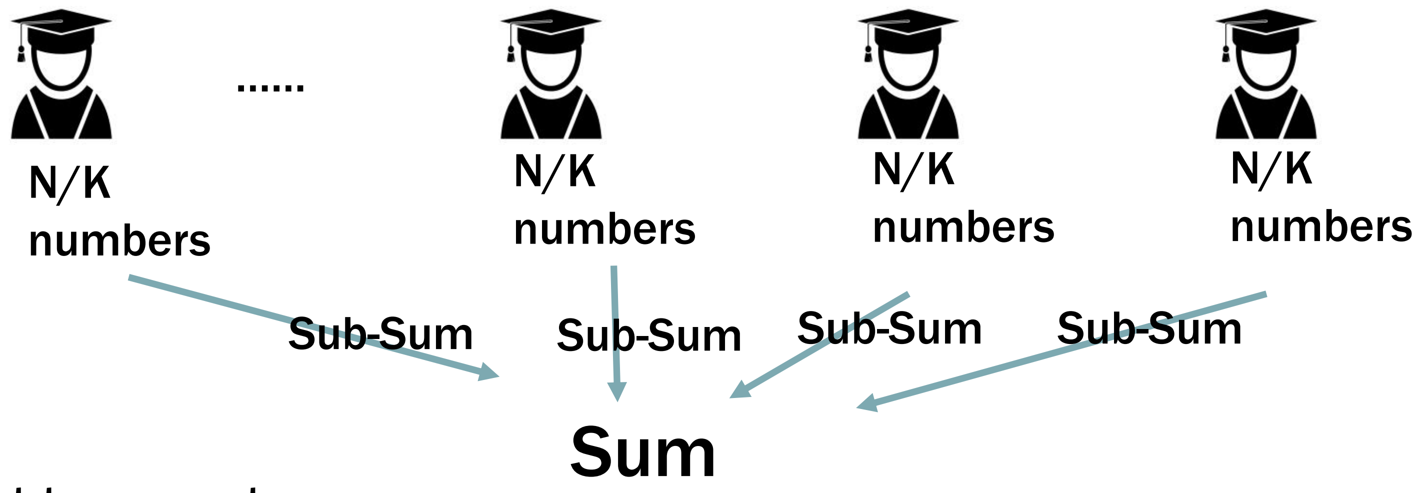
- ❑ Understand the concept of distributed computation
- ❑ Understand the MapReduce computation model
- ❑ Can design proper algorithms based on MapReduce computation model
- ❑ Have a basic concept of Hadoop system

STARTING QUESTION

Give *N* integers to a team of *K* students to compute the sum. The numbers are given to the team leader. If you are the leader, how would you accomplish the computation task as soon as possible?

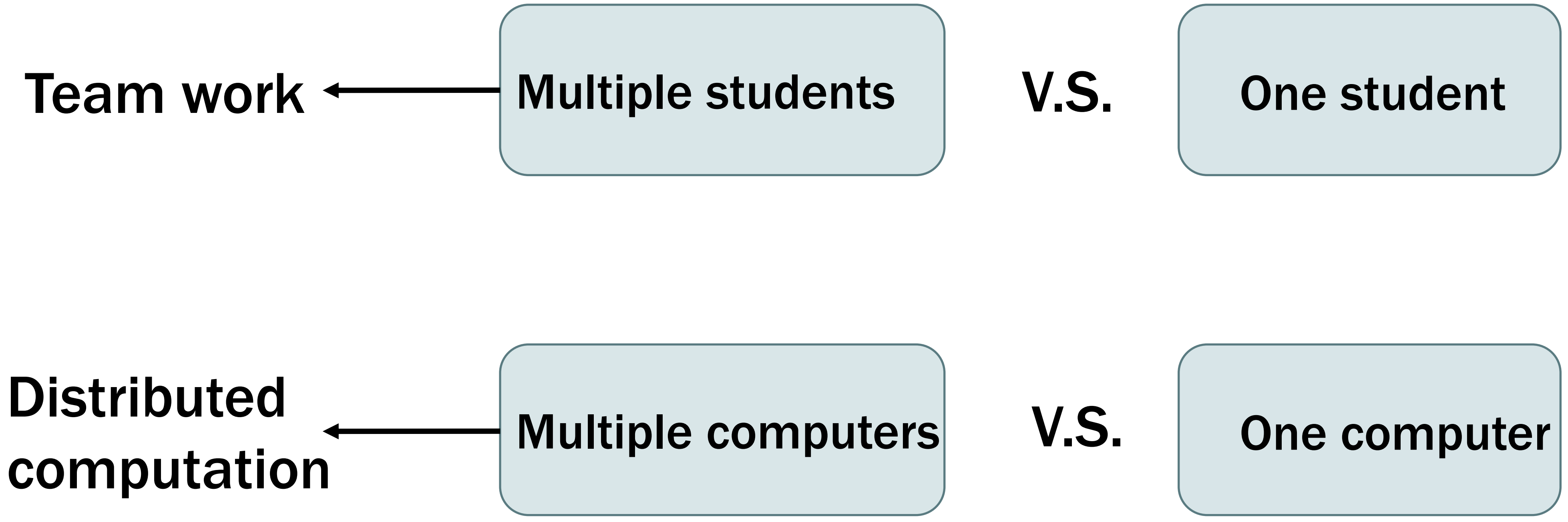
STARTING QUESTION

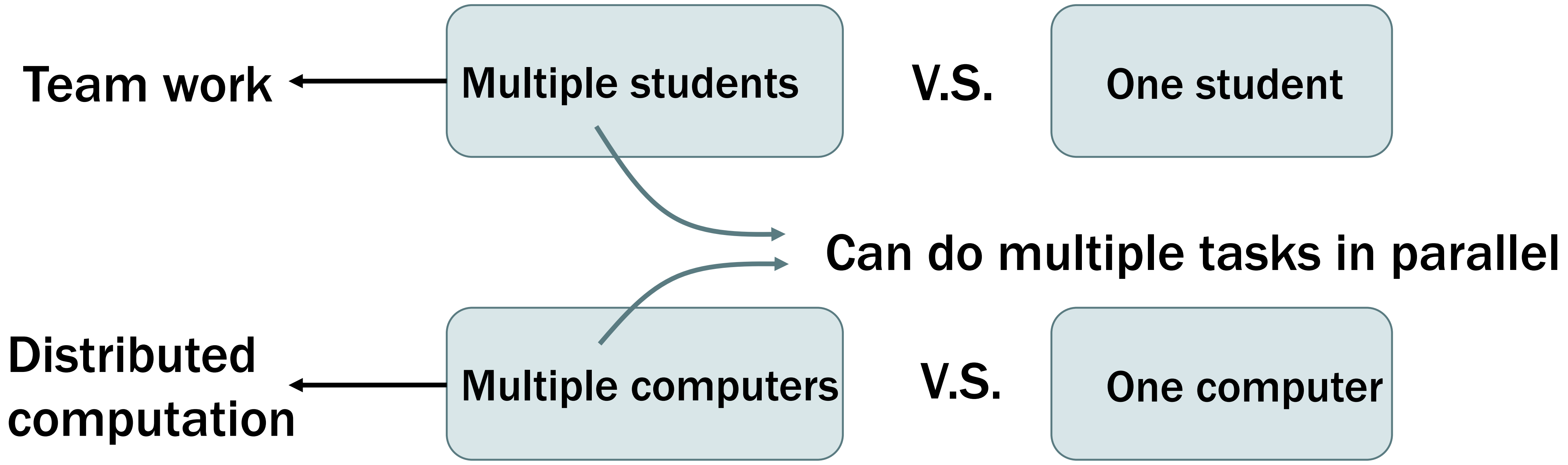
Strategy 1: Divide N numbers to K students to compute, and aggregate the results



Strategy 2: Ask one student to compute

Is there a better strategy?


