GFS and HDFS

DS 5110/CS 5501: Big Data Systems

Spring 2024

Lecture 4a

Yue Cheng

University of Virginia

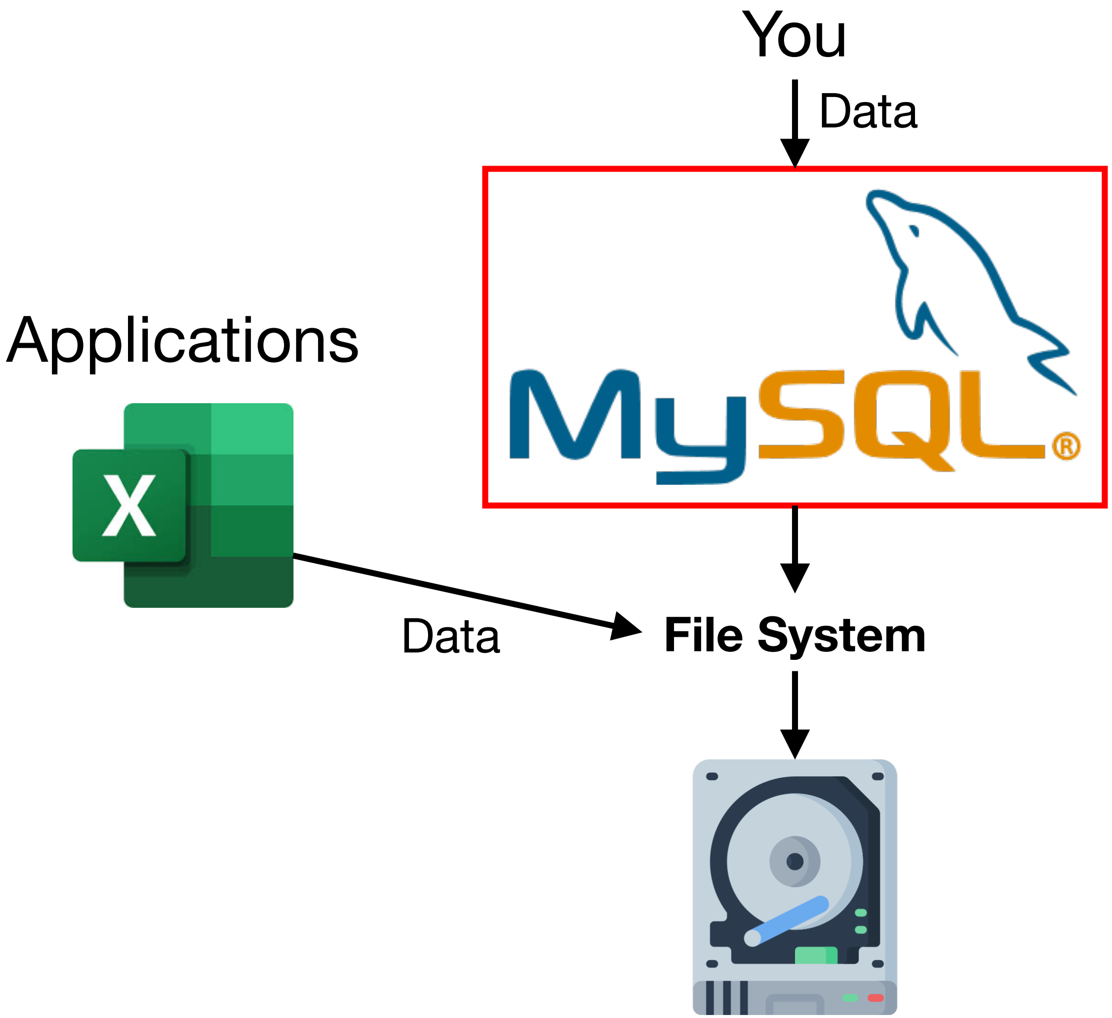
Some material taken/derived from:

- Wisconsin CS 320 by Tyler Caraza-Harter.

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



You

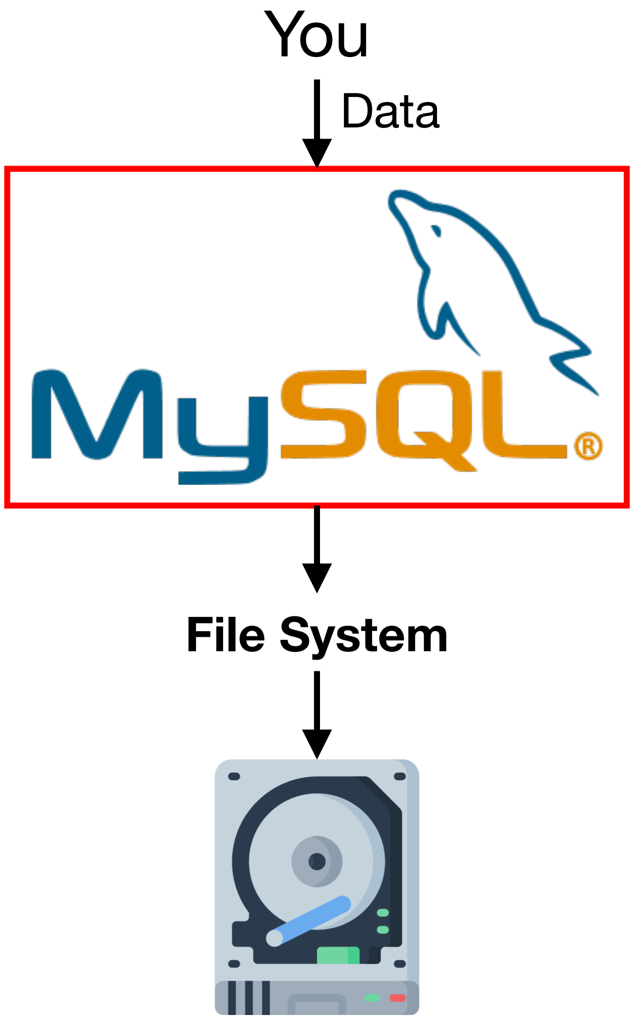
Data

Applications

Data

File System

Problem: What if your data is too big?

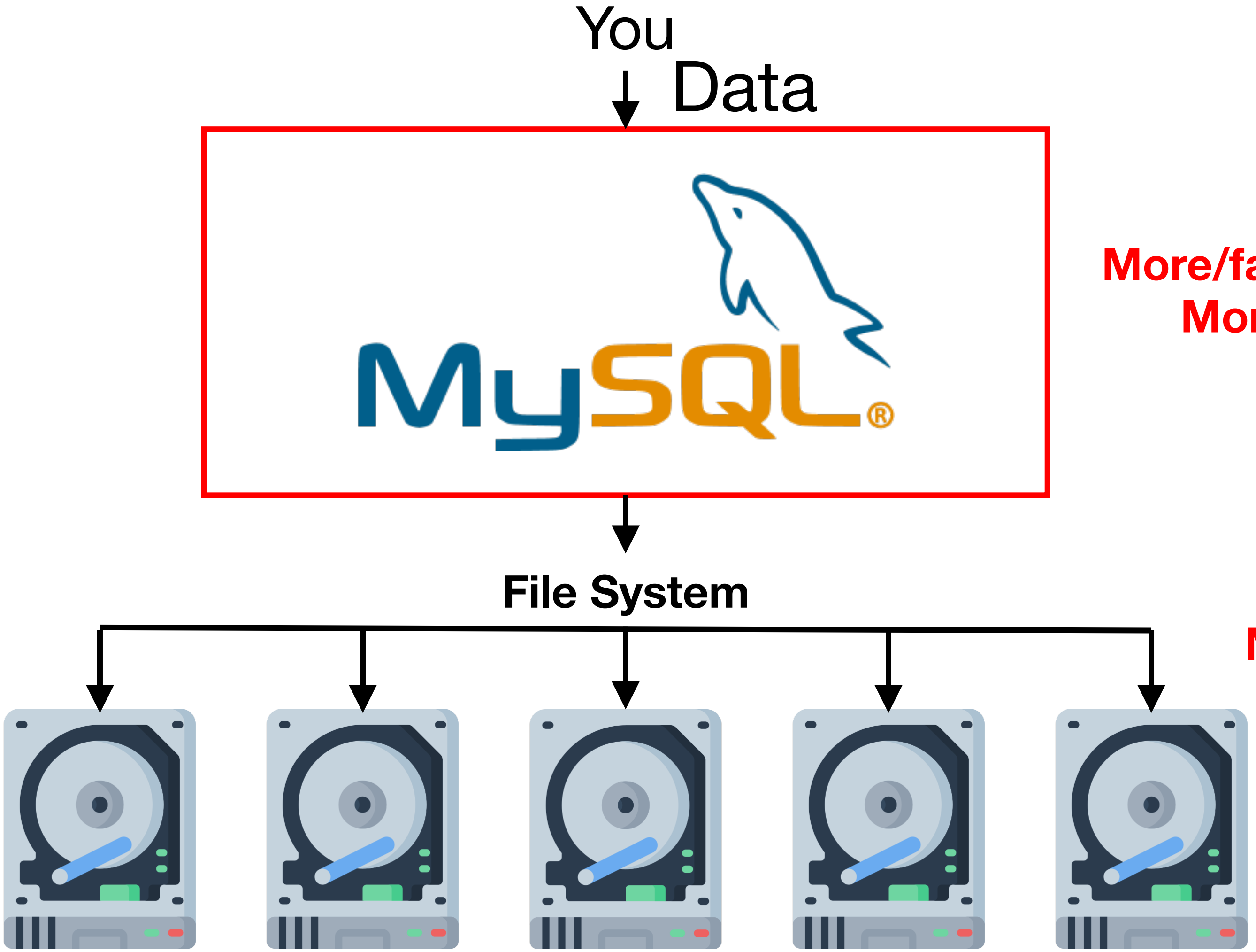


You

Data

File System

Problem 1: What if your data is too big? Option 1: Scale up (buy better hardware)

