

# GFS and HDFS

*DS 5110: Big Data Systems*

*Spring 2025*

Lecture 7a

Yue Cheng



Some material taken/derived from:

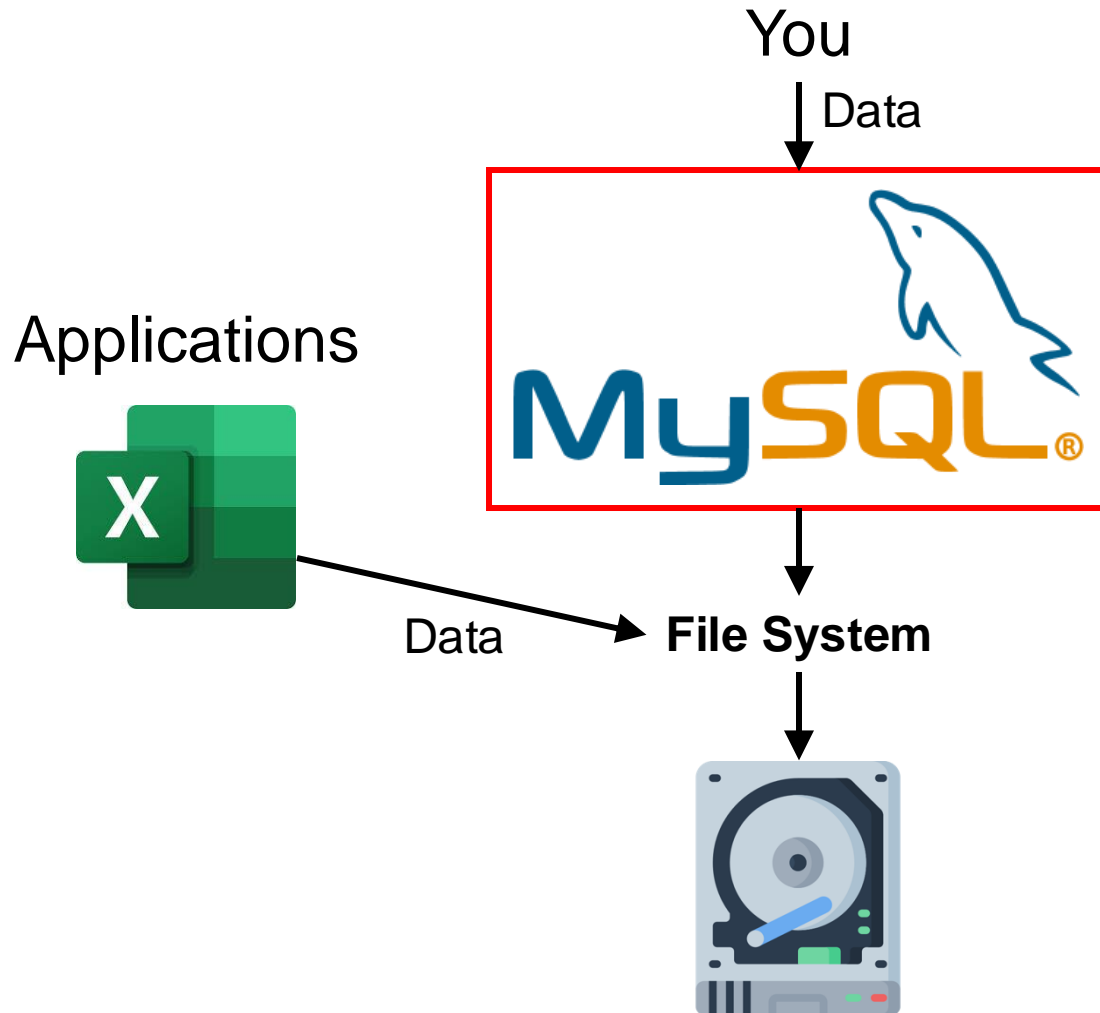
• Wisconsin CS 320 by Tyler Caraza-Harter.

@ 2025 released for use under a [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

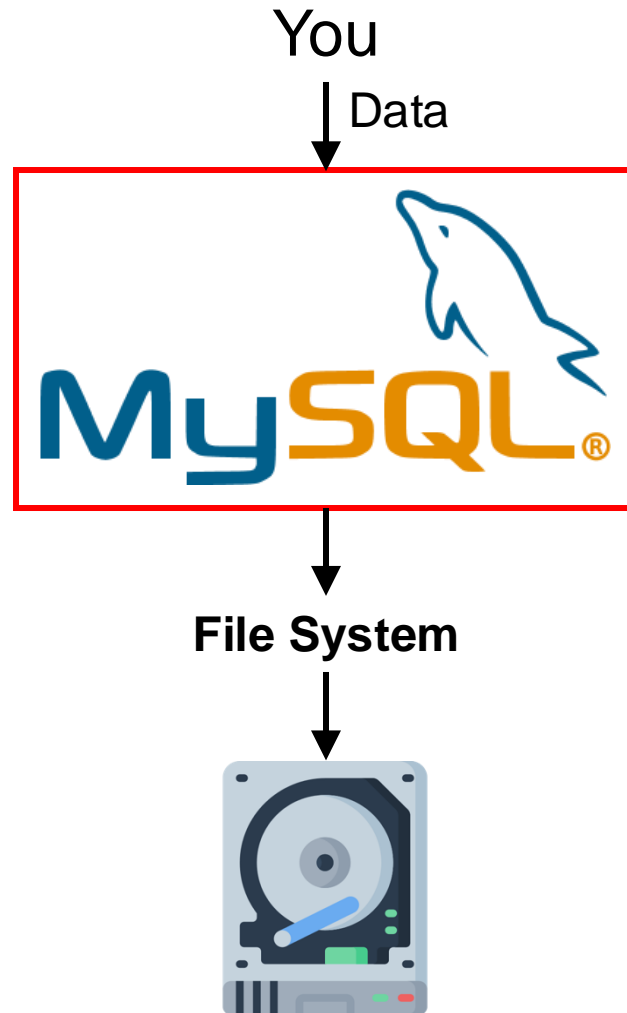
# Learning objectives

- Describe the design of GFS (HDFS)
- Understand partitioning, replication, and the motivation of each technique
- Identify the role that clients, NameNode, DataNodes play for HDFS reads and writes