Team work
Multiple students
V.S.
One student
Distributed computation
Multiple computers
V.S.
One computer


Team work
Multiple students
V.S.
One student
Can do multiple tasks in parallel
Distributed computation
Multiple computers
V.S.
One computer

If the task is to sum up 10000 integers, more people will be better.

Distributed computation is a very important approach in big data applications!

Basics of Distributed Computation

Distributed computation is often discussed in the context of big data

Not small data

DISTRIBUTED SYSTEMS

- Suppose an engineer from Amazon needs to process huge data of daily purchases.
- Just use a single commodity computer?

DISTRIBUTED SYSTEMS

- Suppose an engineer from Amazon needs to process huge data of daily purchases.
- Just use a single commodity computer?
  - Lack of storage
  - Lack of computation power

DISTRIBUTED SYSTEMS

- Suppose an engineer from Amazon needs to process huge data of daily purchases.
- Just use a single commodity computer?
  - Lack of storage
  - Lack of computation power

Use multiple machines ~ Distributed system

WHAT IS DISTRIBUTED SYSTEM

A *distributed system* is a system whose components are located on different **networked computers**, which communicate and coordinate their actions by passing messages to one another from any system. The components interact with one another in order to achieve a common goal.

DISTRIBUTED SYSTEMS FOR BIG DATA: CHALLENGES

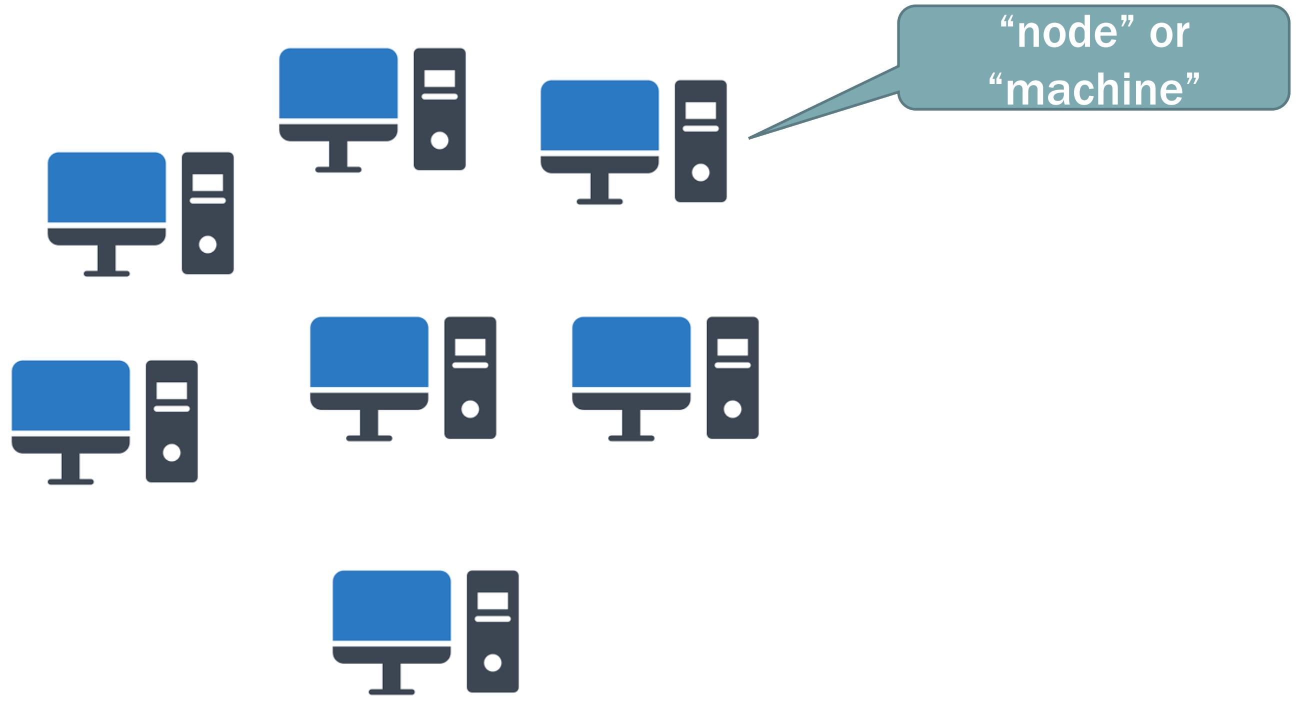
- ❑ How to organize the machines?
- ❑ How to store data across machines?
- ❑ How to compute using multiple machines?

DISTRIBUTED SYSTEMS FOR BIG DATA: CHALLENGES

- ❑ **How to organize the machines?**
- ❑ How to store data across machines?
- ❑ How to compute using multiple machines?

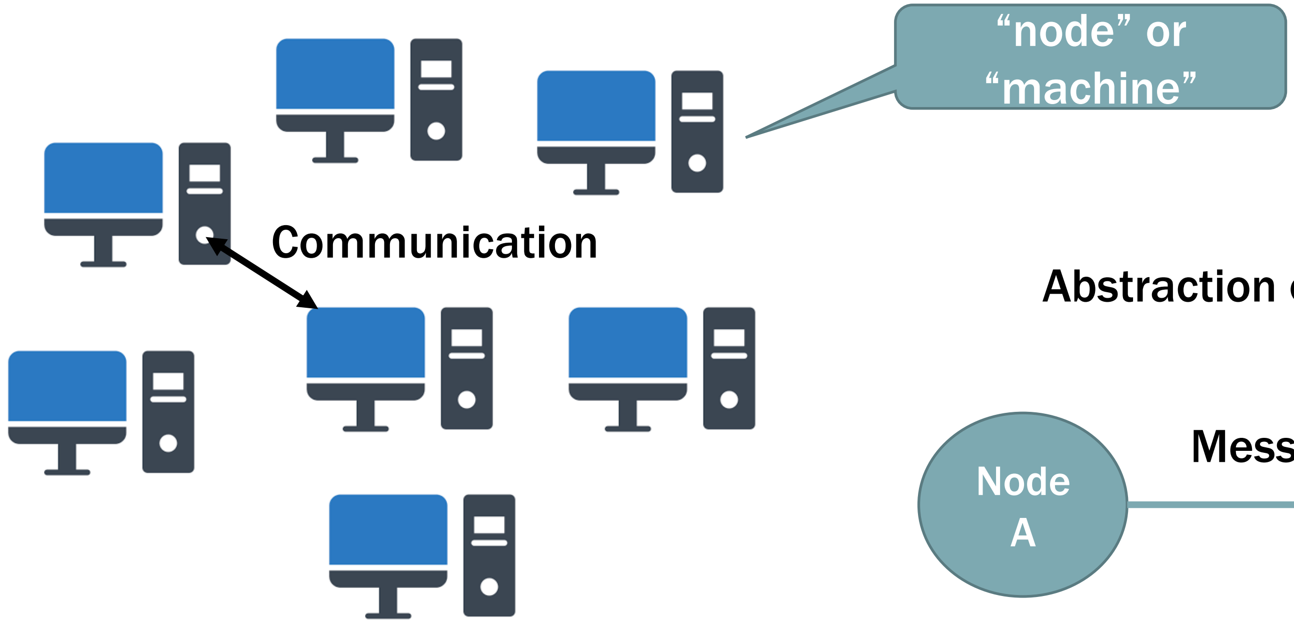
DISTRIBUTED SYSTEMS

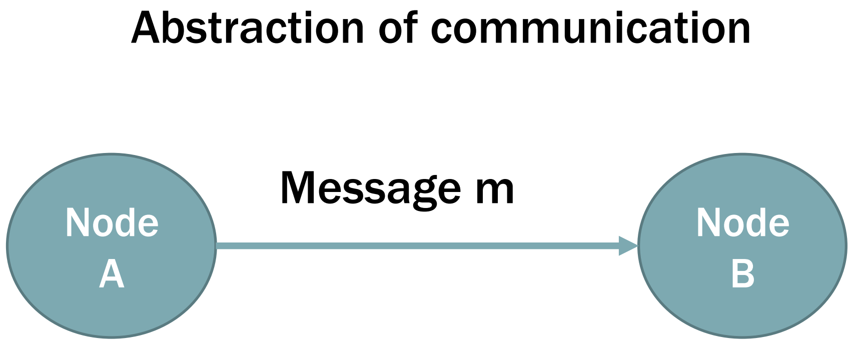
- ❑ The machines are connected via high-bandwidth networks.
- ❑ Two common models:
  - ❑ Fully distributed: Any two machines can access each other
  - ❑ Master-Slaves: There exists one coordinator