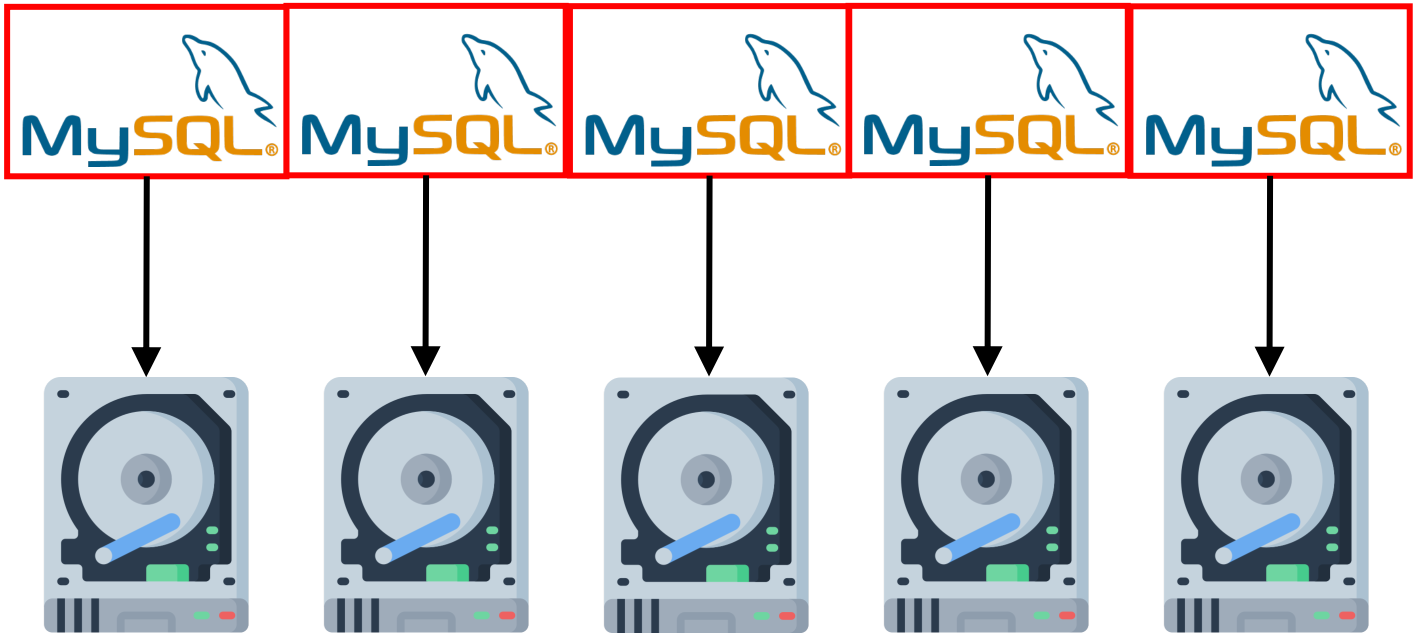


More/faster CPUs
More RAM

More disks

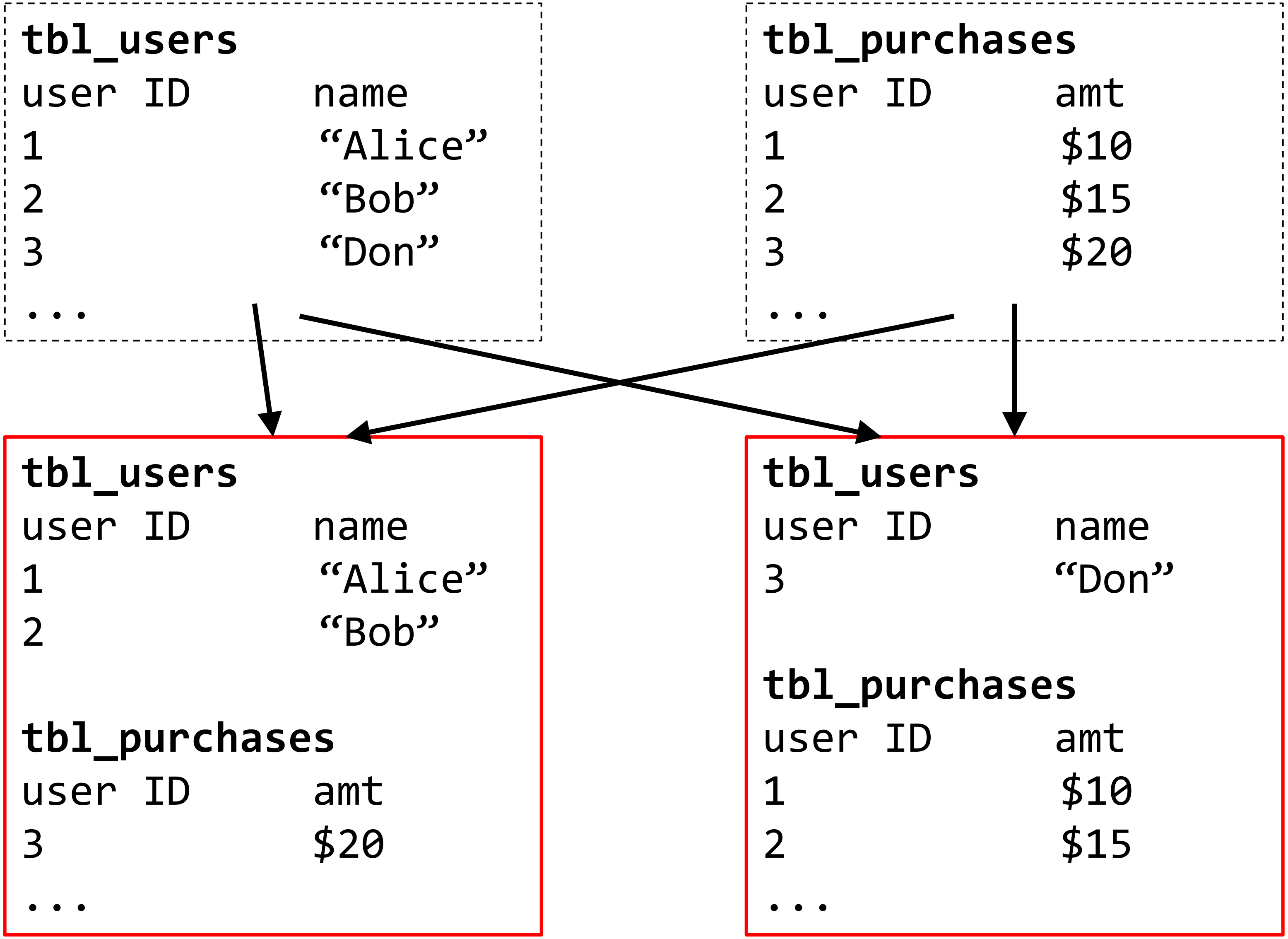
Problem 1: What if your data is too big?
Option 2: Scale out (more machines)

Where does the data actually go?



...1000 more...

Approach: Partition the data



tbl_users

user ID	name
1	“Alice”
2	“Bob”
3	“Don”
...	

tbl_purchases

user ID	amt
1	$10
2	$15
3	$20
...	

tbl_users

user ID	name
1	“Alice”
2	“Bob”

tbl_purchases

user ID	amt
3	$20
...	

tbl_users

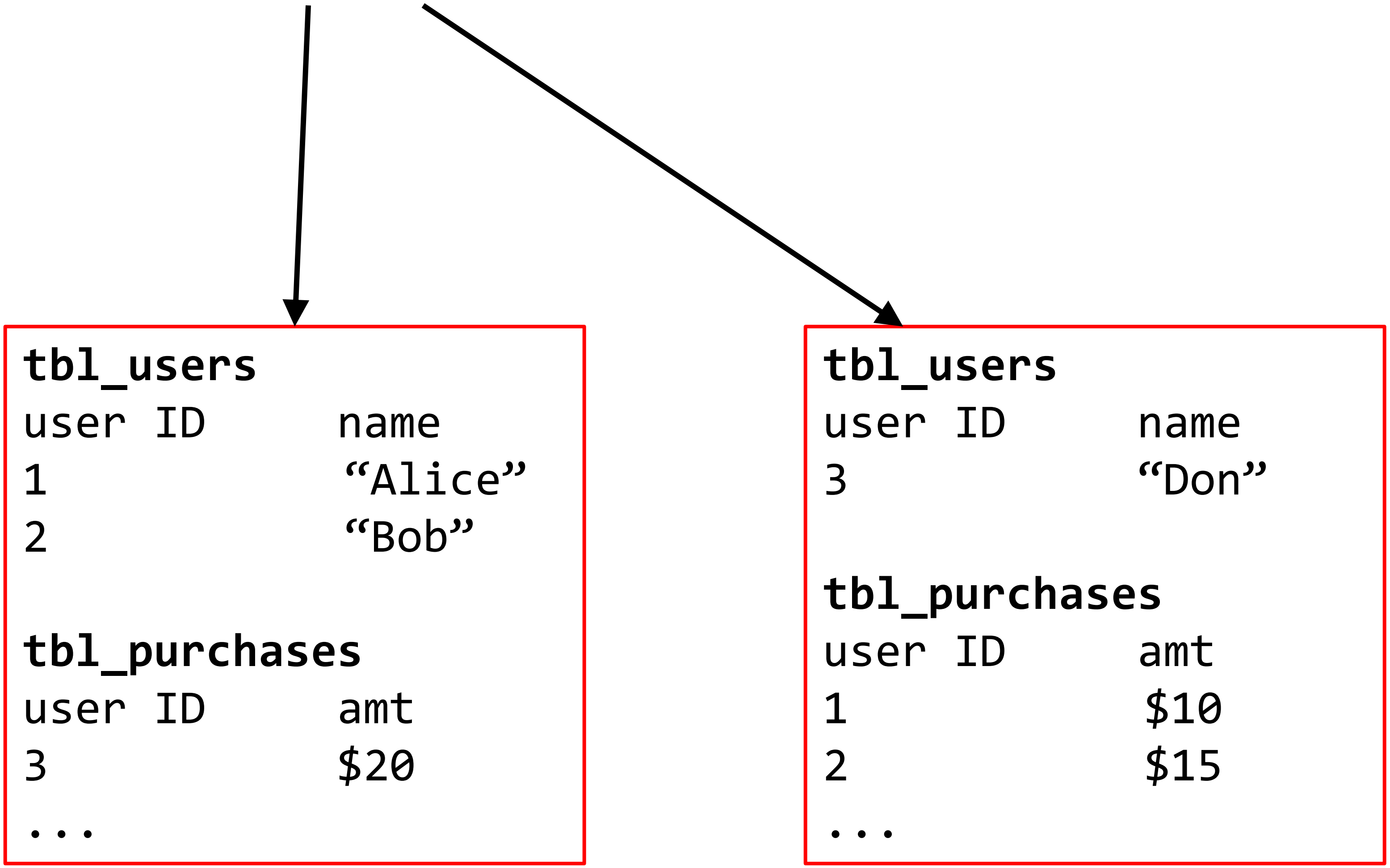
user ID	name
3	“Don”

tbl_purchases

user ID	amt
1	$10
2	$15
...	

Approach: Send queries to multiple DBs

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



tbl_users

user ID	name
1	“Alice”
2	“Bob”

tbl_purchases

user ID	amt
3	$20
...	

tbl_users

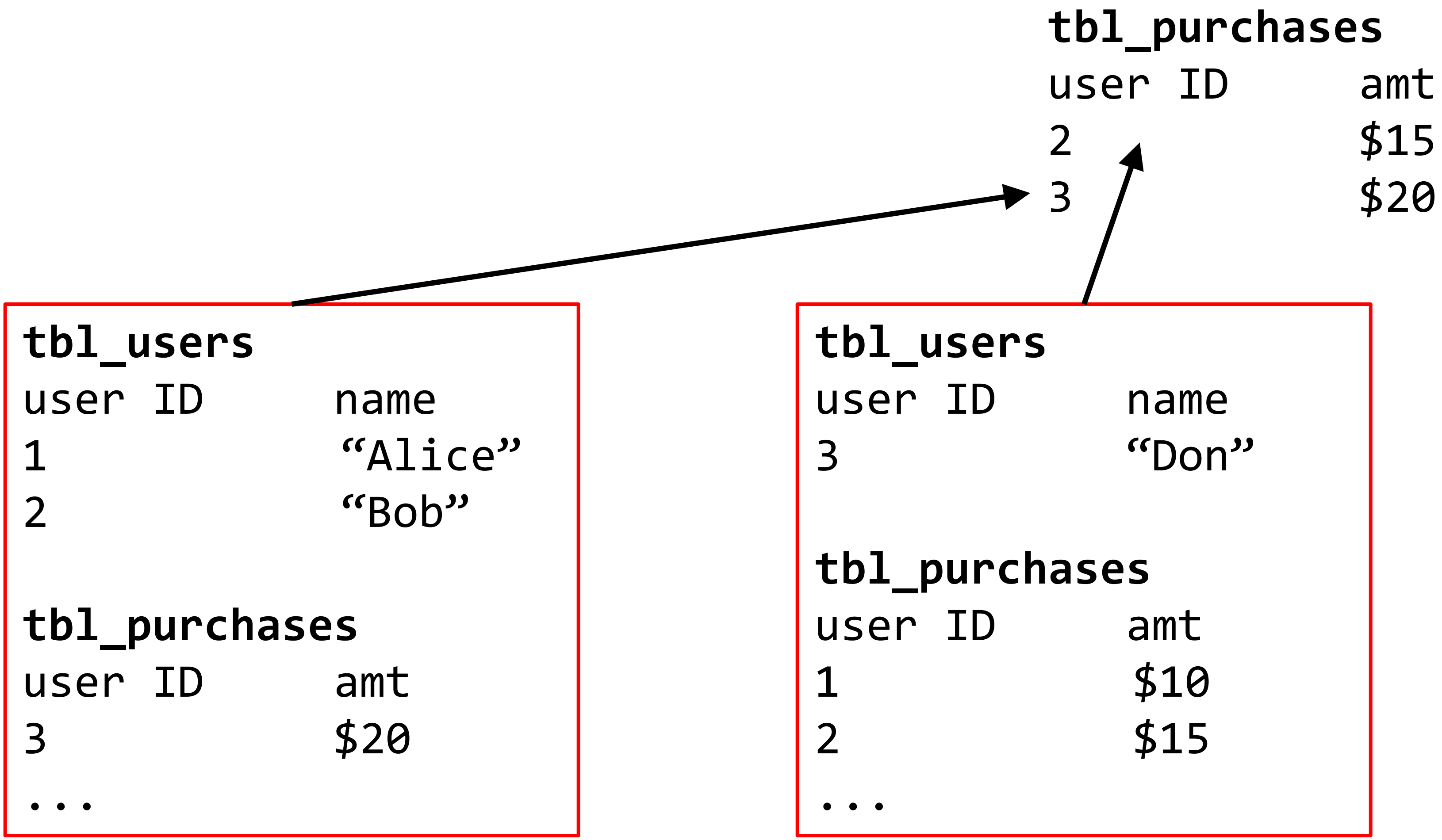
user ID	name
3	“Don”

tbl_purchases

user ID	amt
1	$10
2	$15
...	