# Design: Storage systems are generally built as a composition of layered subsystems

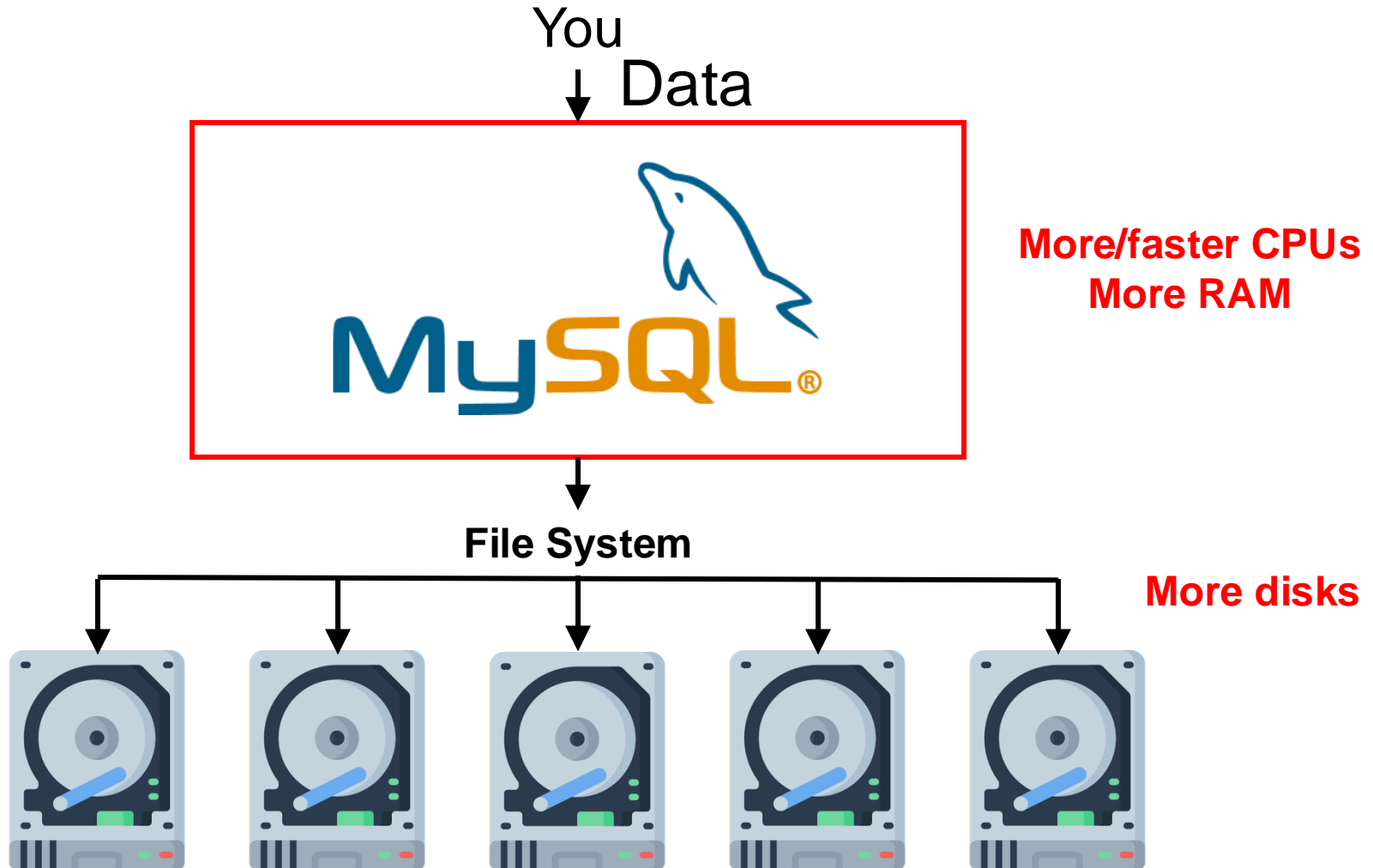


# Problem: What if your data is too big?



# Problem 1: What if your data is too big?

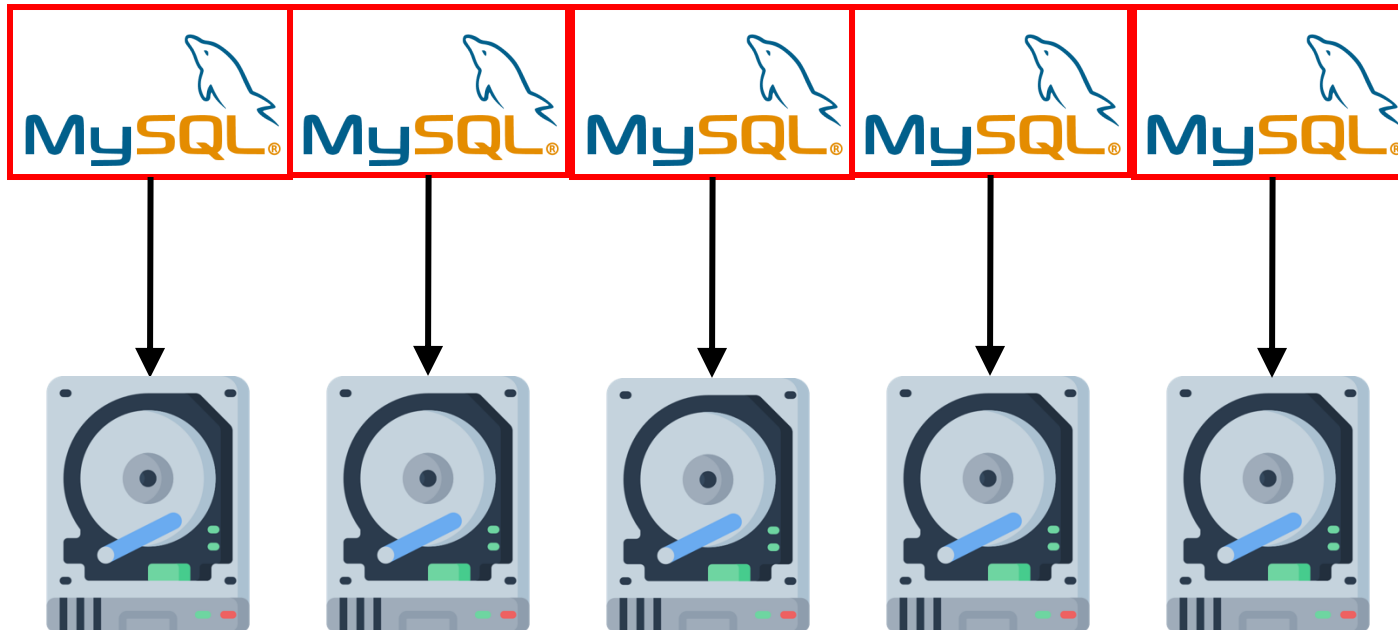
## Option 1: **Scale up** (buy better hardware)



# Problem 1: What if your data is too big?

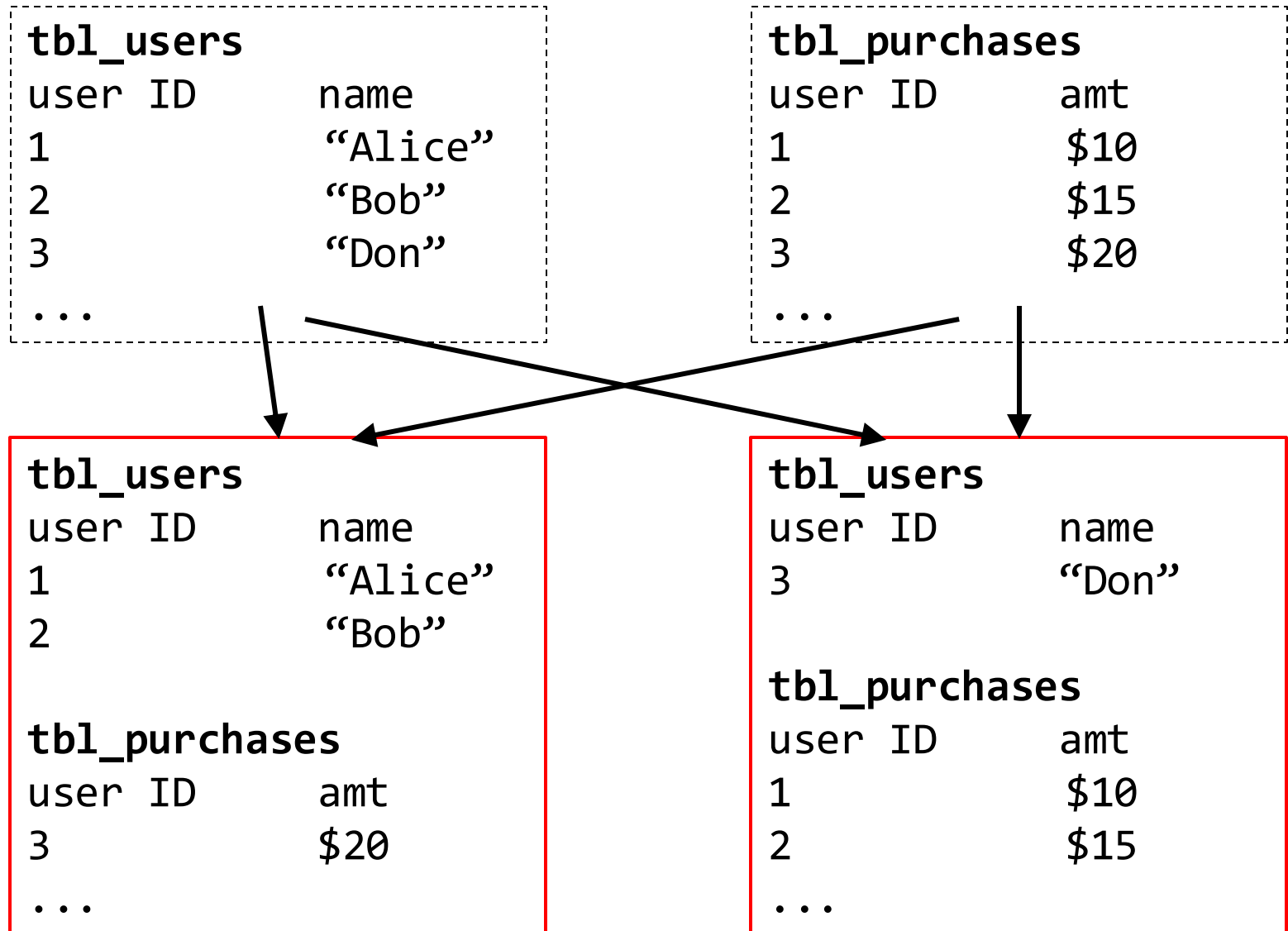
## Option 2: **Scale out** (more machines)

Where does the data actually go?