FULLY DISTRIBUTED MODELS

- Each machine has an IP address
- *A* knows machine *B*'s IP address: *A* can send messages to *B*
- Two machines can communicate with each other via IP address
  - i.e., sending messages between machines





COMMUNICATION IS NOT FOR FREE

Latency: time spent on sending/receiving messages

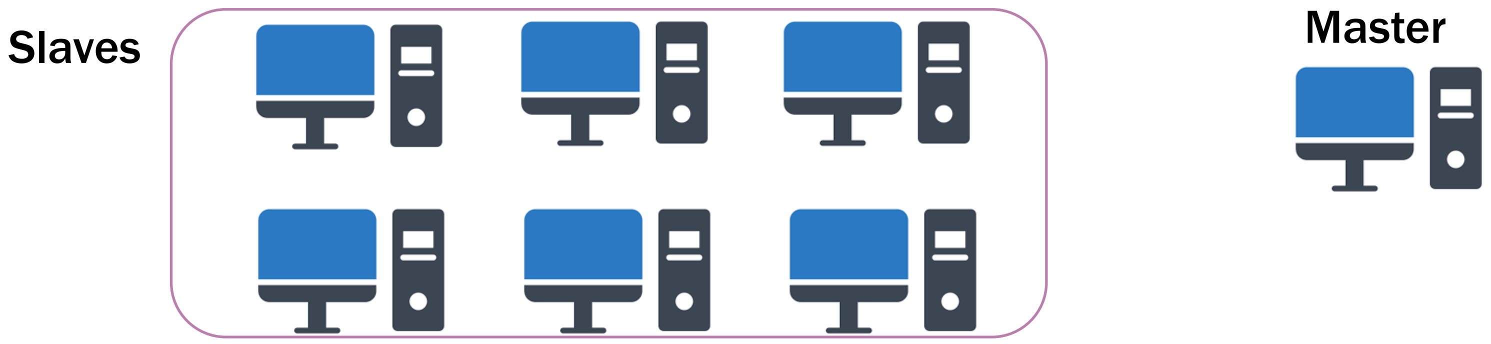
- ❑ Same datacenter: around 1ms
- ❑ One continent to another: around 100ms

Bandwidth: data volume per unit time

- ❑ 3G cellular data: around 1Mbit/s
- ❑ 4G cellular data: around 100Mbit/s

MASTER-SLAVE MODEL

- ❑ Each machine has an IP address
- ❑ There is a machine called master, and the other machines are called slaves.
- ❑ Master is the coordinator, being responsible to
  - ❑ distribute tasks to the slaves, and
  - ❑ receive the results from the slaves



DISTRIBUTED SYSTEMS FOR BIG DATA: CHALLENGES

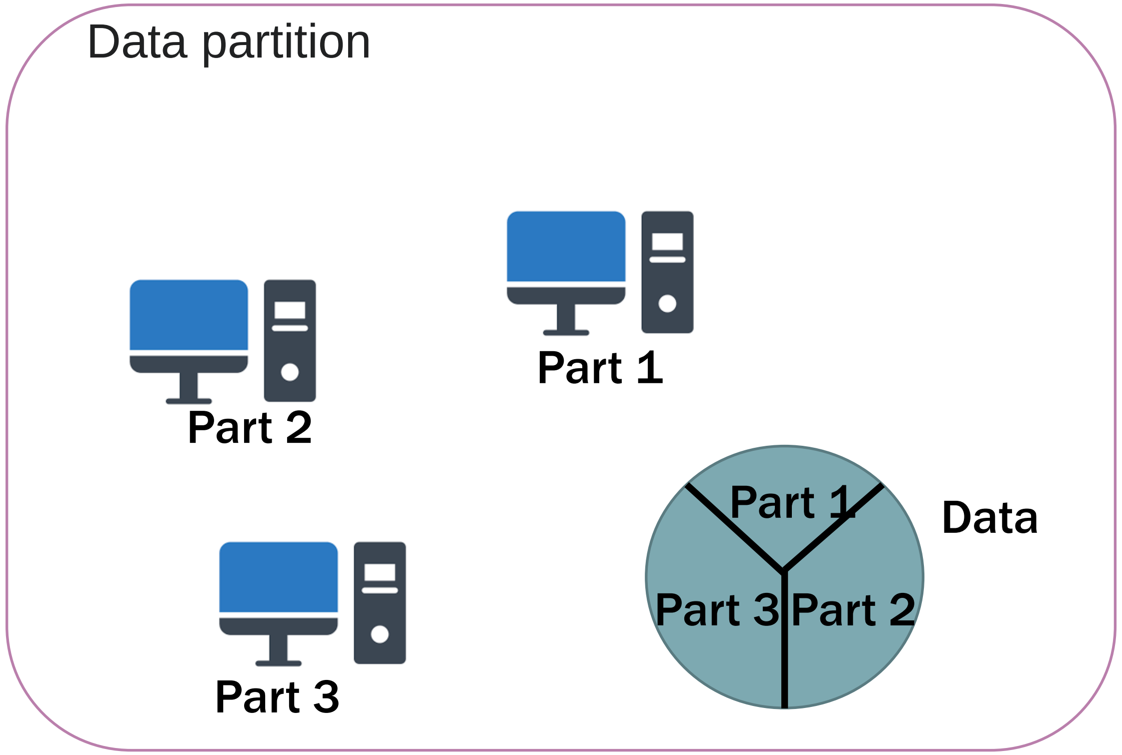
❑ How to organize the machines?

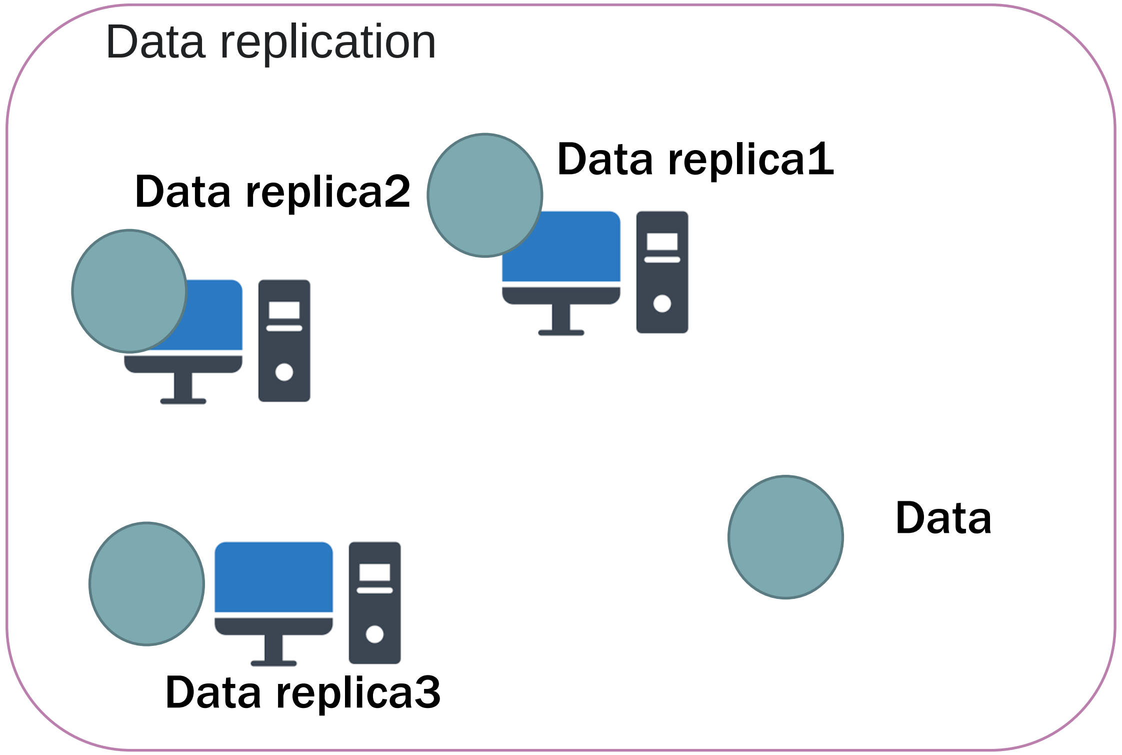
❑ **How to store data across machines?**