… 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

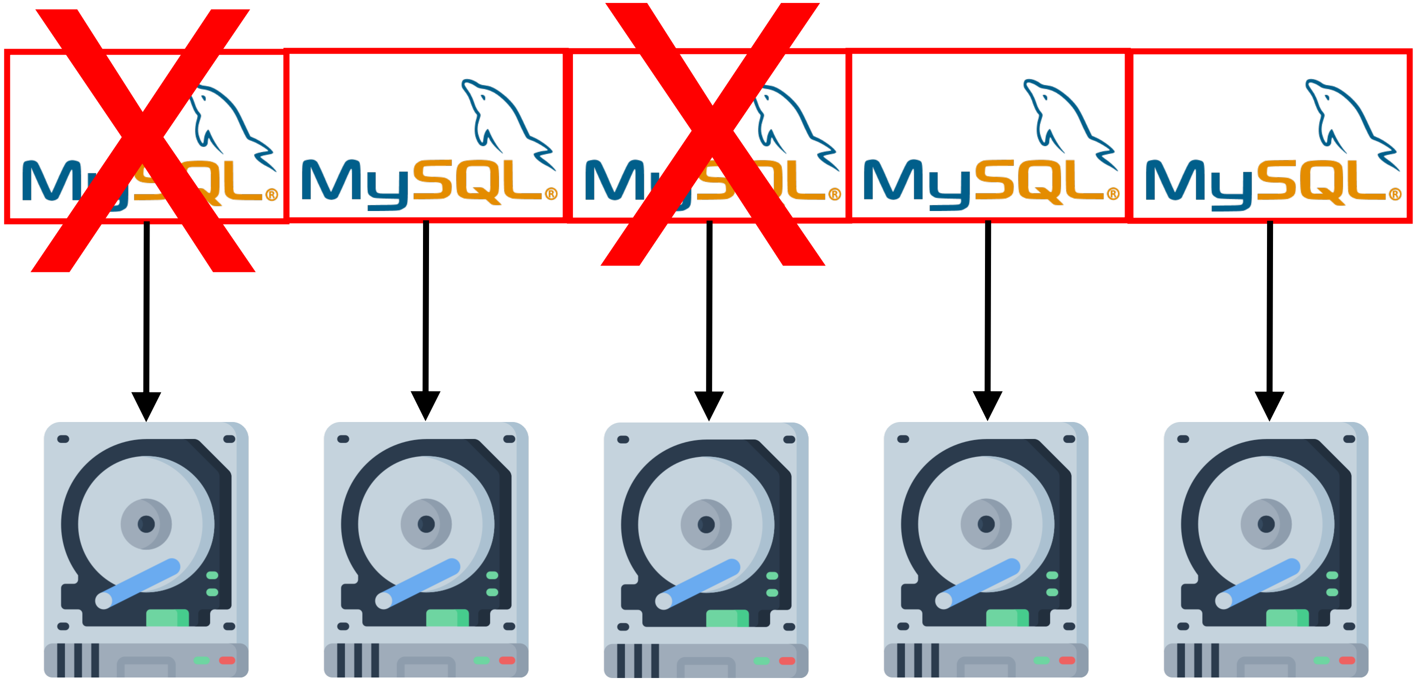
user ID	name
3	“Don”

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…

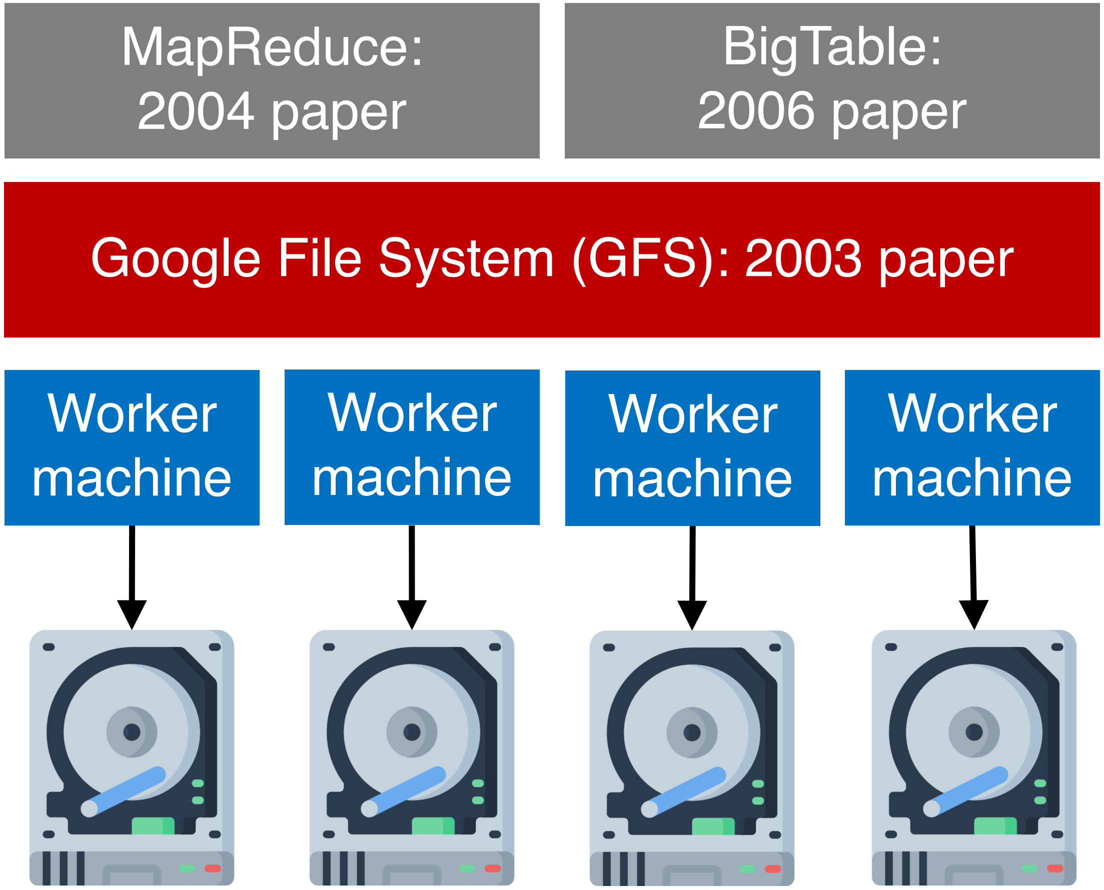


…1000 more…

Motivation for large DFS (GFS / HDFS)

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

Google big data infrastructure



MapReduce: 2004 paper

BigTable: 2006 paper

Google File System (GFS): 2003 paper

Today

Worker machine

Worker machine

Worker machine

Worker machine

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

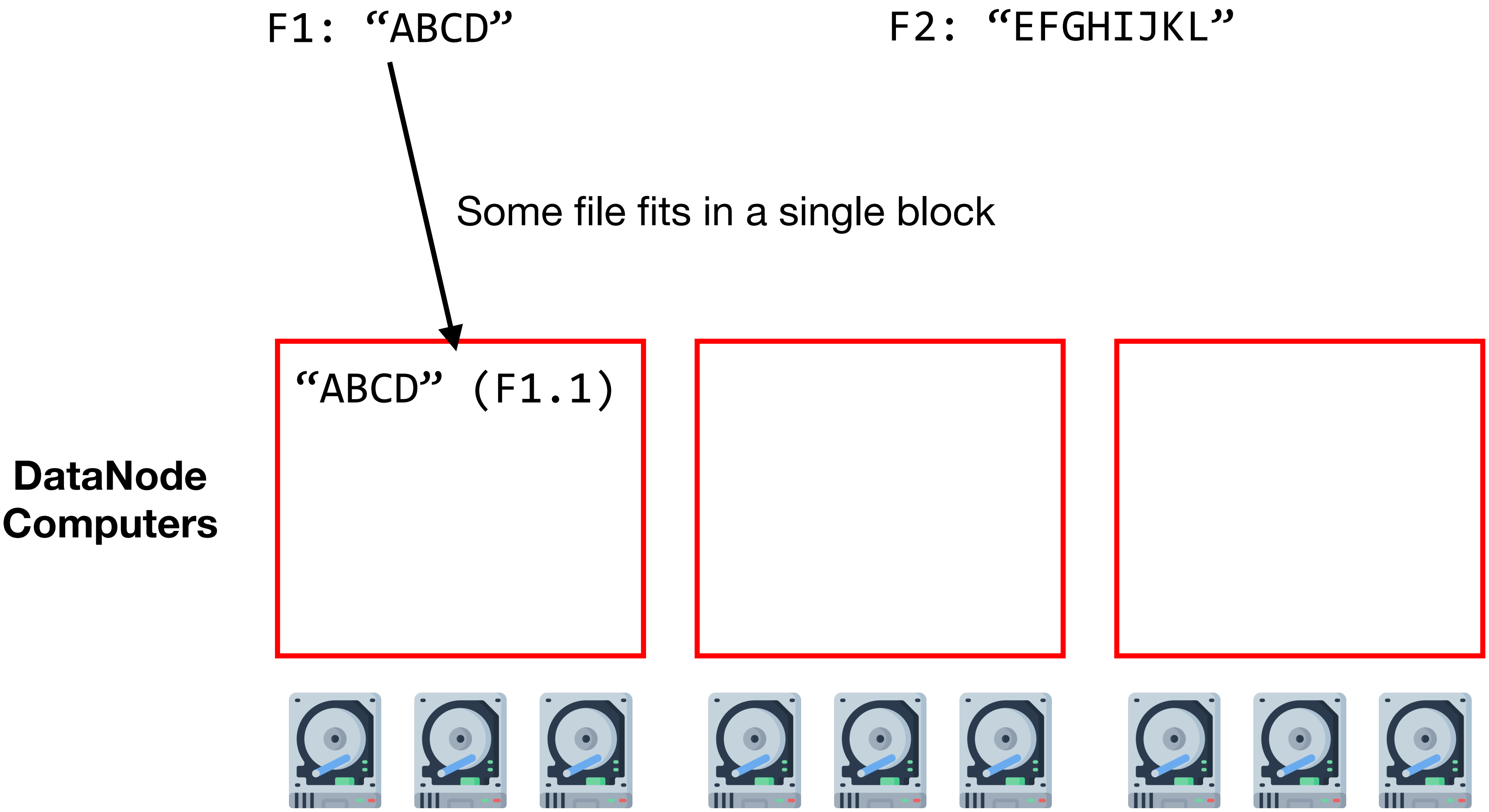
Hadoop ecosystem

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

	Google (paper only)	Hadoop (open source)	Modern Hadoop
Distributed File System	GFS	HDFS	
Distributed Processing & Analytics	MapReduce	Hadoop MapReduce	Spark
Distributed Database	BigTable	HBase	MongoDB

https://hadoop.apache.org/

HDFS: DataNodes store file blocks



F1: “ABCD”

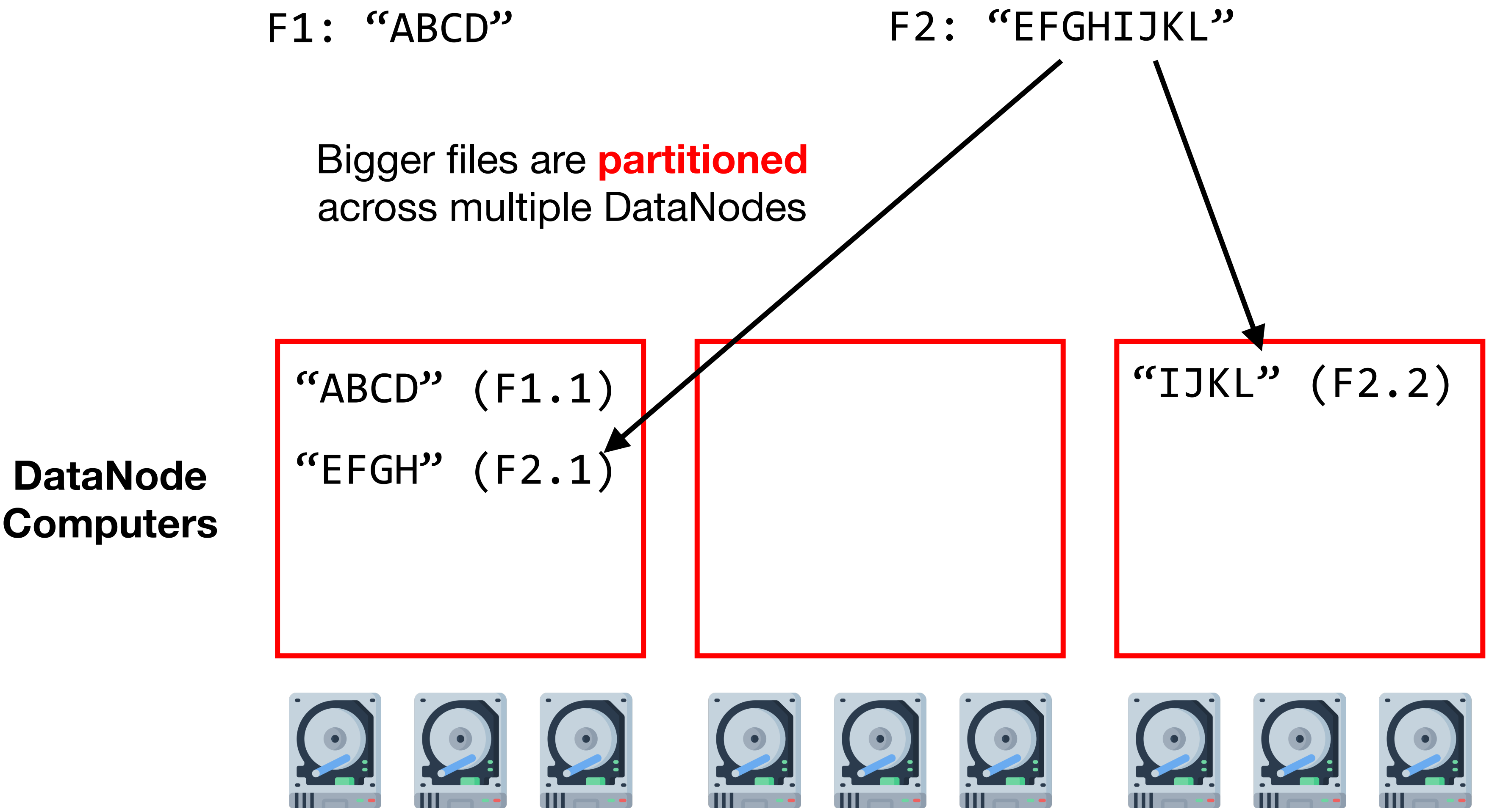
F2: “EFGHIJKL”