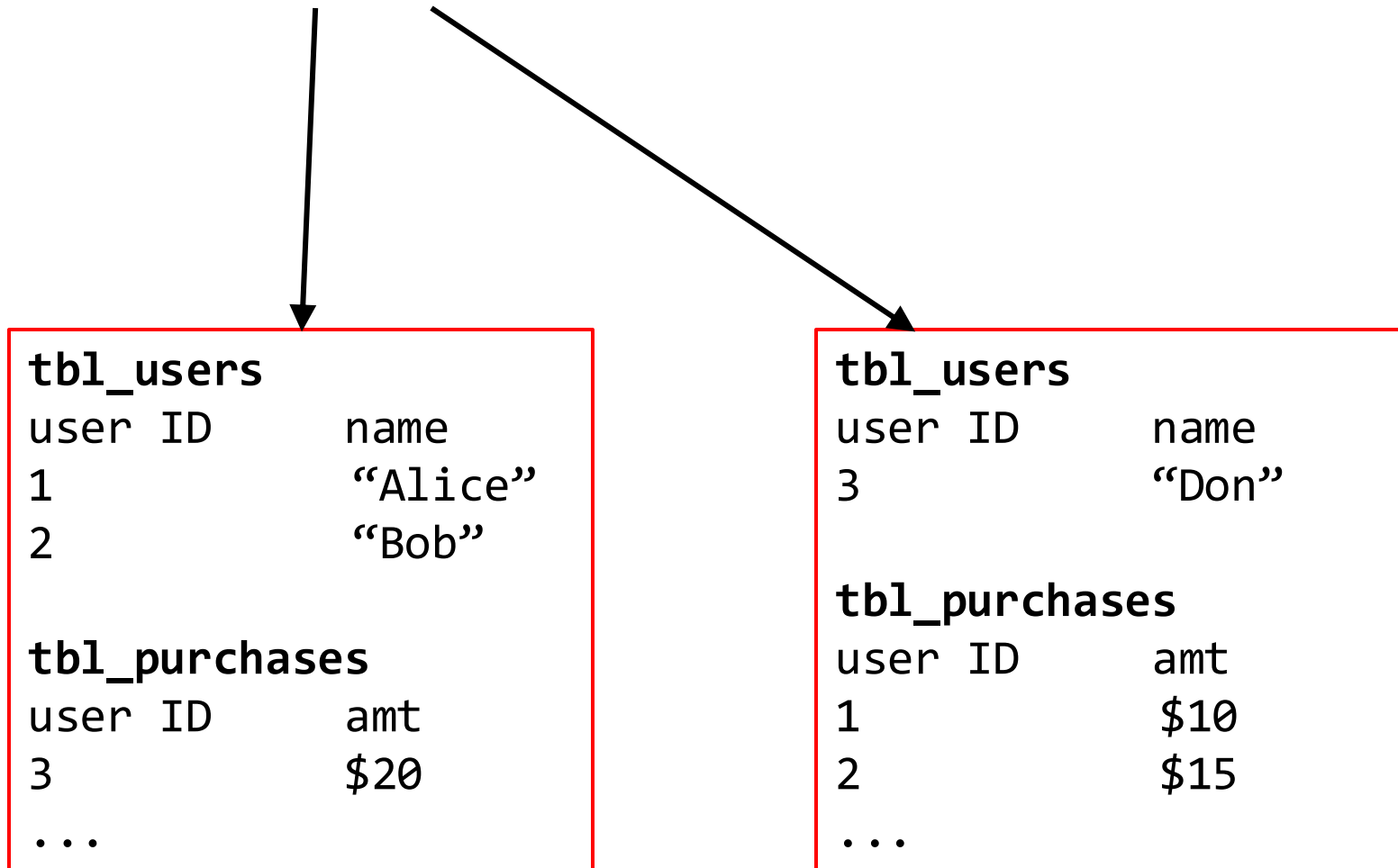
...1000 more...

# Approach: Partition the data



# Approach: Send queries to multiple DBs

```
SELECT * FROM tbl_purchases WHERE amt > 12
```





# ... Combine results

```
SELECT * FROM tbl_purchases WHERE amt > 12
```

| tbl_purchases |      |
|---------------|------|
| user ID       | amt  |
| 2             | \$15 |
| 3             | \$20 |

| tbl_users |         |
|-----------|---------|
| user ID   | name    |
| 1         | "Alice" |
| 2         | "Bob"   |

| tbl_purchases |      |
|---------------|------|
| user ID       | amt  |
| 3             | \$20 |
| ...           |      |

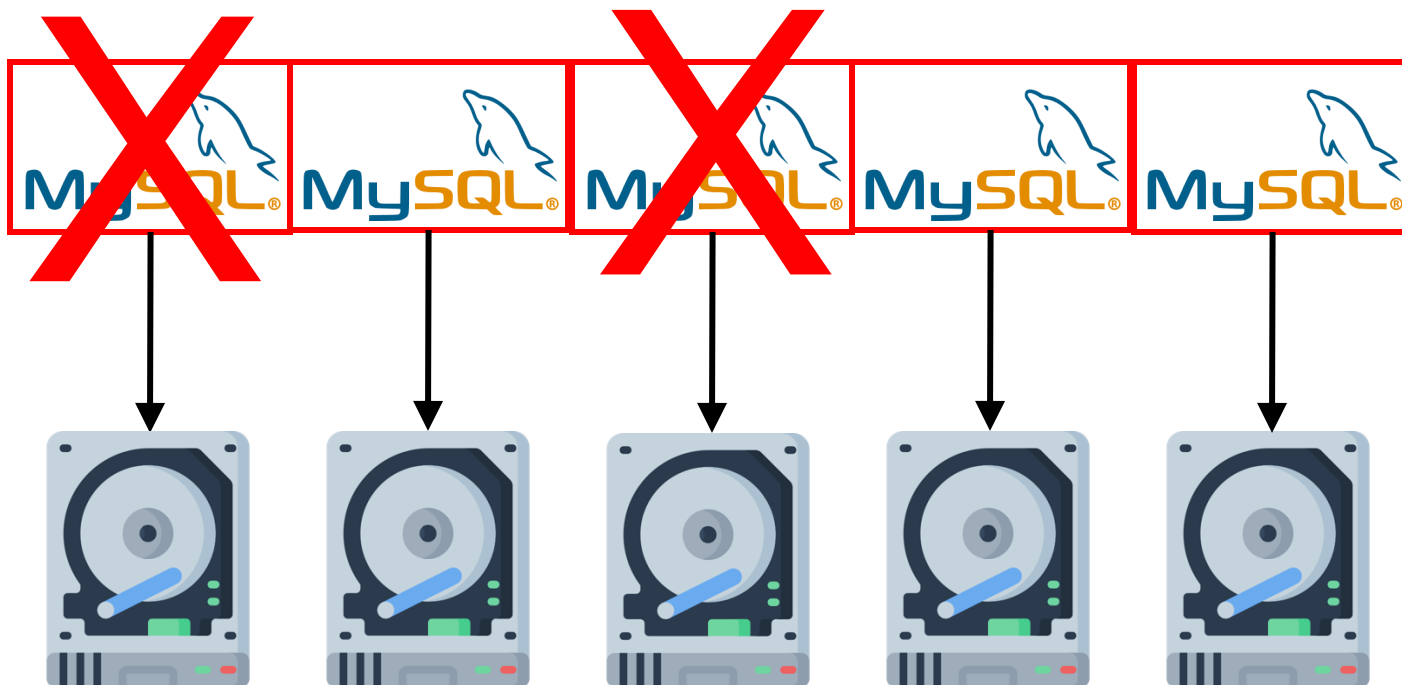
| tbl_users |       |
|-----------|-------|
| user ID   | name  |
| 3         | "Don" |

| tbl_purchases |      |
|---------------|------|
| user ID       | amt  |
| 1             | \$10 |
| 2             | \$15 |
| ...           |      |

# Problem 2: What if your server dies?

Happens all the time when you have 1000s of machines...