❑ How to compute using multiple machines?

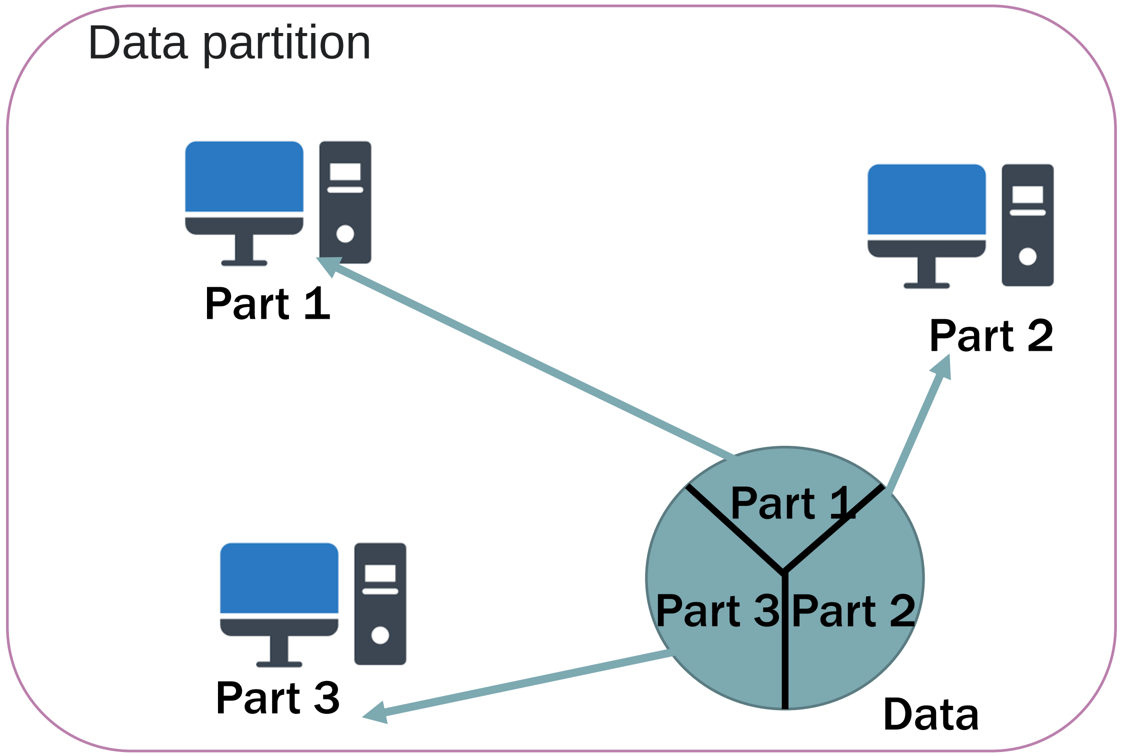
DATA PARTITION AND DATA REPLICATION

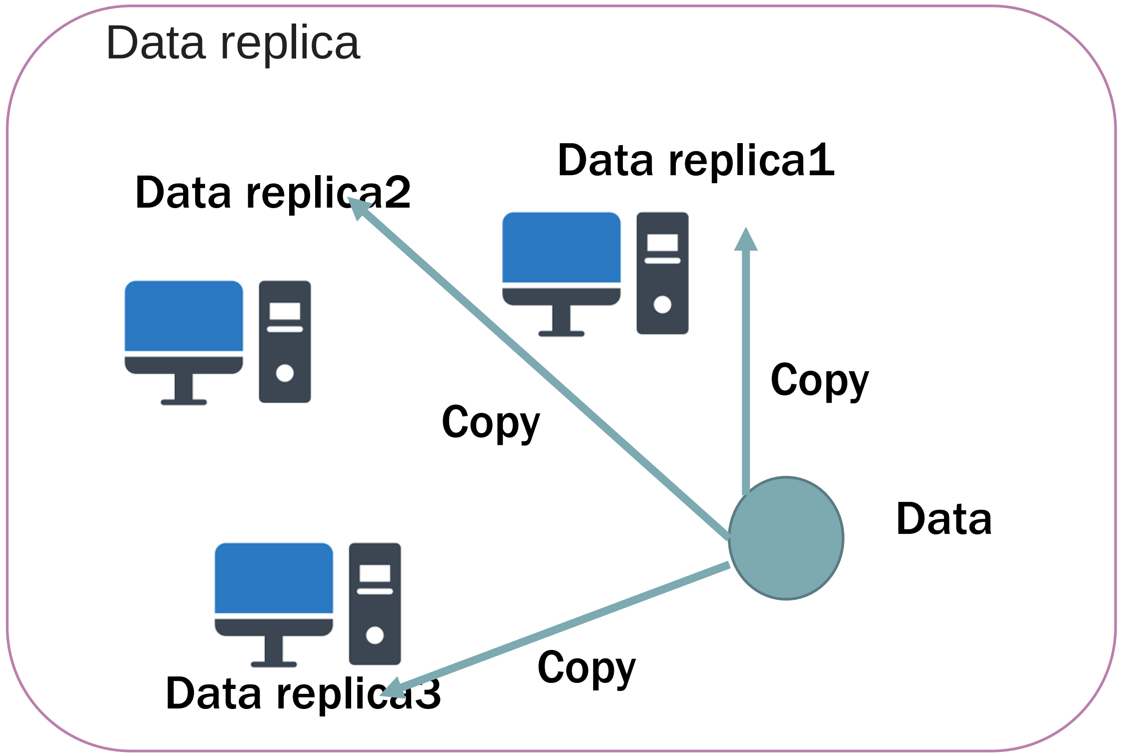
- ❑ Data partition: partition the data into different machines
- ❑ Data replication: each data item can be replicated to multiple copies.