Some file fits in a single block

“ABCD” (F1.1)

DataNode Computers

HDFS: Partitioning across DataNodes



F1: “ABCD”

F2: “EFGHIJKL”

Bigger files are **partitioned** across multiple DataNodes

DataNode Computers

“ABCD” (F1.1)

“EFGH” (F2.1)

“IJKL” (F2.2)

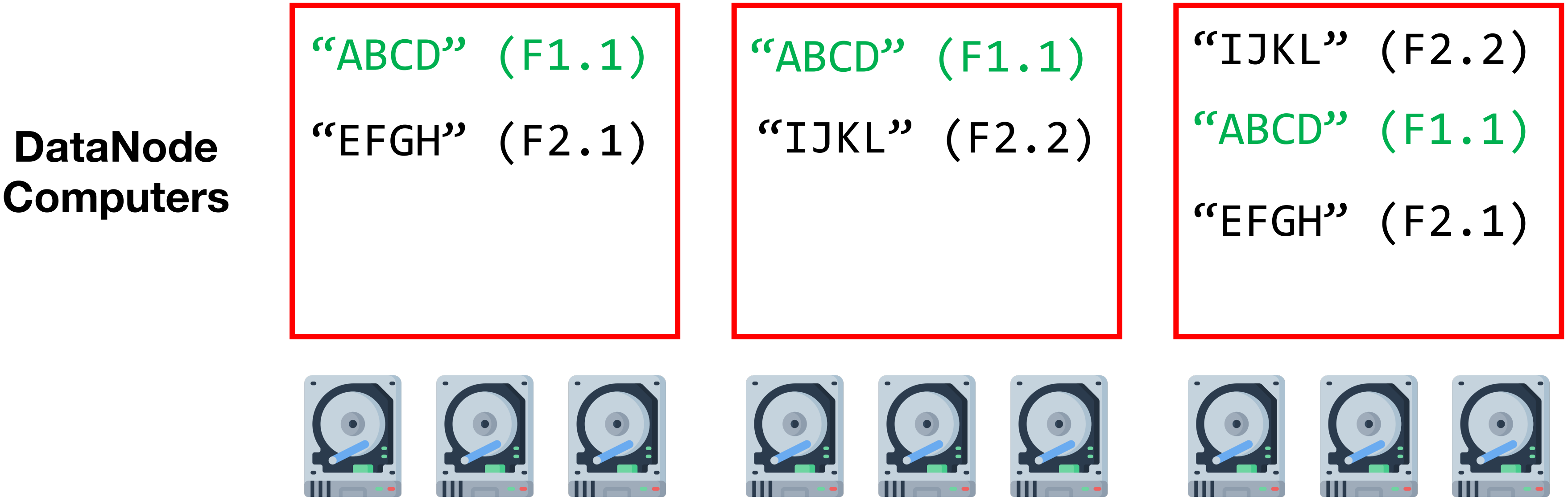
HDFS: Replication across DataNodes

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”

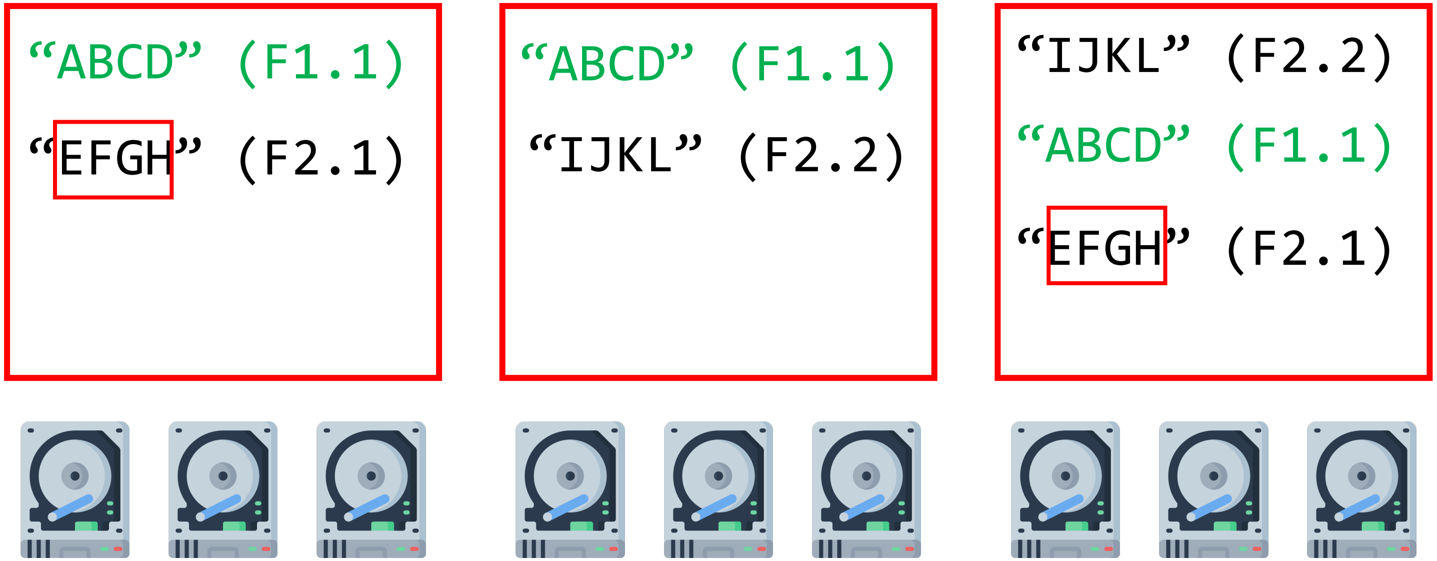
3x replication

F2: “EFGHIJKL”

2x replication

Logical blocks vs. physical blocks

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”

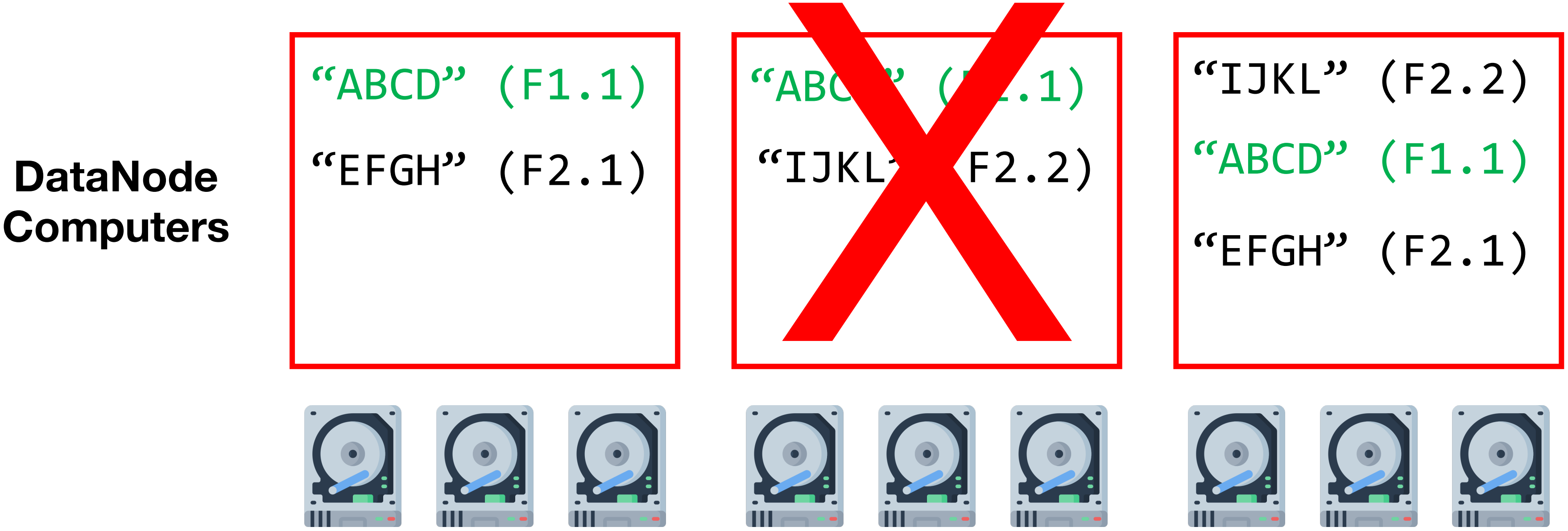
3x replication

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

“ABCD” (F1.1)

“EFGH” (F2.1)

“ABCD” (F1.1)

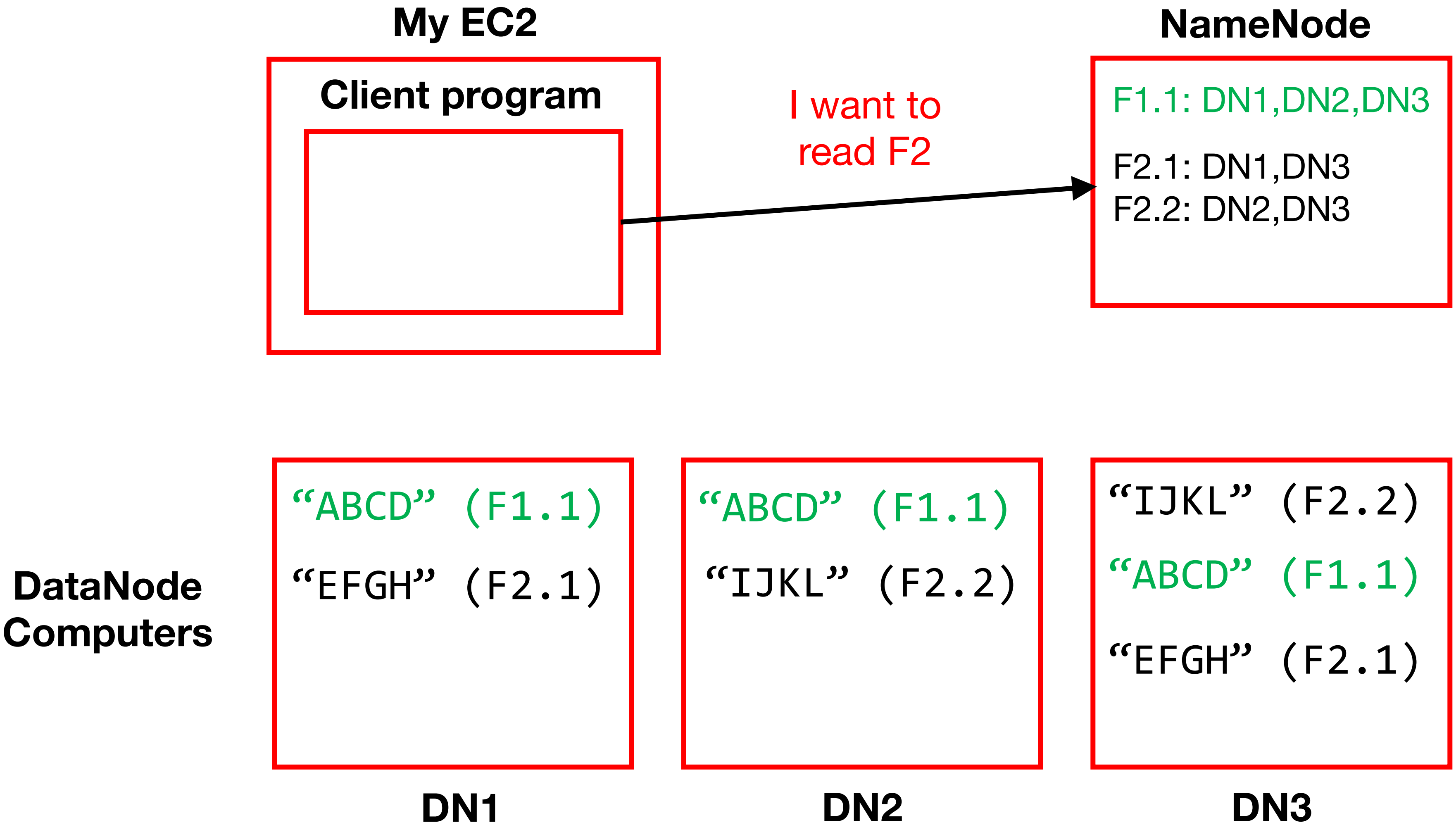
“IJKL” (F2.2)