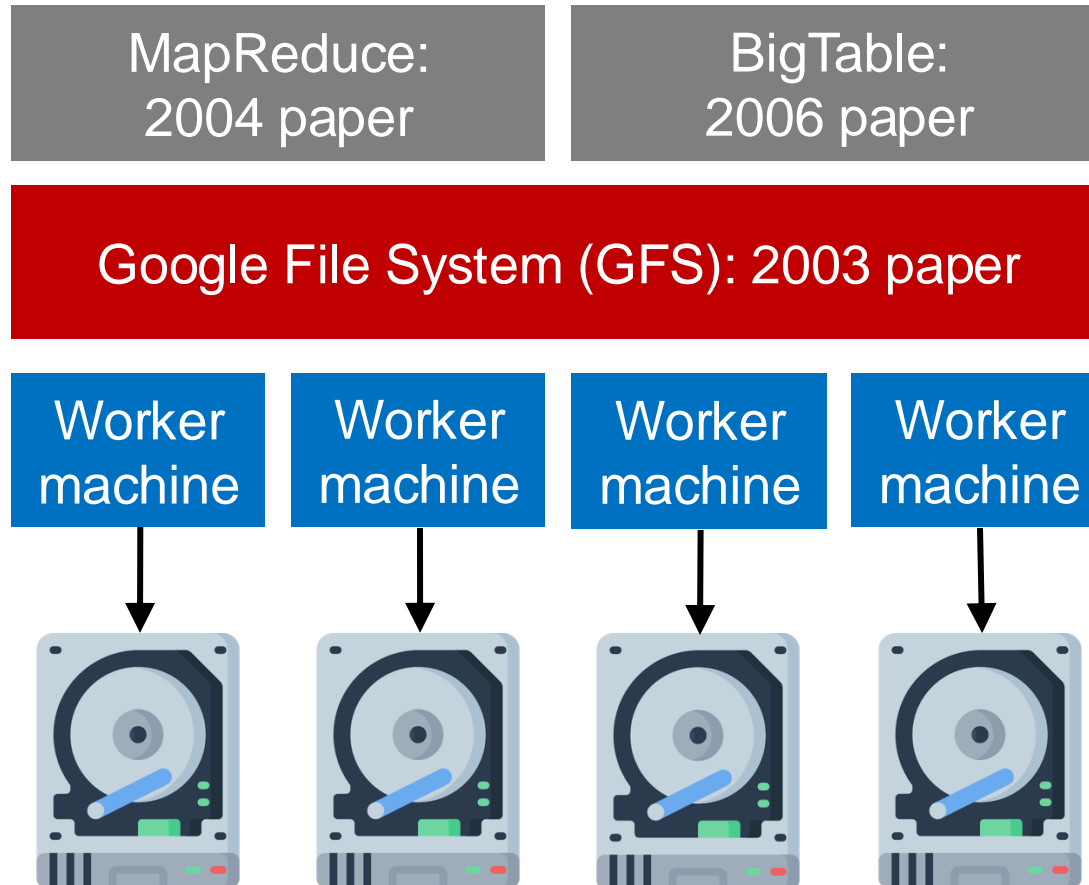


...1000 more...

# Motivation for large DFS (GFS / HDFS)

- Scaling to many machines is essential
- Fault tolerance is essential

# Google big data infrastructure



Radical idea: base everything on lots of cheap, commodity hardware

# Hadoop ecosystem

Yahoo, Facebook, Cloudera, and others developed open-source Hadoop ecosystem, mirroring Google's big data systems

|   | <b>Google<br/>(paper only)</b> | <b>Hadoop<br/>(open source)</b> | <b>Modern<br/>Hadoop</b> |
|---|--------------------------------|---------------------------------|--------------------------|
| <b>Distributed File System</b>                | GFS                            | HDFS                            |                          |
| <b>Distributed Processing &amp; Analytics</b> | MapReduce                      | Hadoop MapReduce                | Spark                    |
| <b>Distributed Database</b>                   | BigTable                       | HBase                           | MongoDB                  |

<https://hadoop.apache.org/>

# HDFS: DataNodes store file blocks

F1: "ABCD"

F2: "EFGHIJKL"

Some file fits in a single block



**DataNode  
Computers**



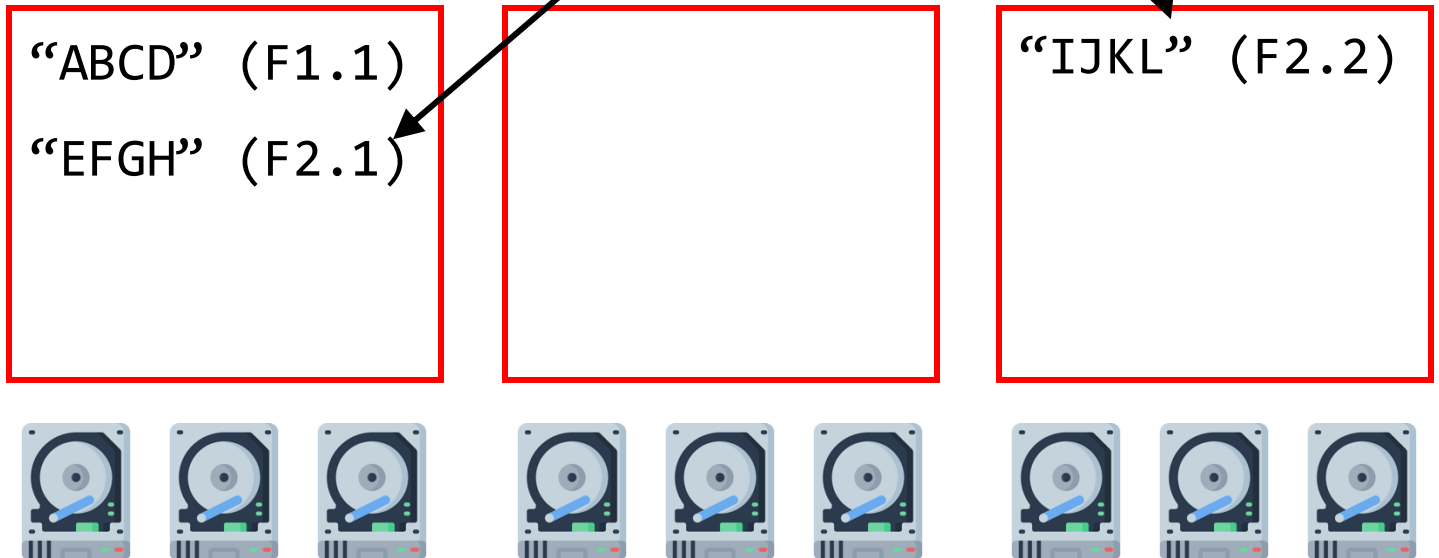
# HDFS: Partitioning across DataNodes

F1: "ABCD"

F2: "EFGHIJKL"

Bigger files are **partitioned**  
across multiple DataNodes

DataNode  
Computers



# HDFS: Replication across DataNodes

F1: "ABCD"

3x replication

F2: "EFGHIJKL"

2x replication

DataNode  
Computers

"ABCD" (F1.1)

"EFGH" (F2.1)

"ABCD" (F1.1)

"IJKL" (F2.2)

"IJKL" (F2.2)

"ABCD" (F1.1)

"EFGH" (F2.1)





# HDFS: Replication across DataNodes

F1: "ABCD"

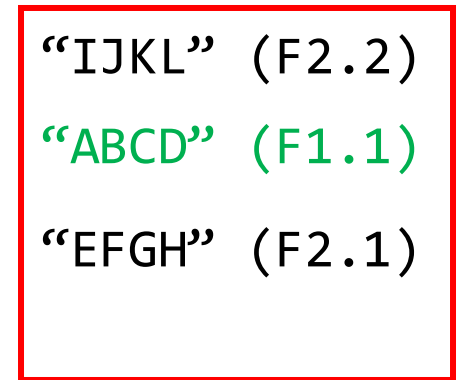
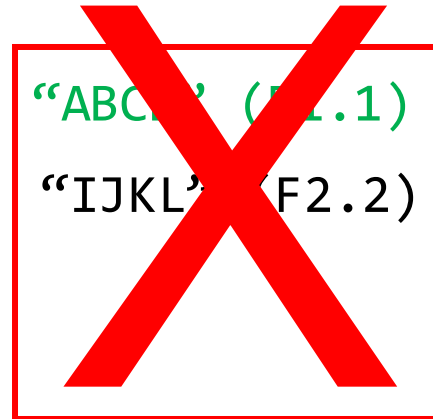
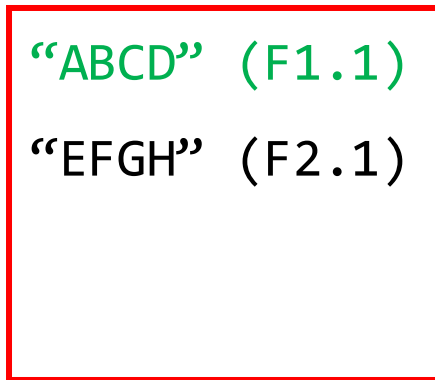
3x replication

F2: "EFGHIJKL"