DATA PARTITION AND DATA REPLICA





EXAMPLE: DATA PARTITION

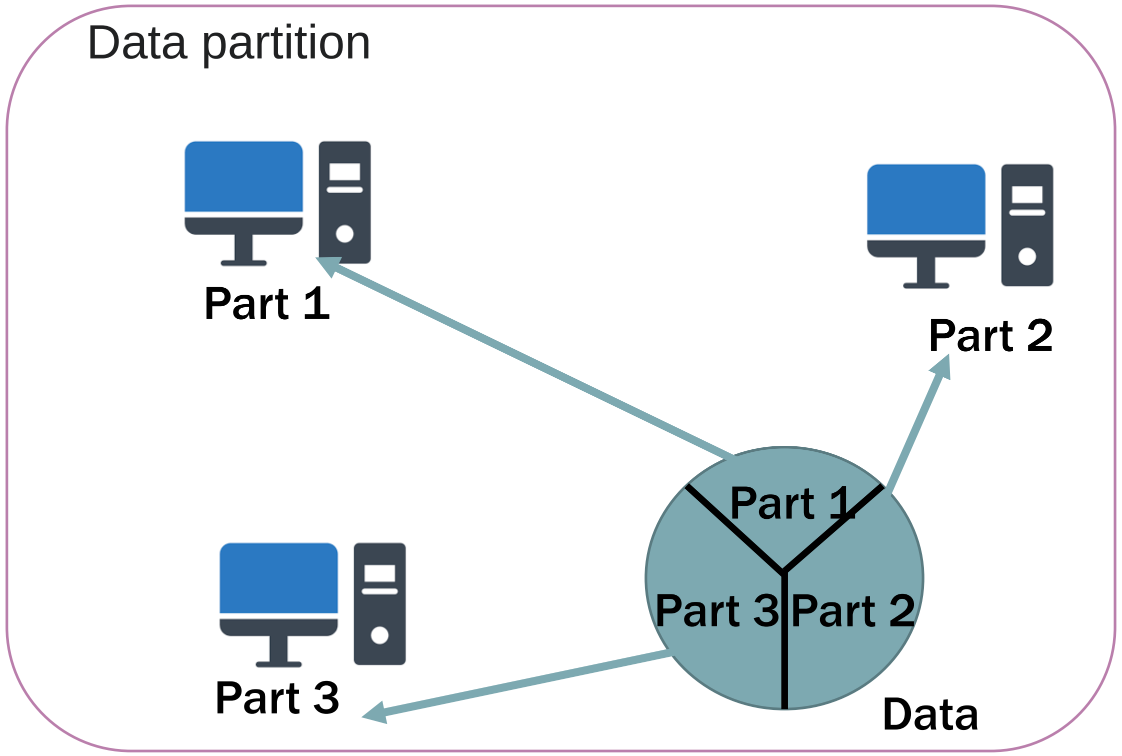
Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22

Name	Age
Kyle	18
Nelson	23

Name	Age
Ben	19
Terry	21

DATA PARTITION



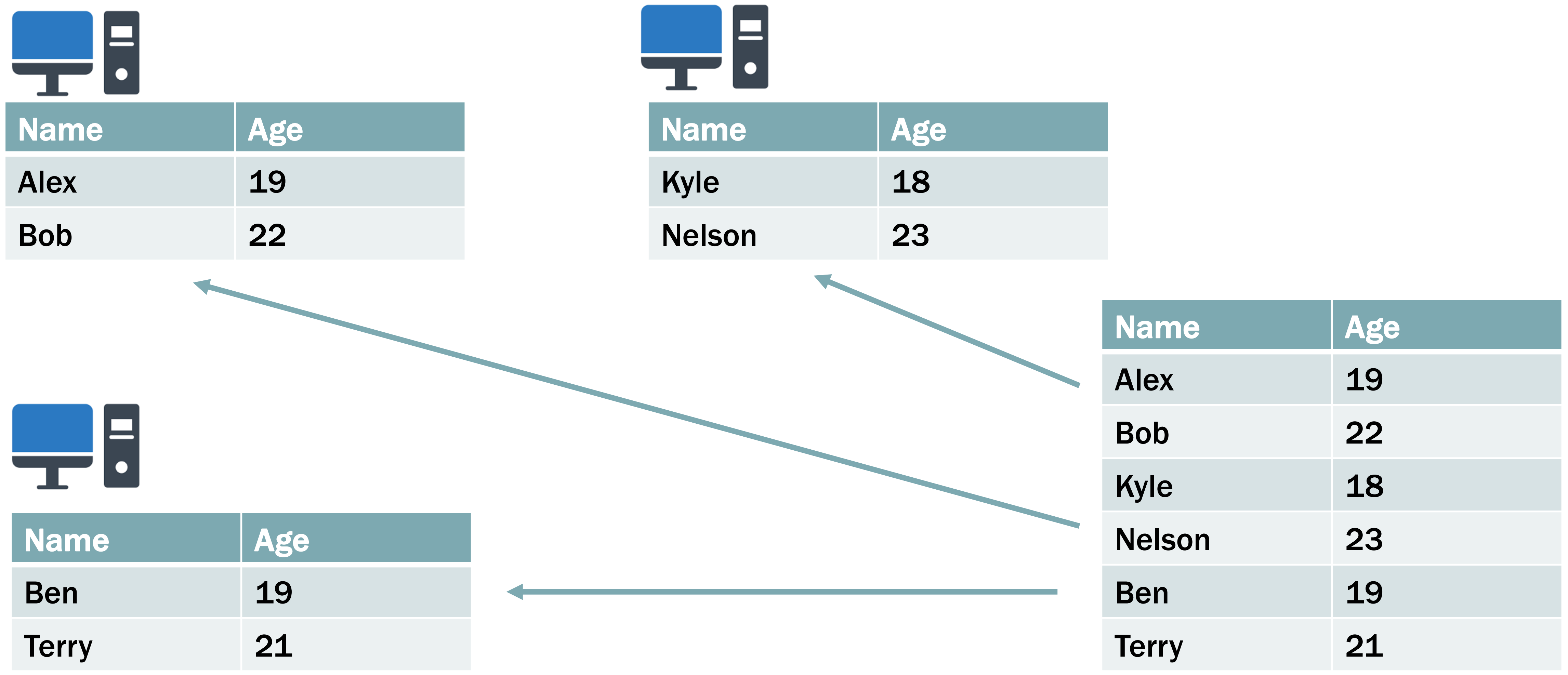
Advantages:

- **No space amplification**
(i.e., the total data size does not change)