“IJKL” (F2.2)

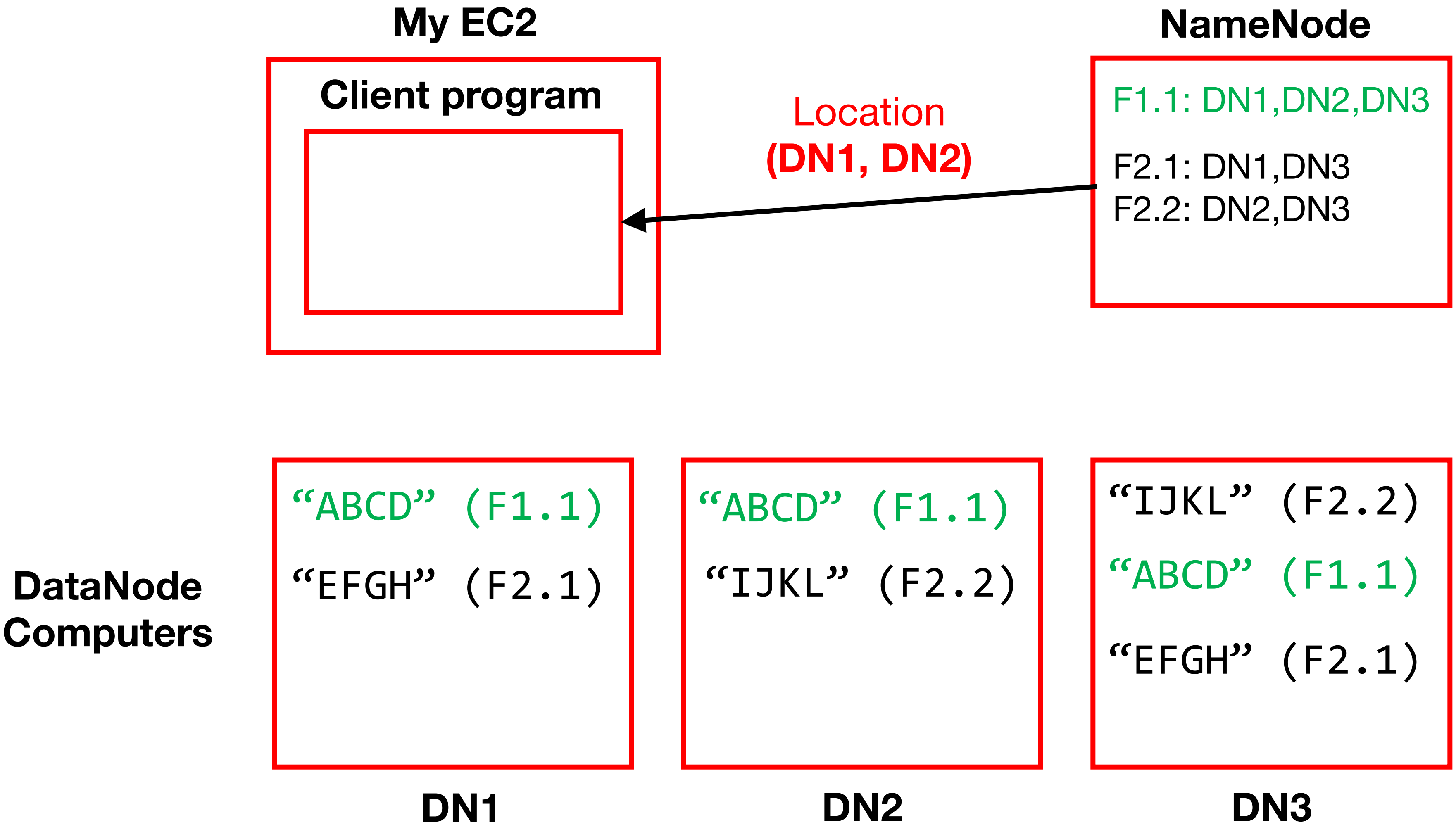
“ABCD” (F1.1)

“EFGH” (F2.1)

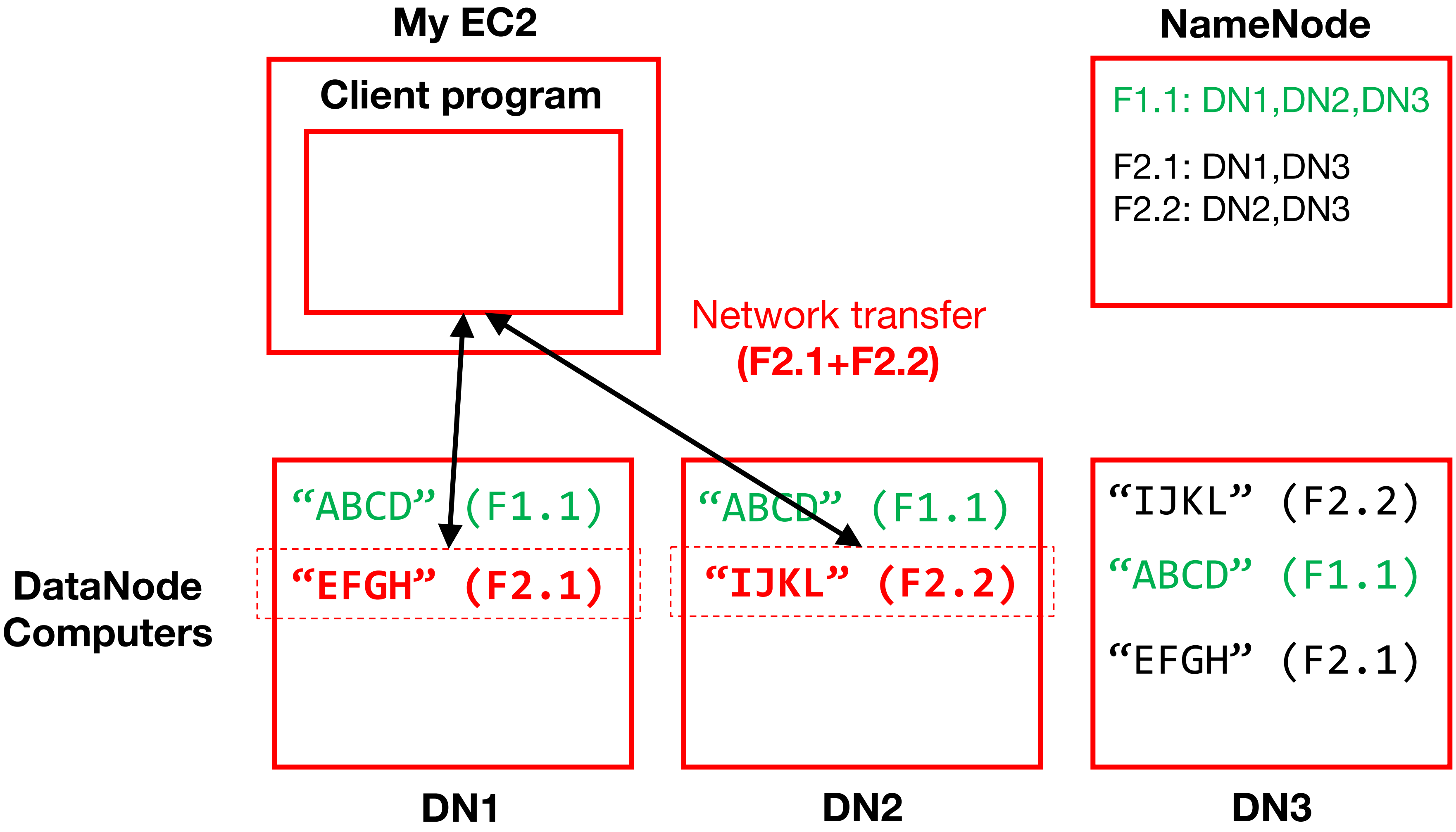
NameNode/Worker architecture



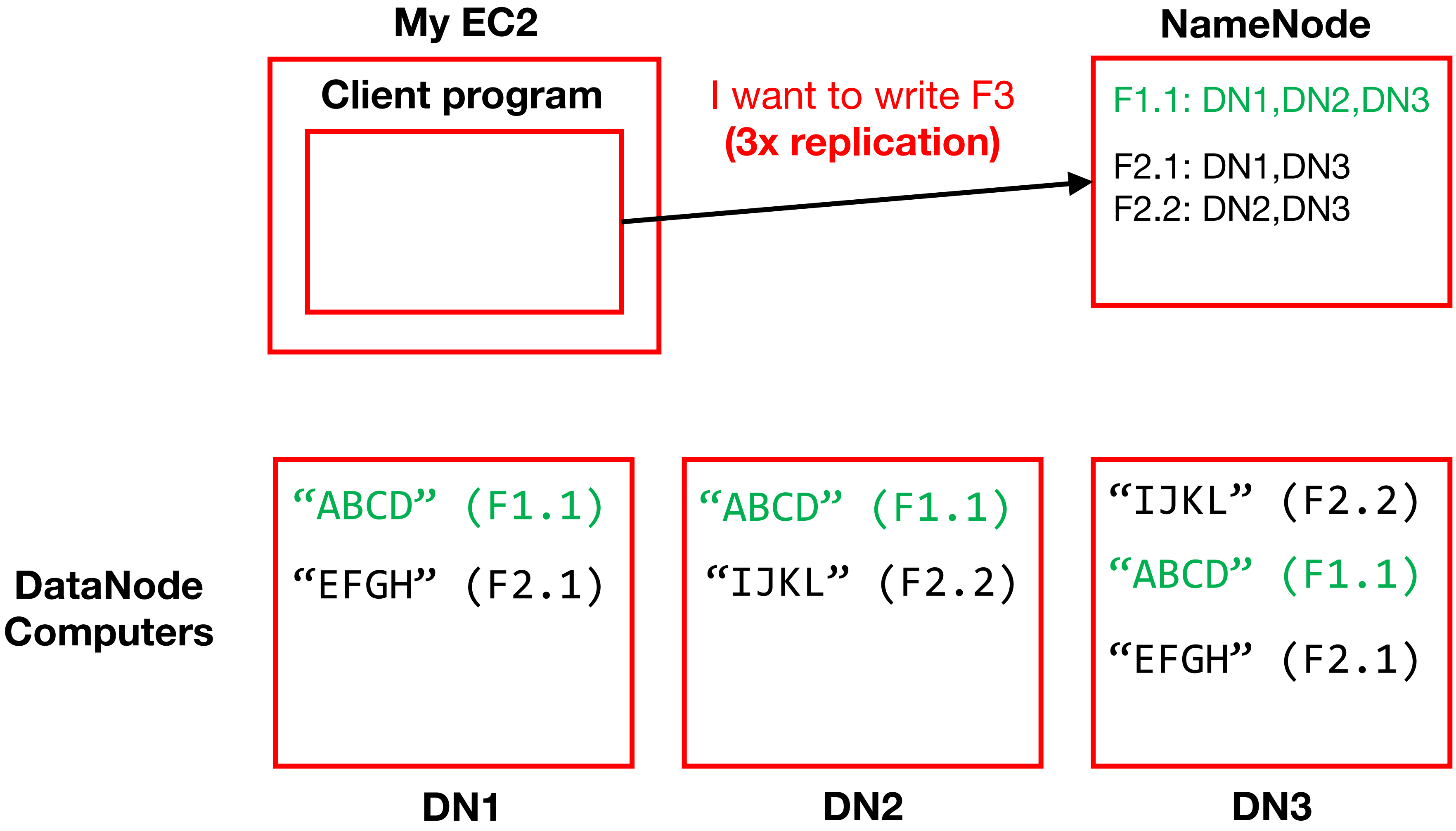
NameNode/Worker architecture



NameNode/Worker architecture



NameNode/Worker architecture



My EC2

Client program

I want to write F3

(3x replication)