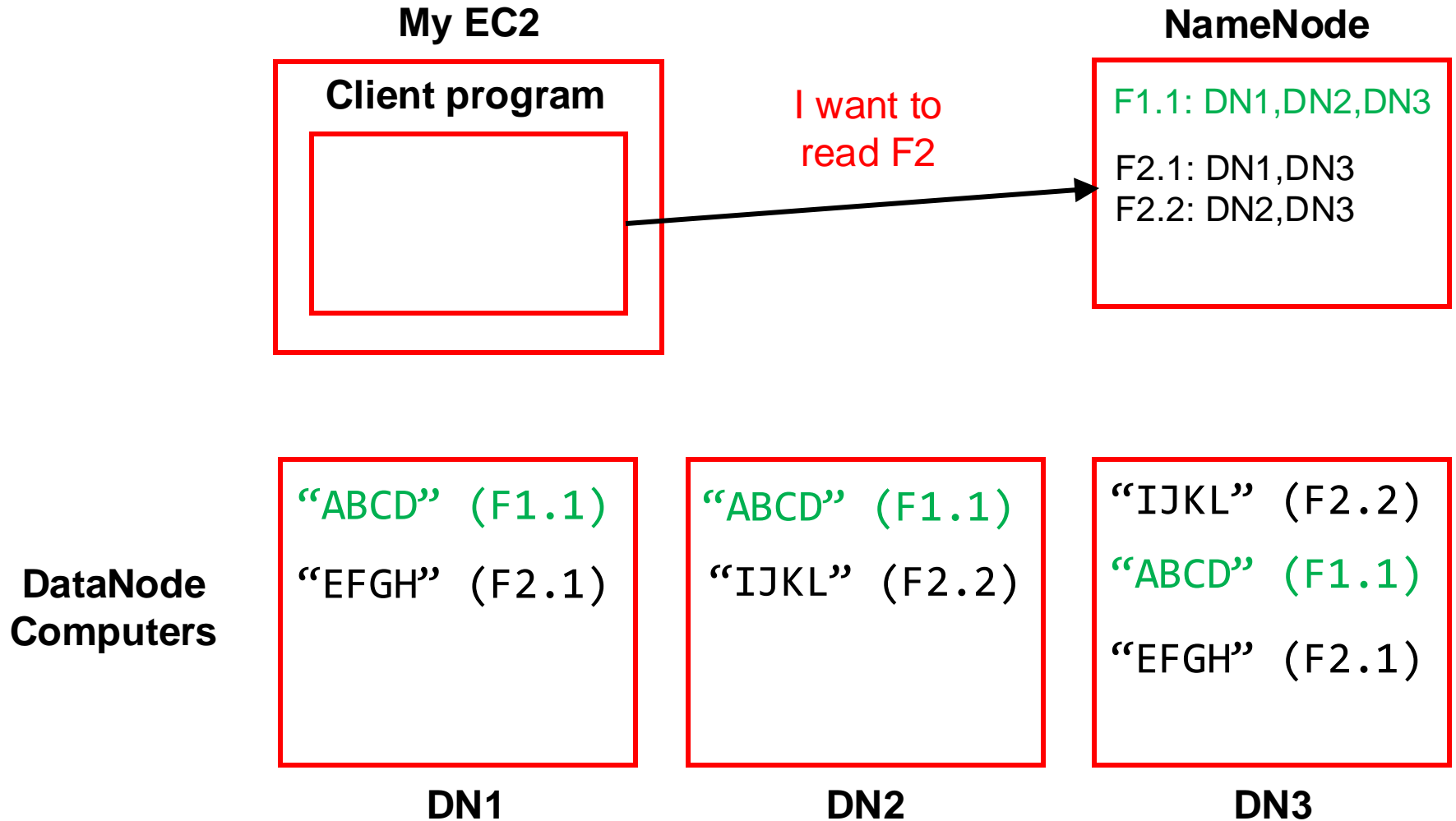
2x replication

If a DataNode dies, we still have all the data.  
**Which file is safer in general? F1 or F2?**