- **Workload can be distributed to multiple machines.**

DATA PARTITION

Select students whose ages < 20



Name	Age
Alex	19
Bob	22

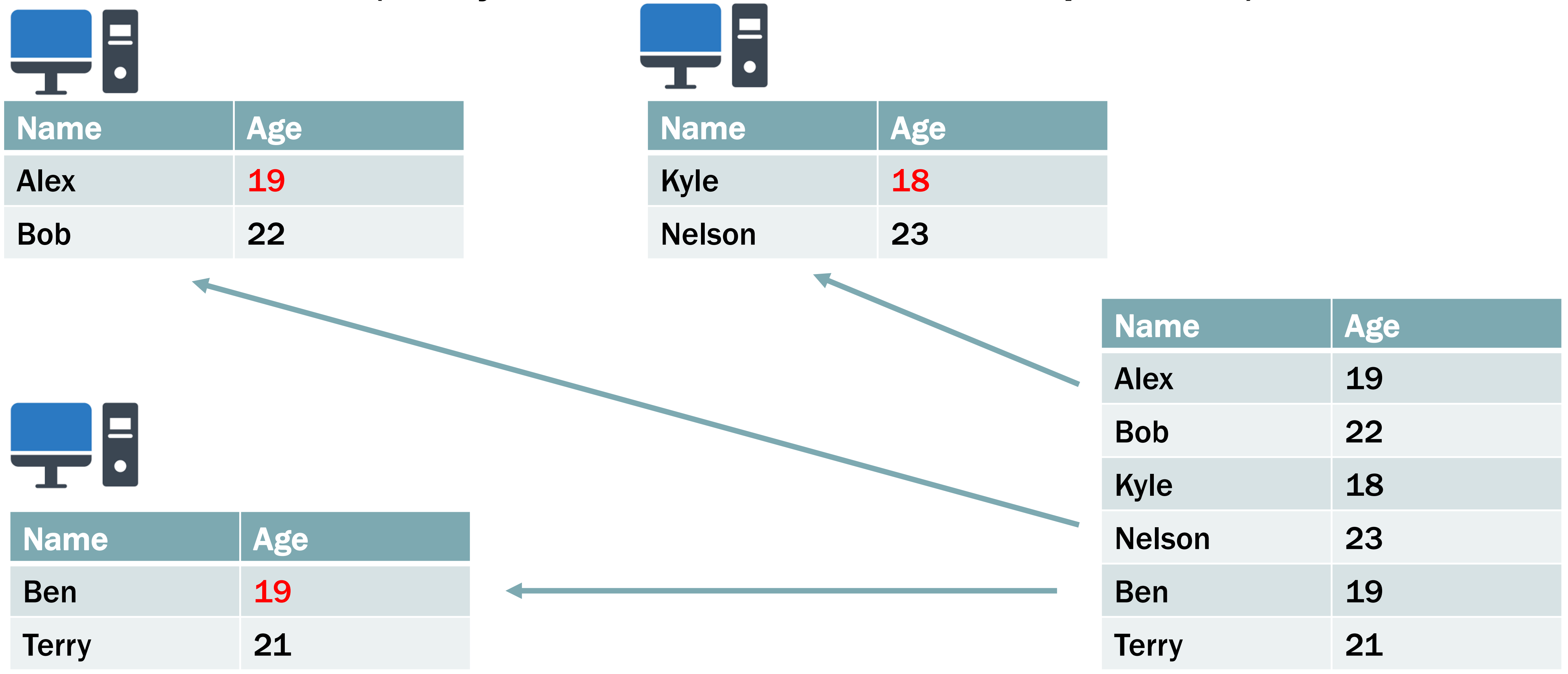
Name	Age
Kyle	18
Nelson	23

Name	Age
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

DATA PARTITION

Select students whose ages < 20
(every machine does some computation)



Name	Age
Alex	19
Bob	22

Name	Age
Kyle	18
Nelson	23

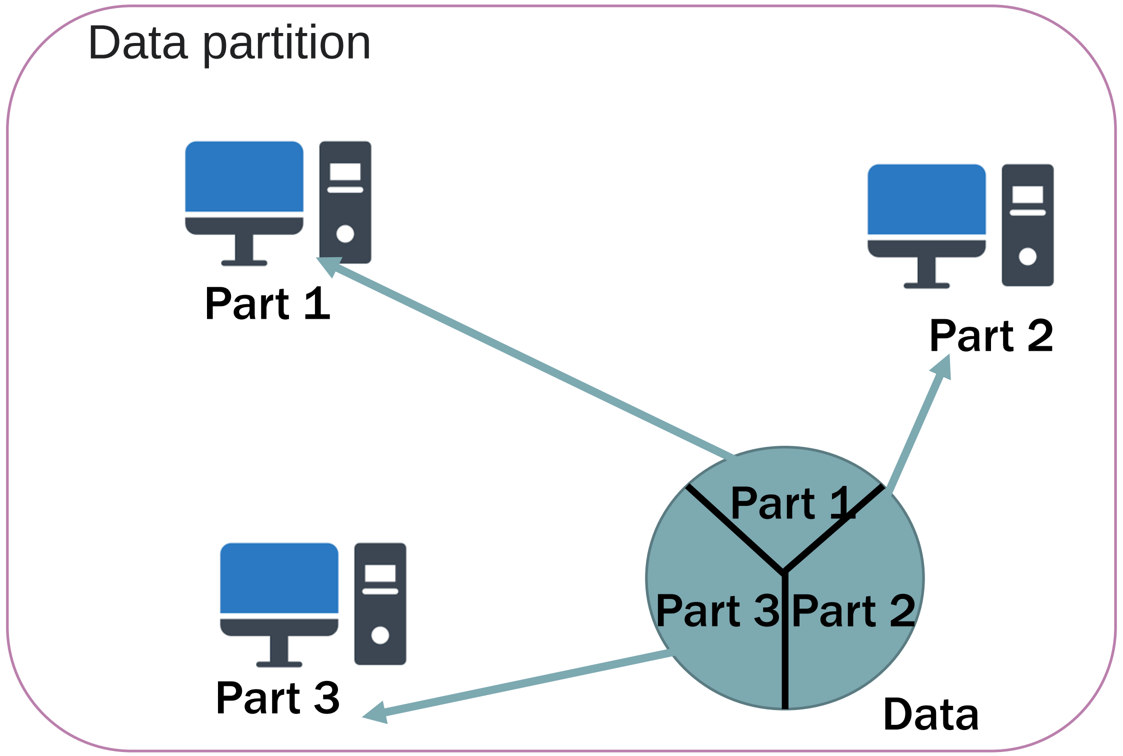
Name	Age
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

ANALYSIS

- ❑ Suppose the cost of doing a task on data D is F(D).
- ❑ Data D is divided into D={D_1,D_2,…,D_i}, which means machine j holds data D_j.
- ❑ The distributed computation cost can be estimated by max{F(D_1),…, F(D_i)}.
- ❑ Usually, we have max{F(D_1),…, F(D_i)} << F(D)

DATA PARTITION

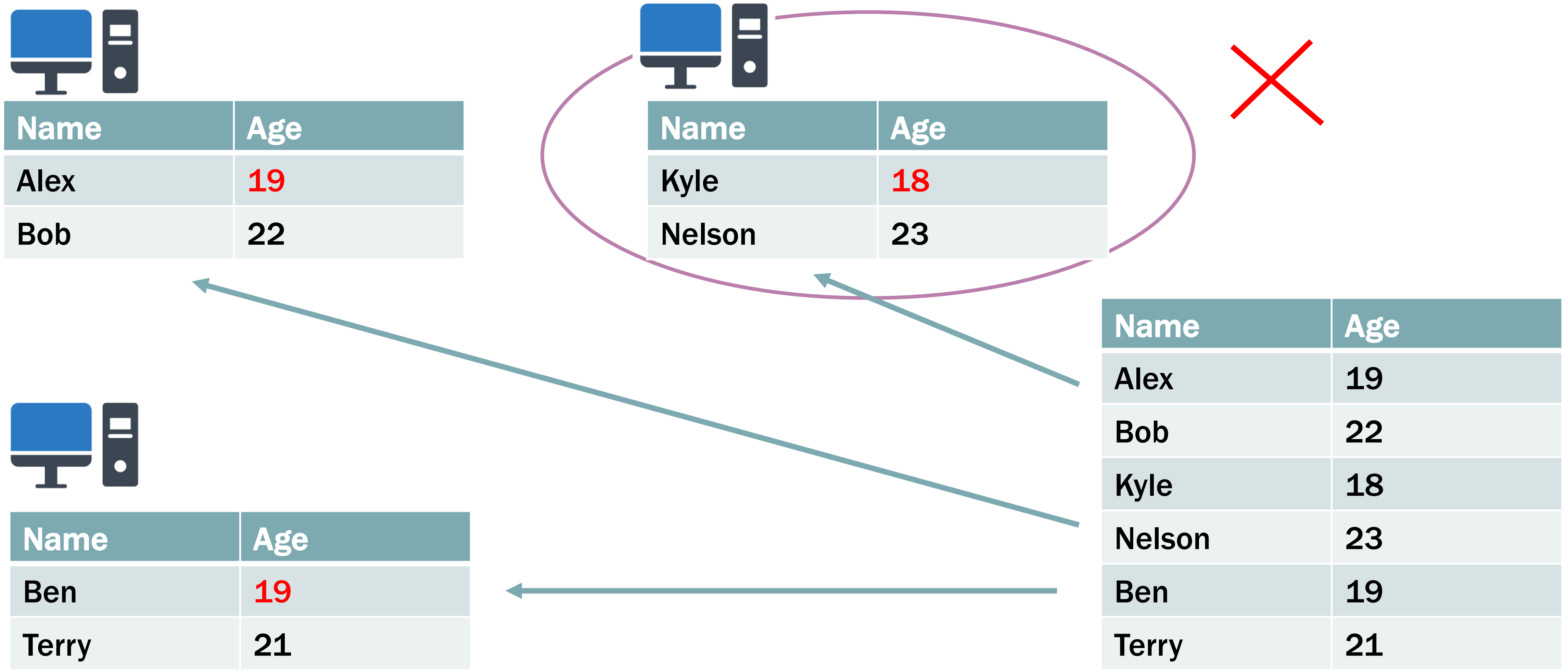


Disadvantages (if only using data partitioning):

- **Machines can be off for many reasons.**
- **Data lost, and results are not accurate.**

DATA PARTITION

Only {Ben, Alex} are returned, no Kyle.