NameNode

F1.1: DN1,DN2,DN3

F2.1: DN1,DN3

F2.2: DN2,DN3

DataNode Computers

DN1

```
"ABCD" (F1.1)
"EFGH" (F2.1)
```

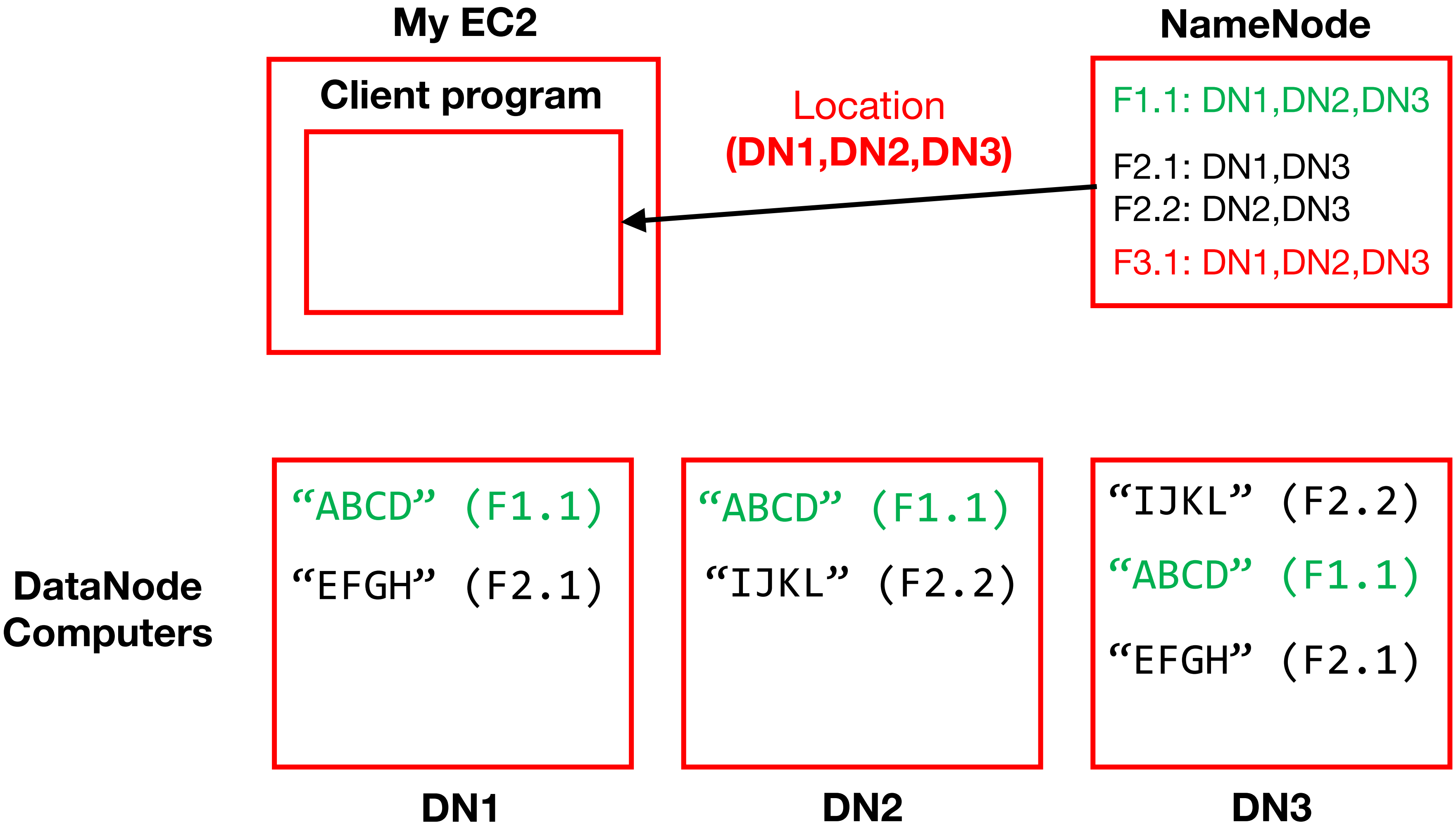
DN2

```
"ABCD" (F1.1)
"IJKL" (F2.2)
```

DN3

```
"IJKL" (F2.2)
"ABCD" (F1.1)
"EFGH" (F2.1)
```

NameNode/Worker architecture



My EC2

Client program

Location
(DN1,DN2,DN3)