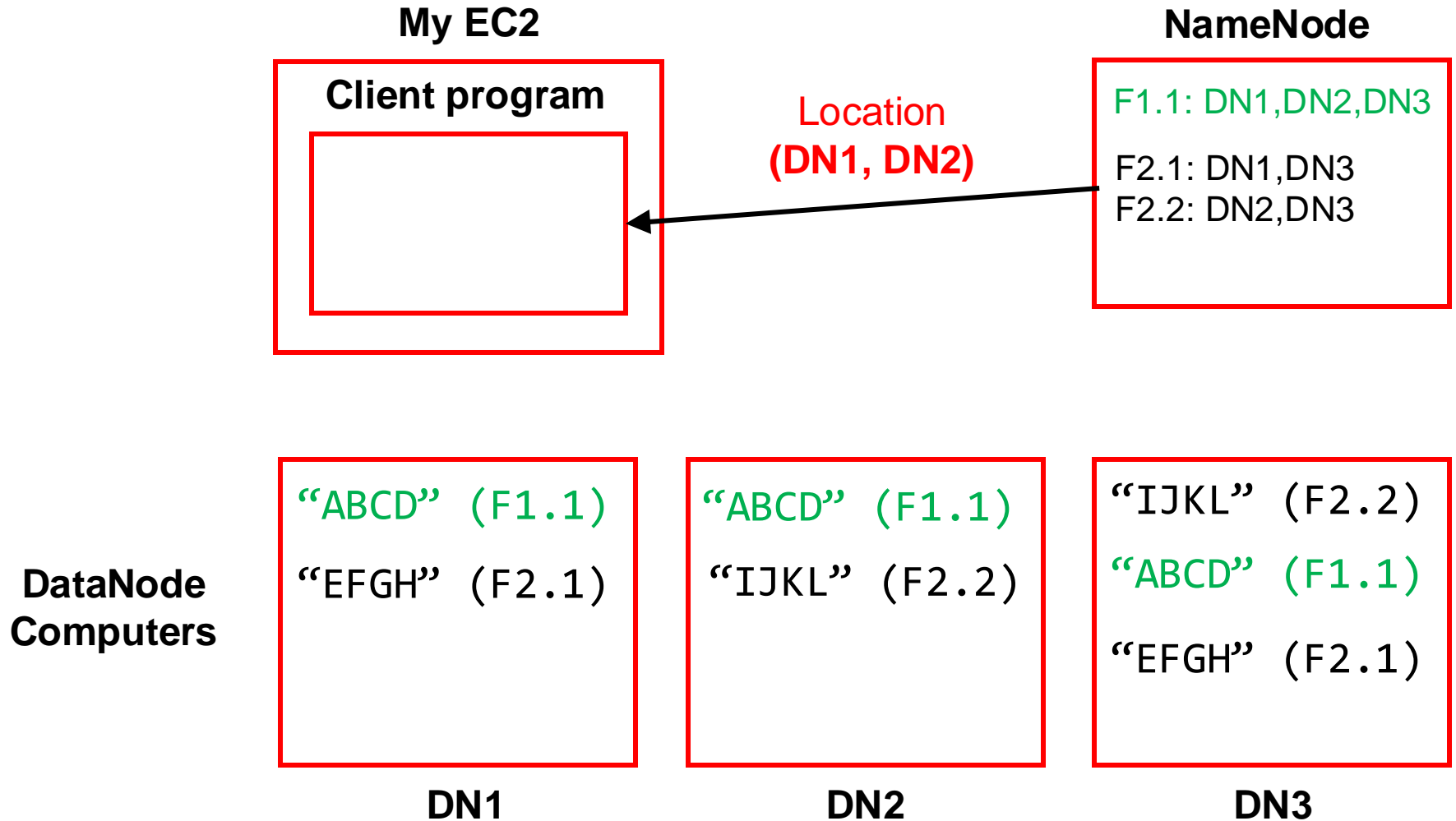
DataNode  
Computers



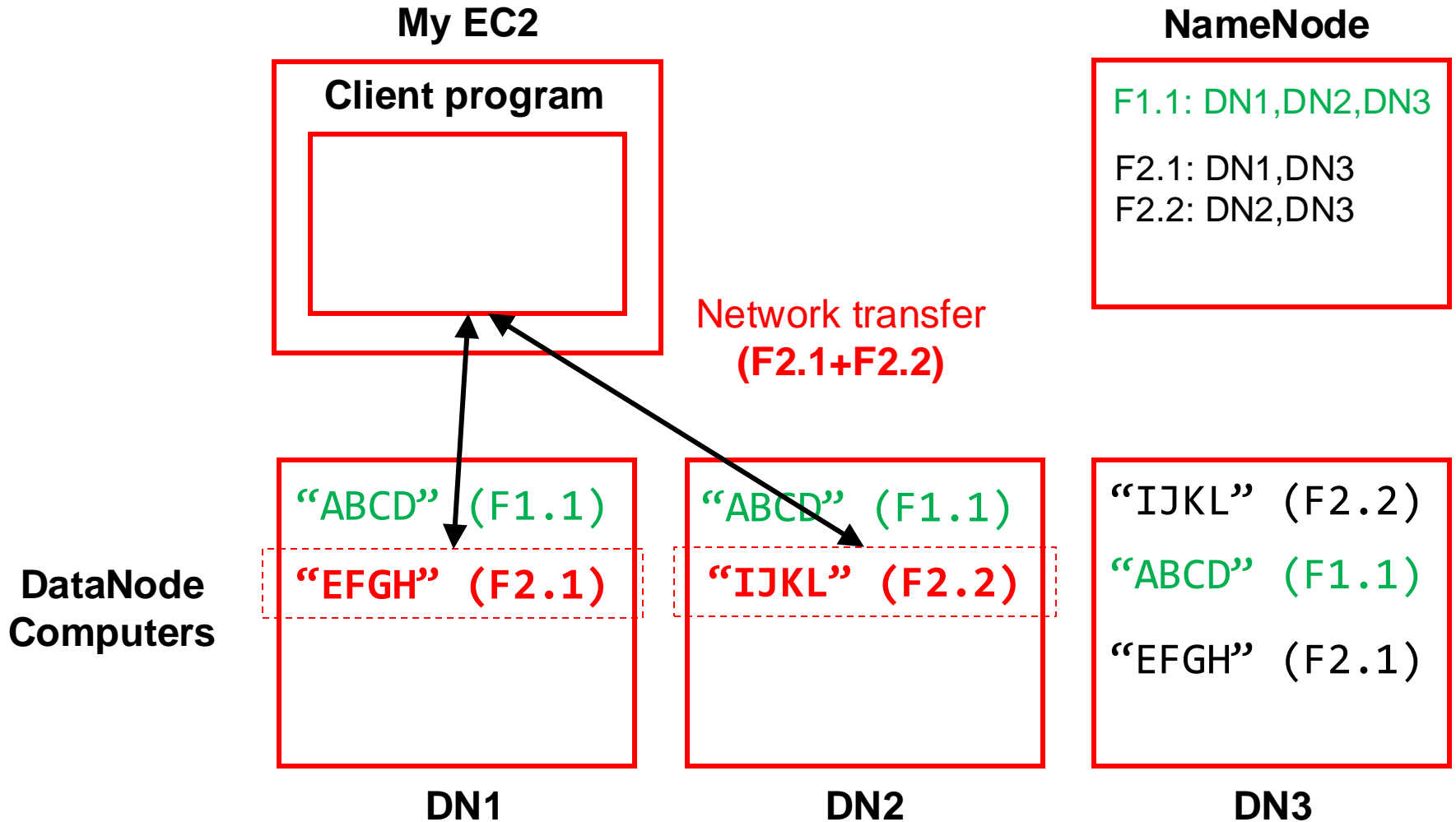
# NameNode/Worker architecture



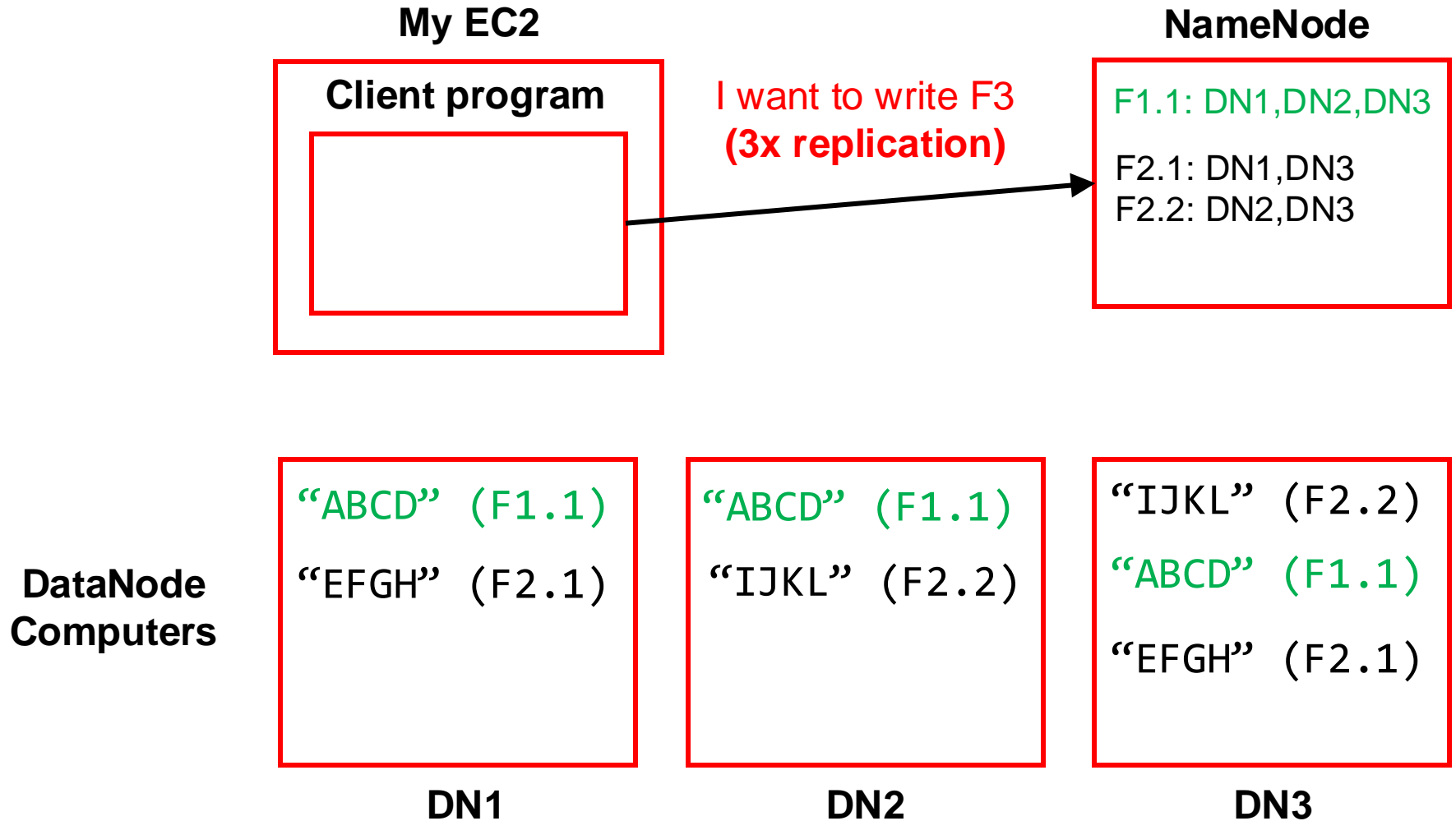
# NameNode/Worker architecture



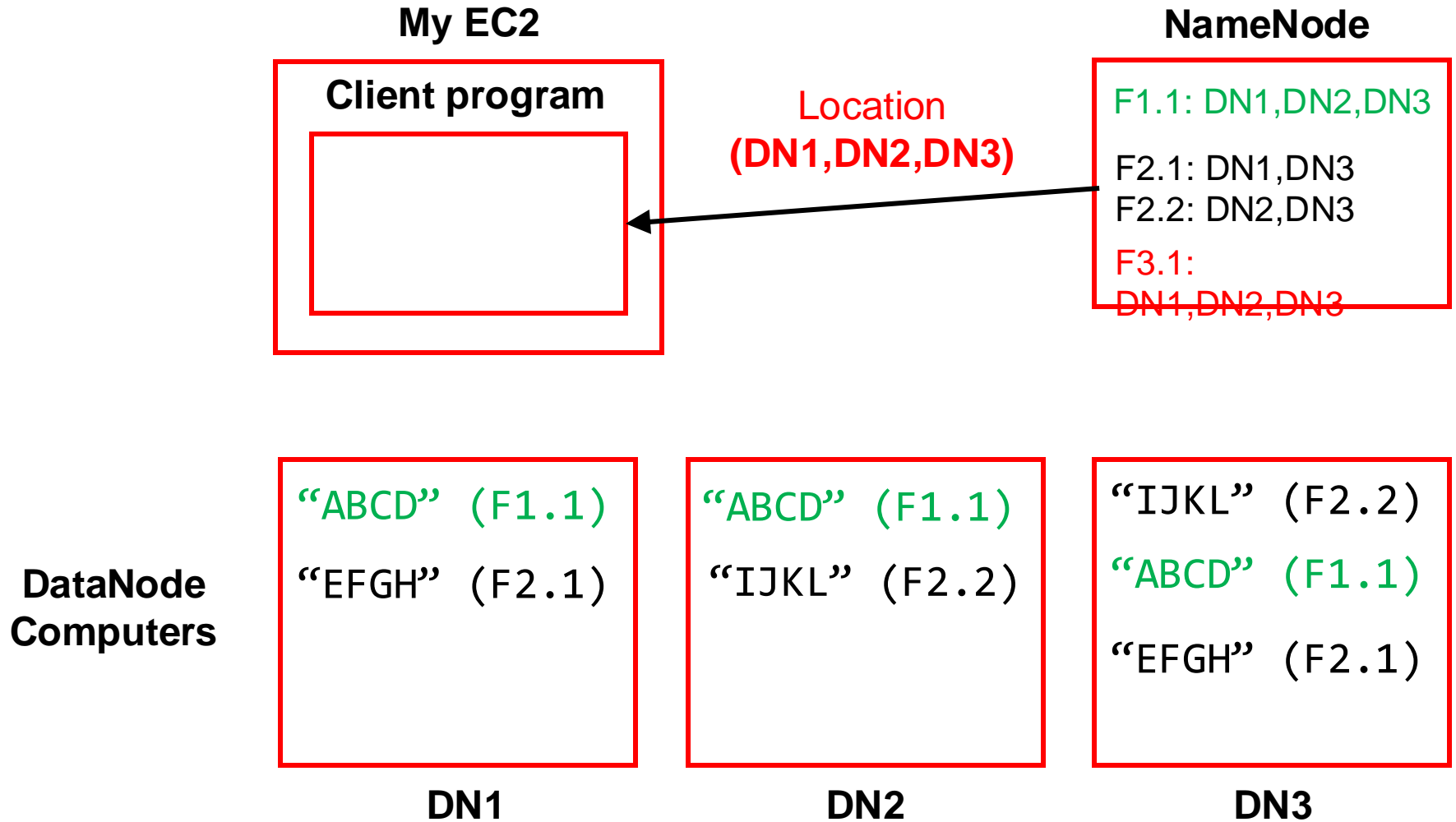
# NameNode/Worker architecture



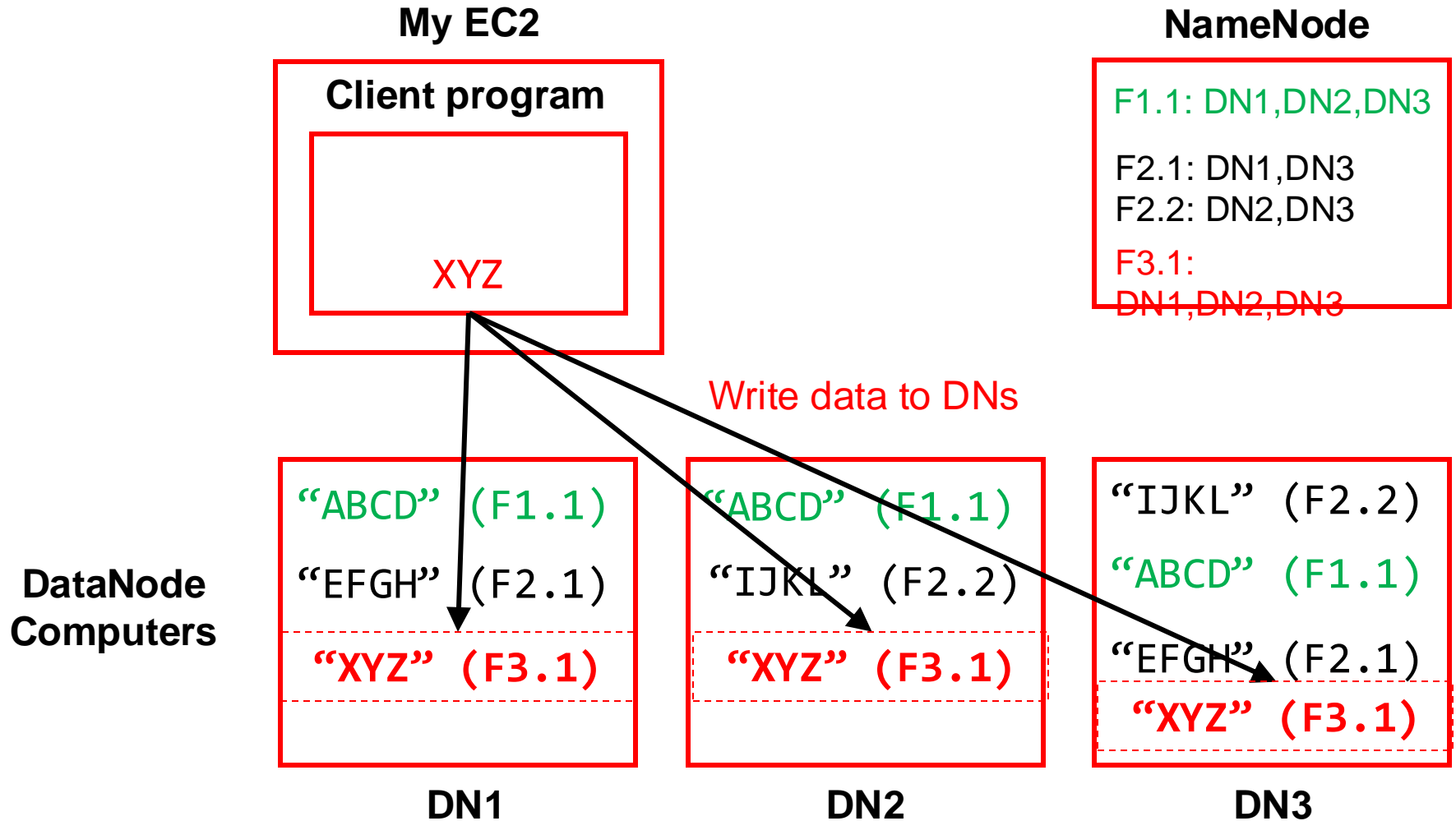
# NameNode/Worker architecture



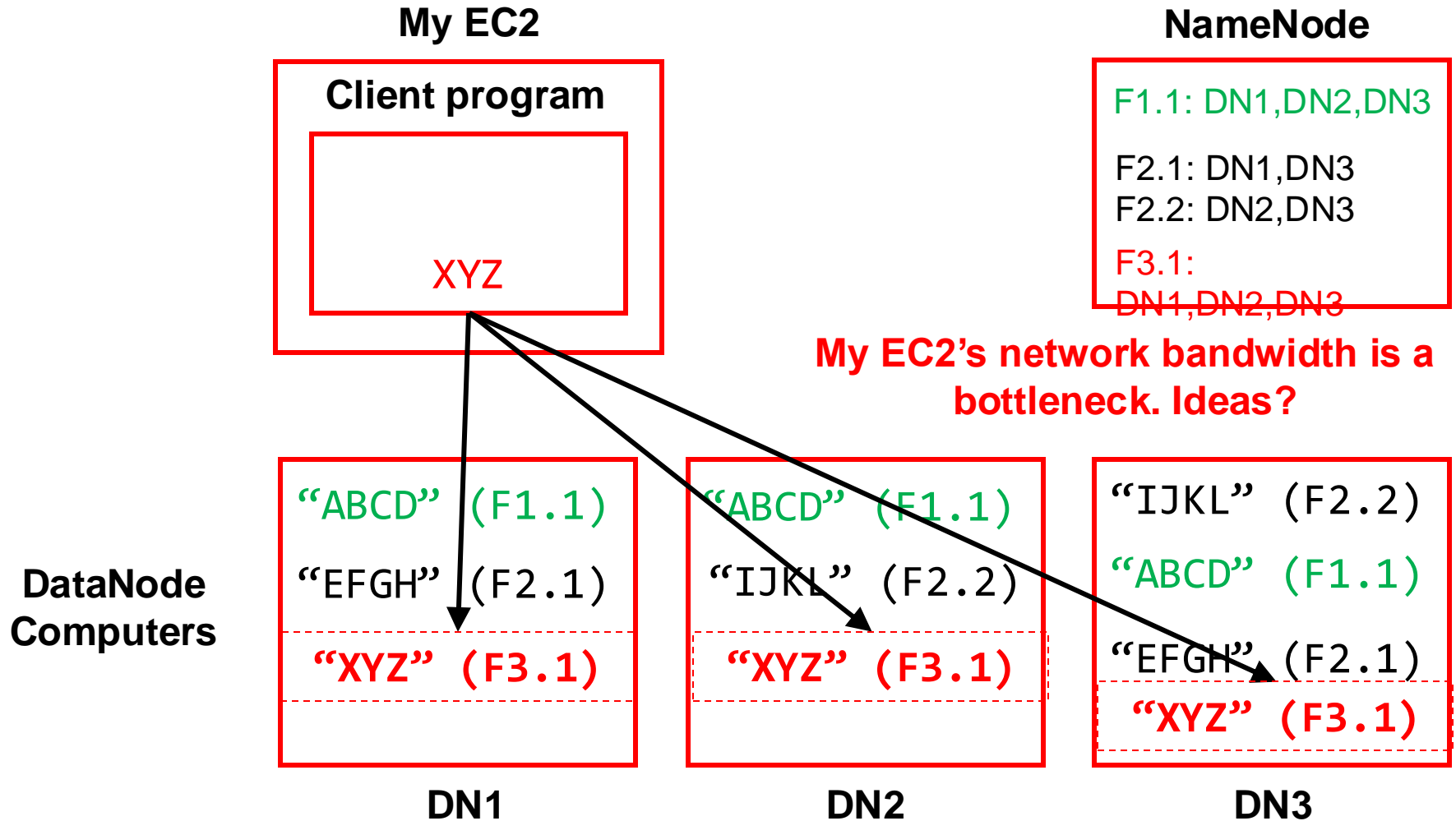
# NameNode/Worker architecture



# NameNode/Worker architecture

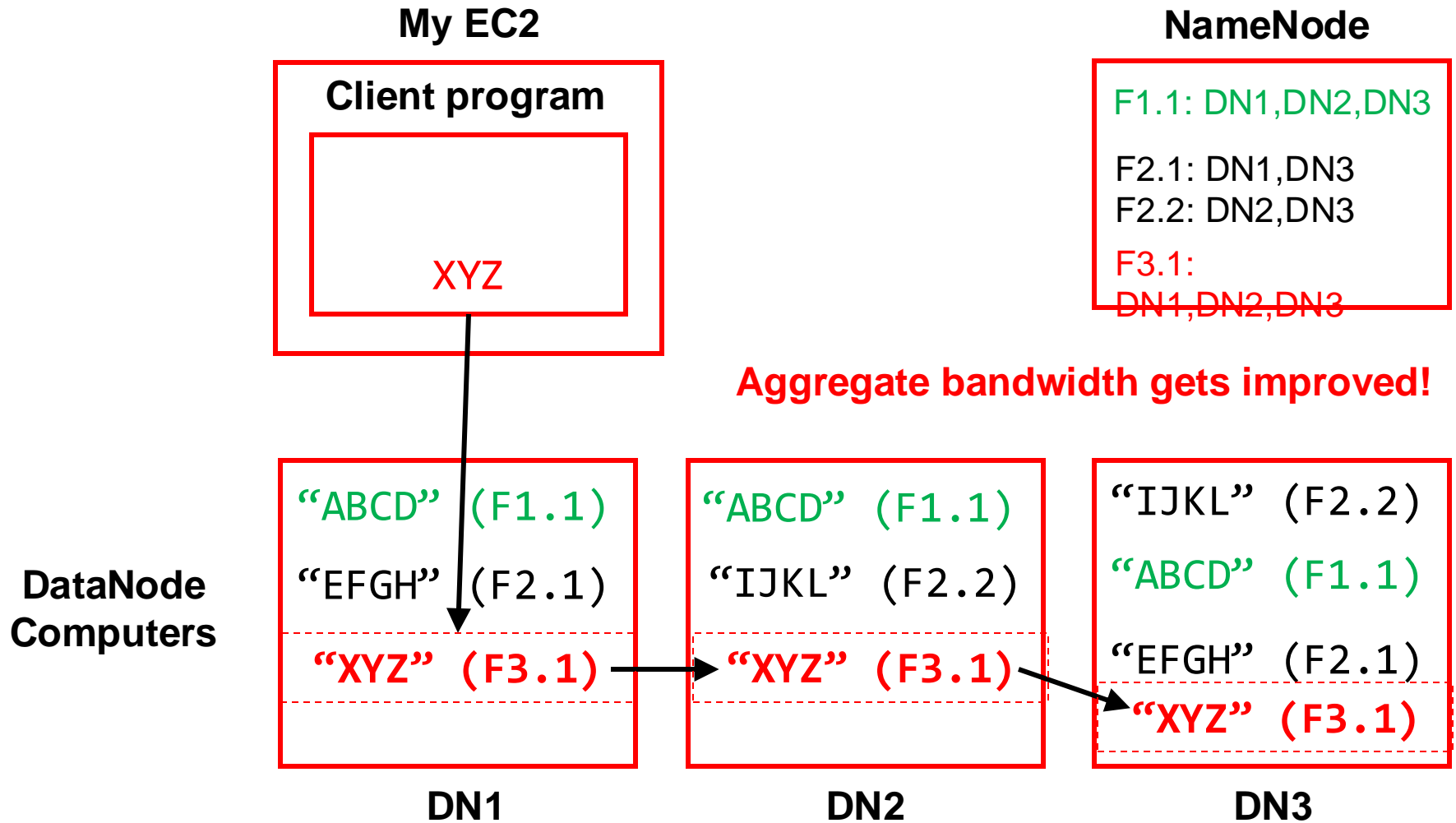


# NameNode/Worker architecture





# Pipelined writes



# How are reads/writes amplified at disk level?

Q1: If a client **writes** 4MB to a 2x replicated file, how much data does HDFS **write** to disks?