Name	Age
Alex	19
Bob	22

Name	Age
Kyle	18
Nelson	23

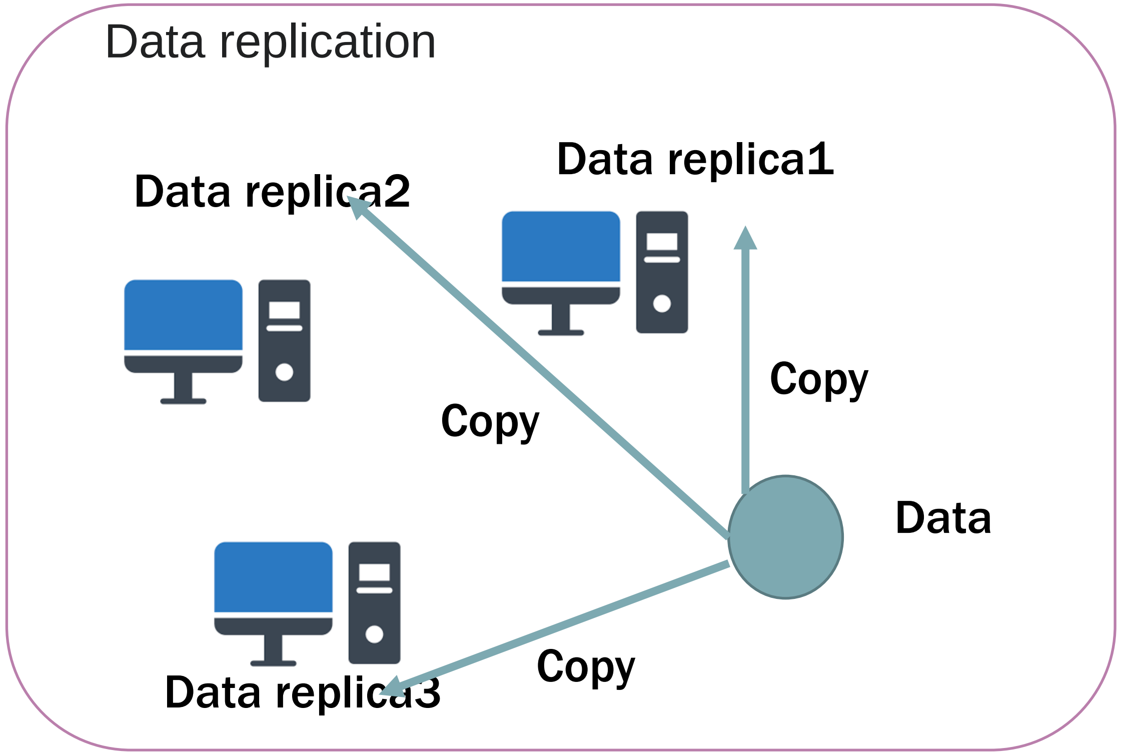
Name	Age
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

FAULT TOLERANCE

- Consider that a machine has 0.0001 probability of not being accessible at some time a day, and there are 1000 machines. What is the probability that all the machines in the cluster are always accessible within a month?
  - $(1-0.0001)^{1000*30}$ is about 5%.
- So, a large-scale distributed machine cluster always encounters machine failures. The process of handling machine failure is called fault tolerance.
- A fault-tolerant system ensures no loss of services when some machine fails.

DATA REPLICATION



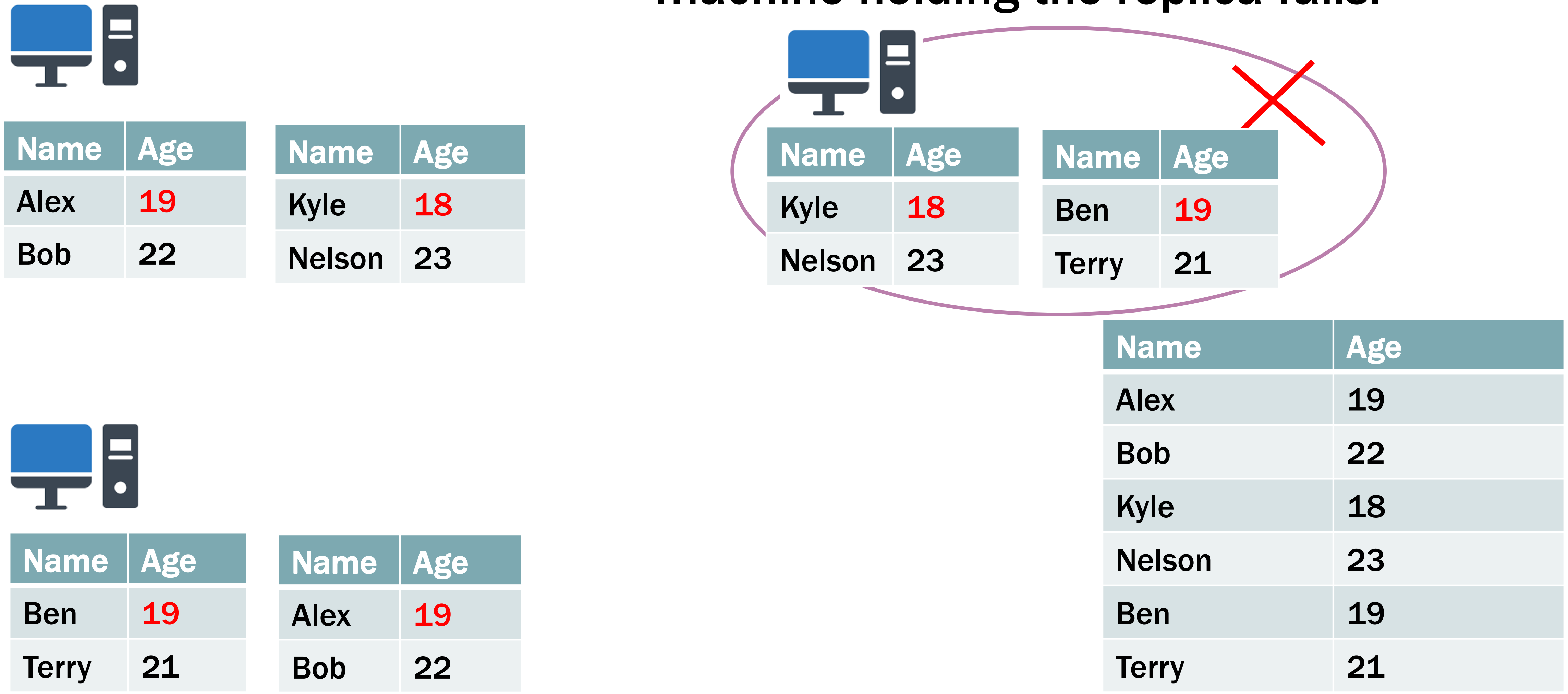
Advantages:

❑ Better fault tolerance
❑ When one machine fails, data may be restored from other replicas

- The same set of data (can be a subset of data) may be stored at different machines.