NameNode

F1.1: DN1,DN2,DN3

F2.1: DN1,DN3
F2.2: DN2,DN3

F3.1: DN1,DN2,DN3

DataNode Computers

DN1

`“ABCD” (F1.1)`

`“EFGH” (F2.1)`

DN2

`“ABCD” (F1.1)`

`“IJKL” (F2.2)`

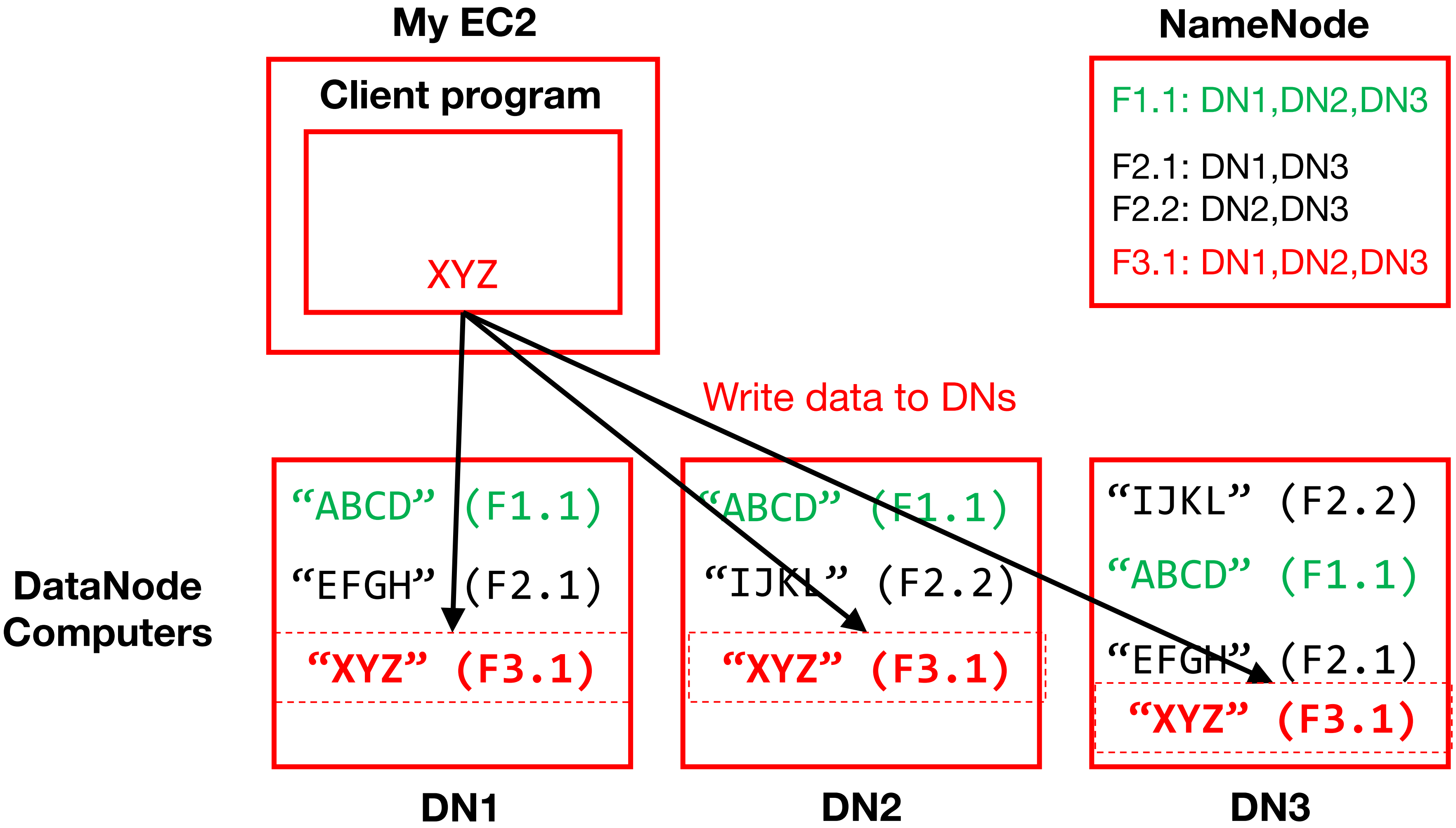
DN3

`“IJKL” (F2.2)`

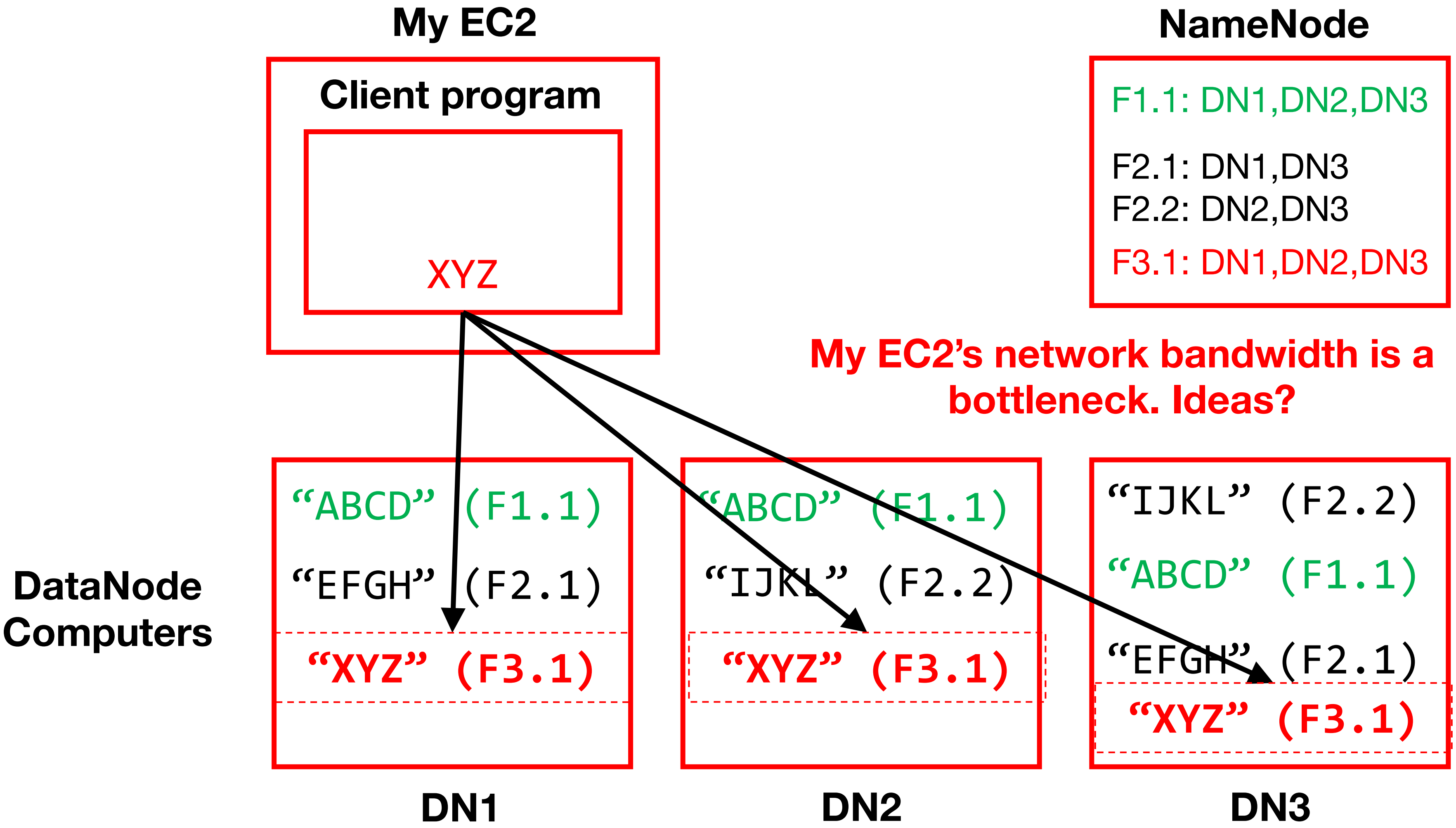
`“ABCD” (F1.1)`

`“EFGH” (F2.1)`

NameNode/Worker architecture



NameNode/Worker architecture



My EC2

Client program

XYZ

NameNode

F1.1: DN1,DN2,DN3

F2.1: DN1,DN3
F2.2: DN2,DN3

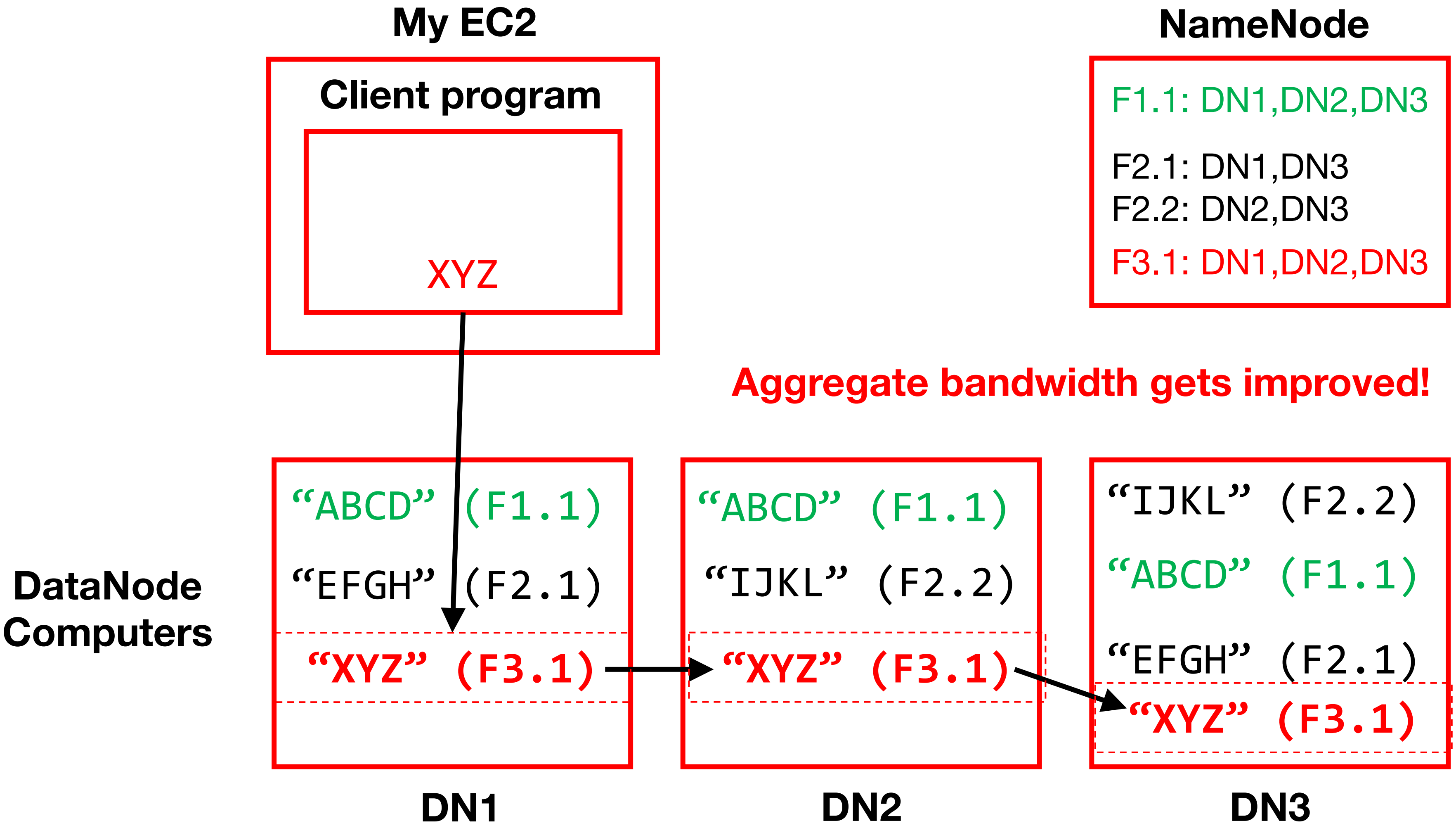
F3.1: DN1,DN2,DN3

My EC2's network bandwidth is a bottleneck. Ideas?

DataNode Computers

DN1	DN2	DN3
"ABCD" (F1.1)	"ABCD" (F1.1)	"IJKL" (F2.2)
"EFGH" (F2.1)	"IJKL" (F2.2)	"ABCD" (F1.1)
"XYZ" (F3.1)	**"XYZ" (F3.1)**	"EFGH" (F2.1)
		"XYZ" (F3.1)

Pipelined writes

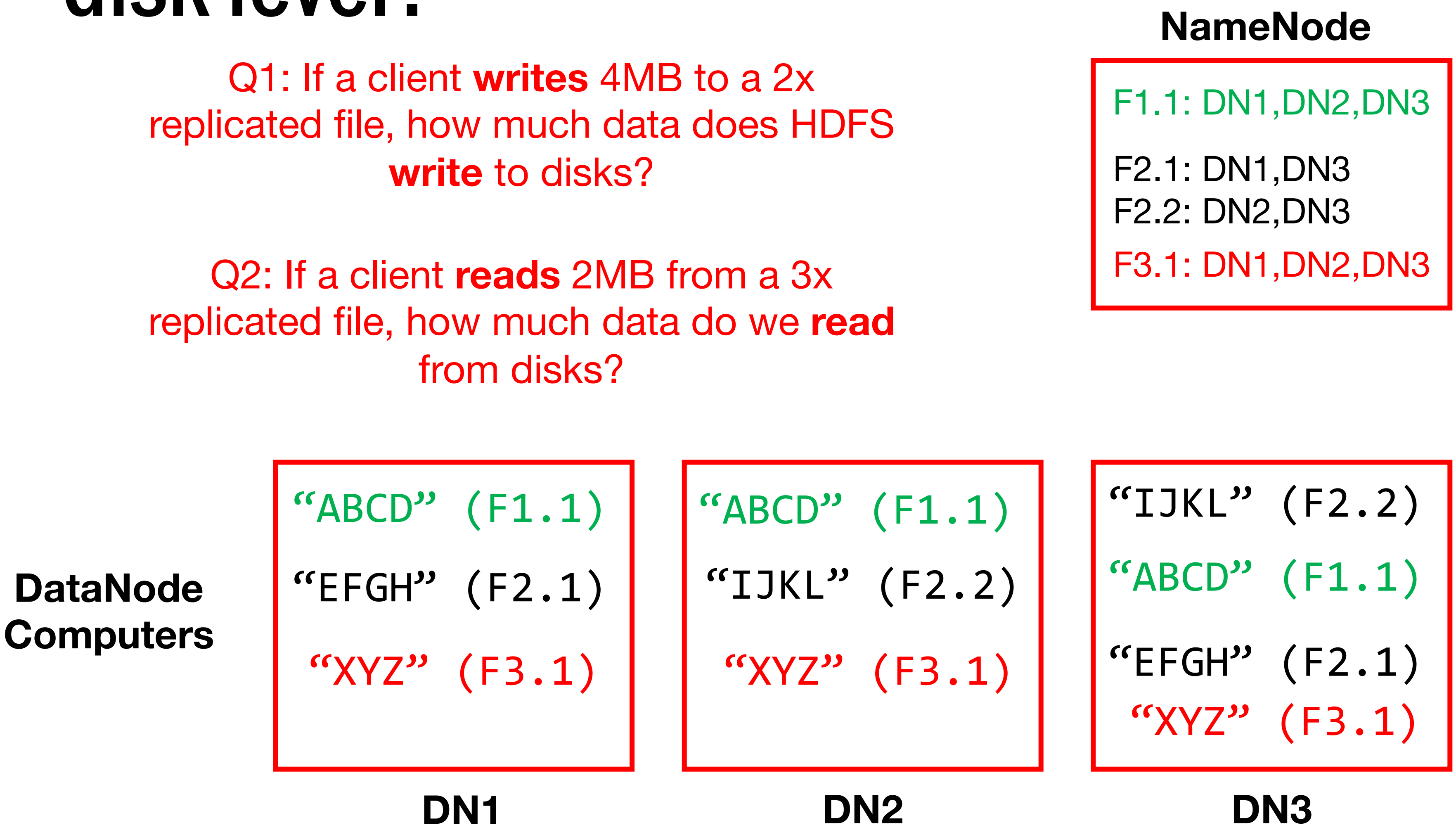


Aggregate bandwidth gets improved!

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

DN1

“ABCD” (F1.1)
“EFGH” (F2.1)
“XYZ” (F3.1)

DN2

“ABCD” (F1.1)
“IJKL” (F2.2)
“XYZ” (F3.1)

DN3

“IJKL” (F2.2)
“ABCD” (F1.1)
“EFGH” (F2.1)
“XYZ” (F3.1)

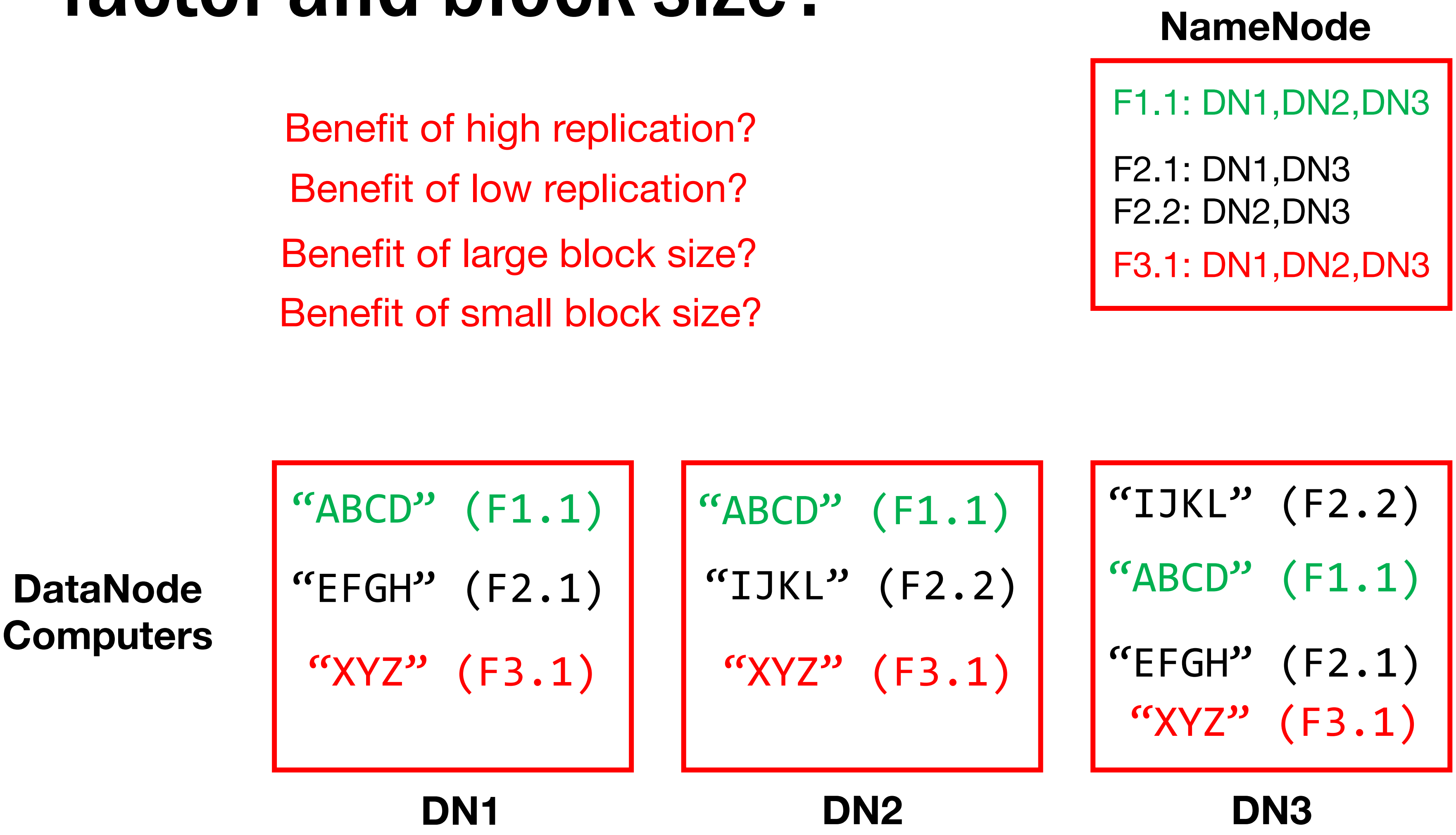
What are the tradeoffs of replication factor and block size?

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

DN1

“ABCD” (F1.1)
“EFGH” (F2.1)
“XYZ” (F3.1)

DN2

“ABCD” (F1.1)
“IJKL” (F2.2)
“XYZ” (F3.1)

DN3

“IJKL” (F2.2)
“ABCD” (F1.1)
“EFGH” (F2.1)
“XYZ” (F3.1)

What are the tradeoffs of replication factor and block size?