Q2: If a client **reads** 2MB from a 3x replicated file, how much data do we **read** from disks?

## NameNode

F1.1: DN1, DN2, DN3

F2.1: DN1, DN3

F2.2: DN2, DN3

F3.1:

~~DN1, DN2, DN3~~

## DataNode Computers

“ABCD” (F1.1)

“EFGH” (F2.1)

“XYZ” (F3.1)

DN1

“ABCD” (F1.1)

“IJKL” (F2.2)

“XYZ” (F3.1)

DN2

“IJKL” (F2.2)

“ABCD” (F1.1)

“EFGH” (F2.1)

“XYZ” (F3.1)

DN3

# What are the tradeoffs of replication factor and block size?

Benefit of high replication?

Benefit of low replication?

Benefit of large block size?

Benefit of small block size?

NameNode

F1.1: DN1, DN2, DN3

F2.1: DN1, DN3

F2.2: DN2, DN3

F3.1:

DN1, DN2, DN3

DataNode  
Computers

“ABCD” (F1.1)

“EFGH” (F2.1)

“XYZ” (F3.1)

DN1

“ABCD” (F1.1)

“IJKL” (F2.2)

“XYZ” (F3.1)

DN2

“IJKL” (F2.2)

“ABCD” (F1.1)

“EFGH” (F2.1)

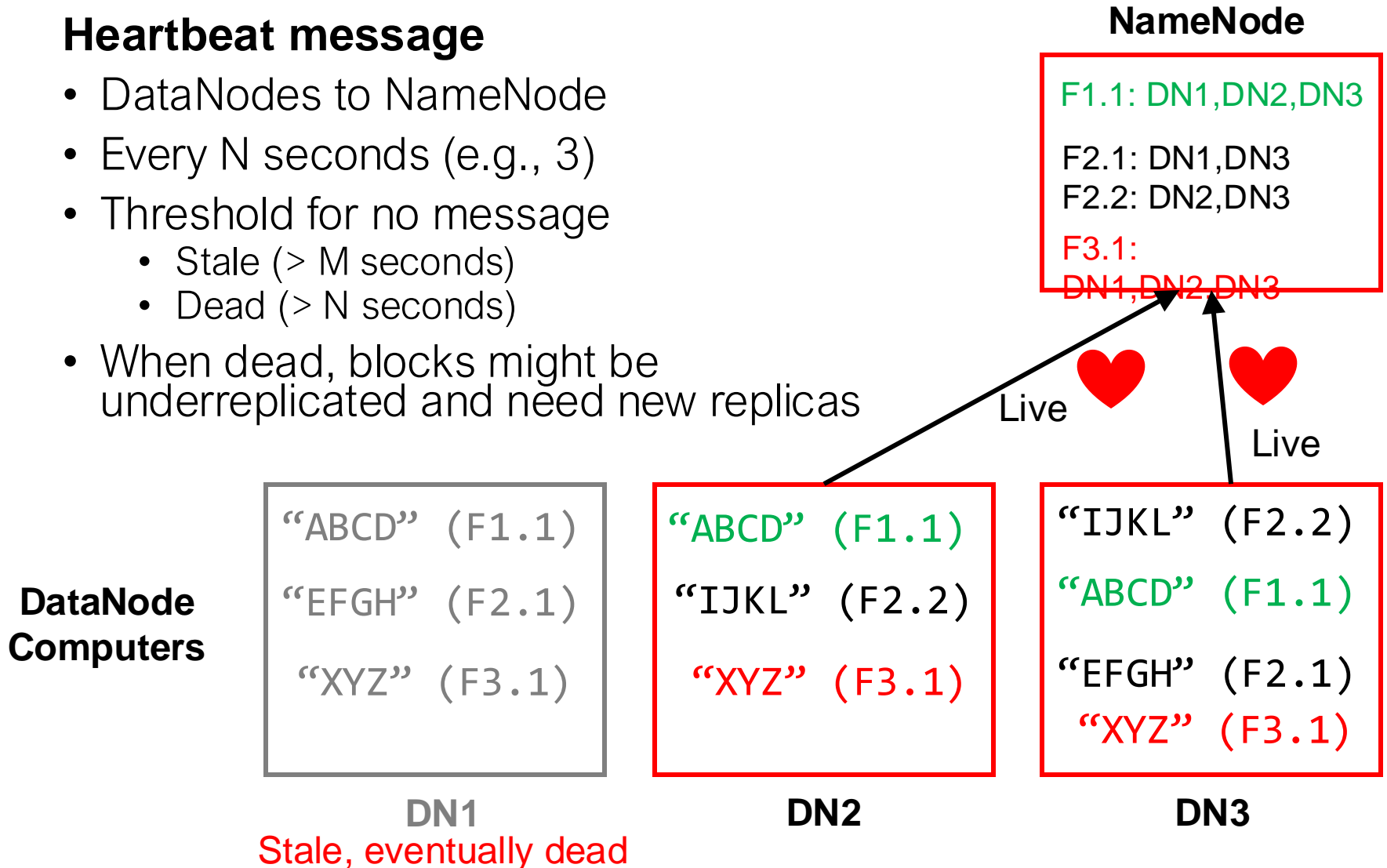
“XYZ” (F3.1)

DN3

# How do we know when a DataNode fails?

## Heartbeat message

- DataNodes to NameNode
- Every N seconds (e.g., 3)
- Threshold for no message
  - Stale (> M seconds)
  - Dead (> N seconds)
- When dead, blocks might be underreplicated and need new replicas



# Summary: Some key ideas applied to GFS/HDFS

- To build complex systems...
- To scale out...
- To handle faults...
- To detect faults...
- To optimize I/O...

# Discussion: GFS eval (GFS paper)

List your takeaways from “Fig 3: Aggregate Throughput”