DATA REPLICATION

We can still get Kyle even when one machine holding the replica fails.



Name	Age
Alex	19
Bob	22

Name	Age
Kyle	18
Nelson	23

Name	Age
Kyle	18
Nelson	23

Name	Age
Ben	19
Terry	21

Name	Age
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22

Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

DATA REPLICATION

There is no free lunch. What is the cost of using data replication?

DATA REPLICATION

1st Disadvantage: More data in each machine, and probably a high processing cost in each machine

Name	Age
Alex	19
Bob	22

Name	Age
Kyle	18
Nelson	23

Name	Age
Kyle	18
Nelson	23

Name	Age
Ben	19
Terry	21

Name	Age
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22

Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

DATA REPLICATION: POSSIBLE SOLUTIONS

- Primary replica and secondary replicas
- During query, first access the primary replica. Only when the primary replica is not accessible, we go for one of secondary replicas.
- Maintains high query efficiency.

DATA REPLICATION

Name	Age
Alex	19
Bob	22

Name	Age
Kyle	18
Nelson	23

Name	Age
Kyle	18
Nelson	23

Name	Age
Ben	19
Terry	21

2nd disadvantage: Consistency between the replicas

Name	Age
Ben	19
Terry	21

Name	Age
Alex	19
Bob	22

Name	Age
Alex	19
Bob	22
Kyle	18
Nelson	23
Ben	19
Terry	21

DATA REPLICATION

Updating one tuple requires updating all the replicas containing the tuple, causing delays (i.e., low availability).

Other options?

STRICT CONSISTENCY AND EVENTUAL CONSISTENCY

- Modern systems consider tradeoffs between consistency and availability.

- Strict (replica) consistency
  - When an update is committed to one replica, all the other replicas must be immediately updated as well, before handling any data reads.
  - Strongly consistent between data replicas
  - Low availability

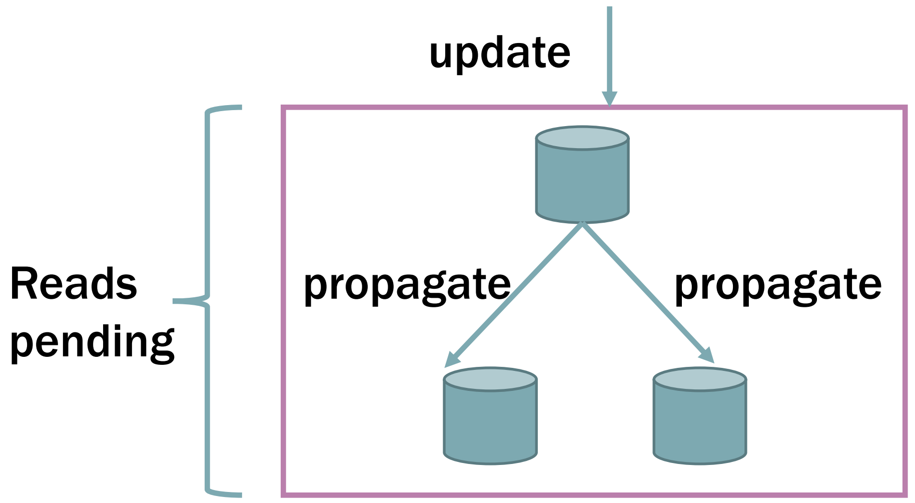
STRICT CONSISTENCY AND EVENTUAL CONSISTENCY

- Modern systems consider tradeoffs between consistency and availability.

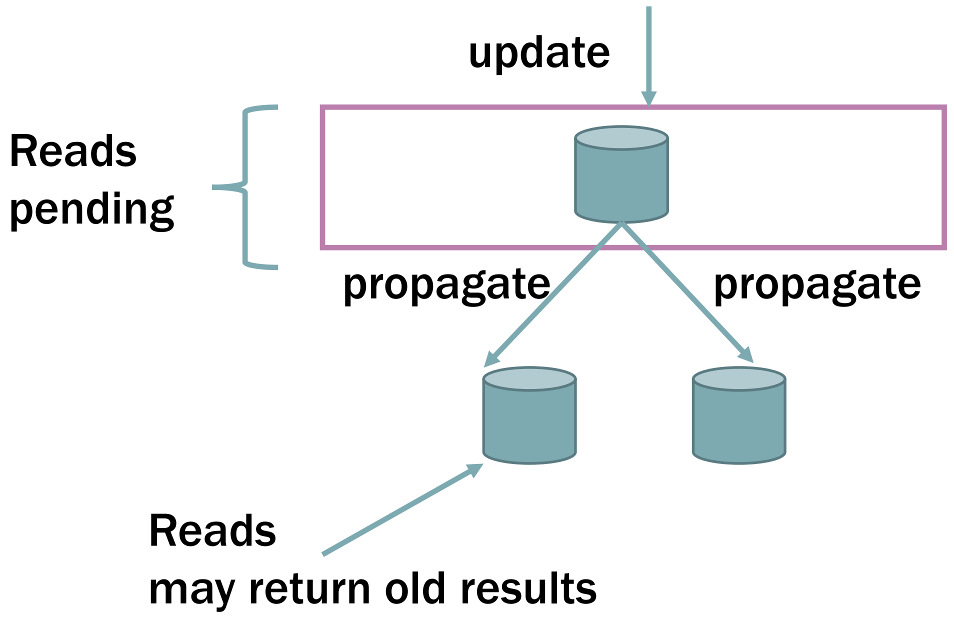
- Eventual (replica) consistency
  - Update is first committed to one replica, and once updated, the data reads are allowed. The updates will be ultimately propagated to other replicas.
  - Weak consistency
  - High availability

STRICT CONSISTENCY AND EVENTUAL CONSISTENCY

Strict consistency



Eventual consistency



REMARKS

- In distributed systems, there is a tradeoff between consistency and system availability

- Usually, bank or financial related systems may choose strict consistency; web applications which allow some inconsistency may choose eventual consistency.

DISTRIBUTED SYSTEMS FOR BIG DATA: CHALLENGES

- ❑ How to organize the machines?
- ❑ How to store data across machines?
- ❑ **How to compute using multiple machines?**

TYPICAL PROCEDURE OF DISTRIBUTED COMPUTATION

Master-Slave Model:

Step 1: data is partition to different machines

Step 2: master assigns tasks to each slave (or slaves request tasks from the master)

Step 3: slaves accomplish their own tasks locally and send results to master if needed.

Step 4: master aggregates the results and go back to Step 2 if needed.

Note: in practice, some detailed steps vary depending on the specific computation tasks.

TYPICAL PROCEDURE OF DISTRIBUTED COMPUTATION

Fully Distributed Model:

Step 1: data is partitioned to different machines

Step 2: each machine gathers the received messages (if any)

Step 3: each machine conducts local computation

Step 4: each machine sends messages to some other machines and goes to step 2 if needed.

Note: in practice, some detailed steps vary depending on the specific computation tasks.

SUMMARY OF DISTRIBUTED COMPUTATION

- ❑ Two ways of organizing machines
  - ❑ Fully distributed
  - ❑ Master-Slave

- ❑ Two data storage methods
  - ❑ Data partition
  - ❑ Data replication

- ❑ Two typical ways of distributed computation

MapReduce:
A general distributed paradigm

DISTRIBUTED COMPUTATION PARADIGM

"The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues."

Lead of Google AI

PROGRAMING WITH DISTRIBUTED MACHINES

- Programing with distributed machines is hard
- Compare with the codes to run in a single machine, coding for distributed systems requires
  - Handling communication between machines
  - Coping with machine failures
  - Data replicas
  - Data partitioning
  - …

Coding from scratch for distributed
Machines is painful, even for simple computation logics.