Better FT
Better locality
Better LB

Benefit of high replication?

Faster writes
Lower storage cost

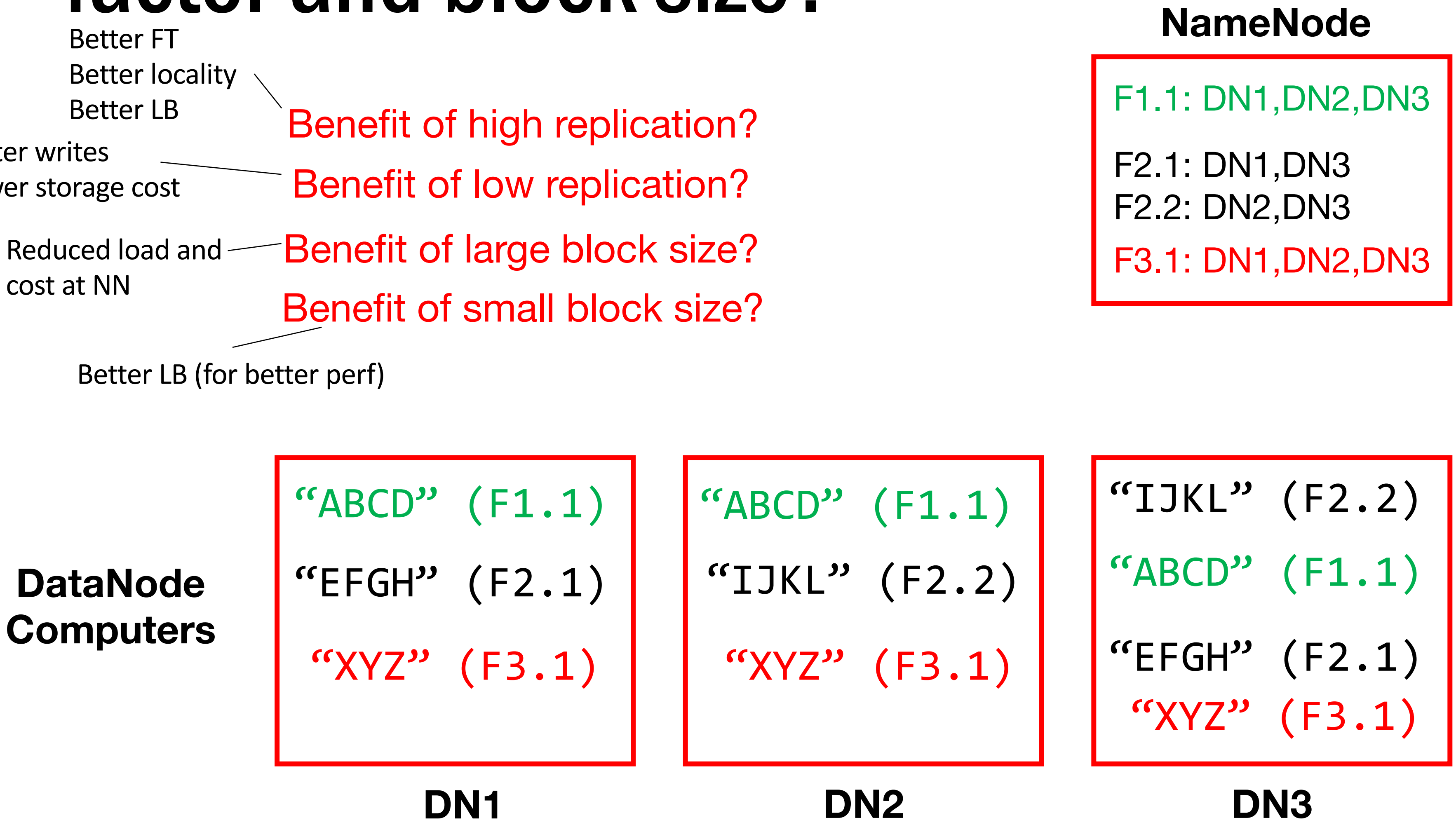
Benefit of low replication?

Reduced load and cost at NN

Benefit of large block size?

Benefit of small block size?

Better LB (for better perf)



NameNode

F1.1: DN1,DN2,DN3

F2.1: DN1,DN3
F2.2: DN2,DN3

F3.1: DN1,DN2,DN3

DataNode Computers

DN1

“ABCD” (F1.1)
“EFGH” (F2.1)
“XYZ” (F3.1)

DN2

“ABCD” (F1.1)
“IJKL” (F2.2)
“XYZ” (F3.1)

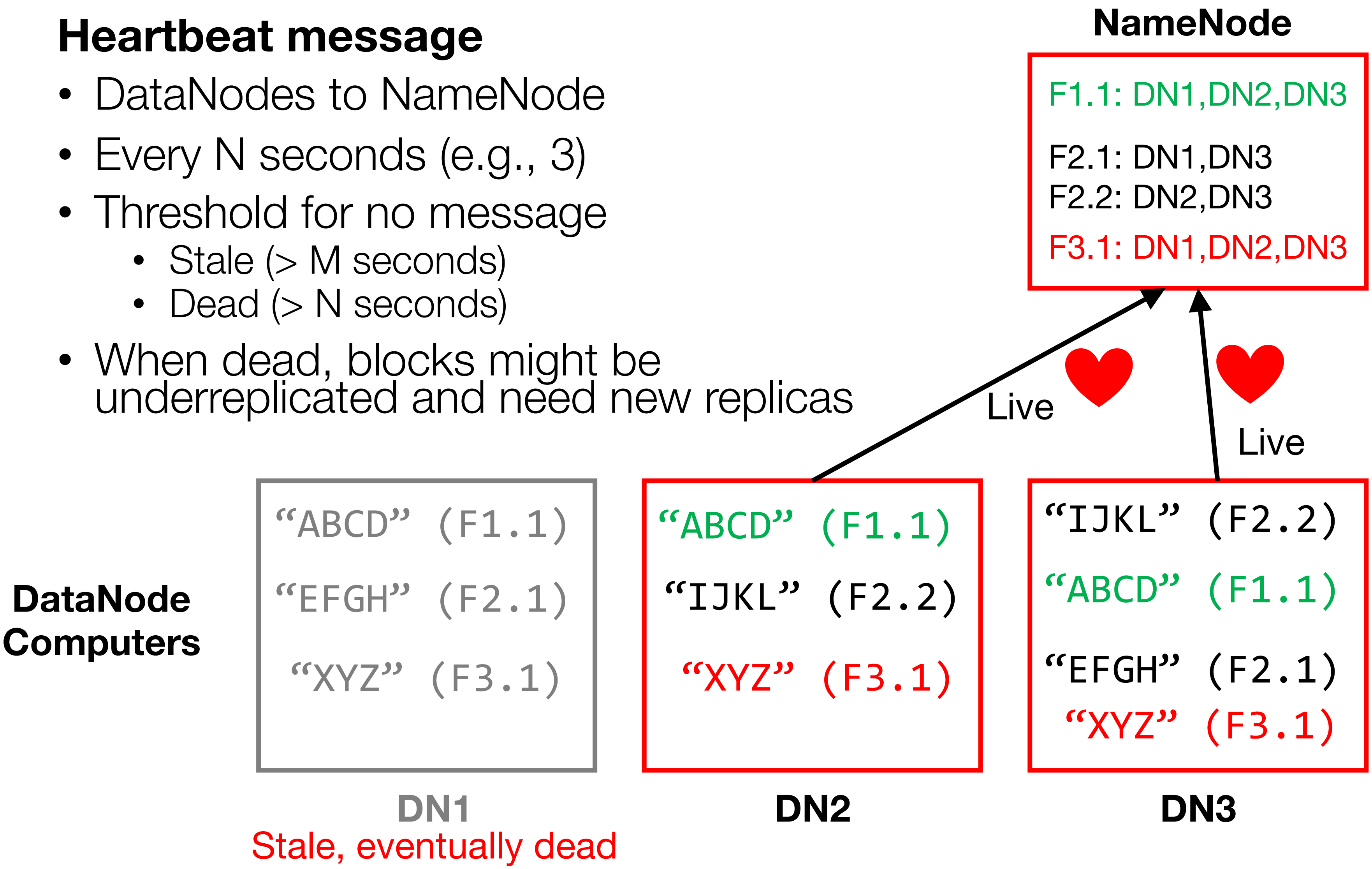
DN3

“IJKL” (F2.2)
“ABCD” (F1.1)
“EFGH” (F2.1)
“XYZ” (F3.1)

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



NameNode

F1.1: DN1,DN2,DN3

F2.1: DN1,DN3
F2.2: DN2,DN3

F3.1: DN1,DN2,DN3

DataNode Computers

DN1	DN2	DN3
"ABCD" (F1.1)	"ABCD" (F1.1)	"IJKL" (F2.2)
"EFGH" (F2.1)	"IJKL" (F2.2)	"ABCD" (F1.1)
"XYZ" (F3.1)	"XYZ" (F3.1)	"EFGH" (F2.1)
		"XYZ" (F3.1)

DN1 Stale, eventually dead

DN2

DN3

Summary: Some key ideas applied to GFS/HDFS

- To build complex systems…

- To scale out…

- To handle faults…

- To detect faults…

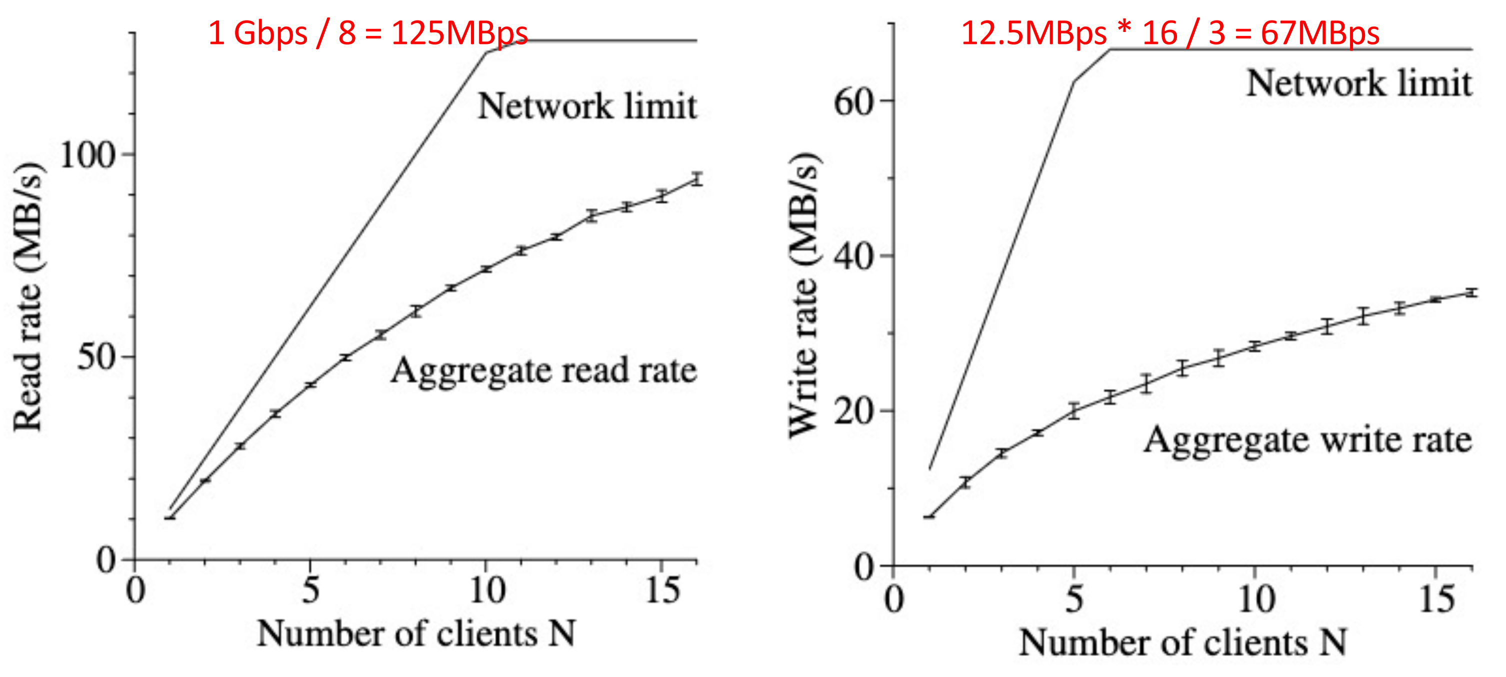
- To optimize I/O…

Summary: Some key ideas applied to GFS/HDFS

- To build complex systems…
  - Compose layers of subsystems
- To scale out…
  - Partition your data
- To handle faults…
  - Replicate your data
- To detect faults…
  - Send heartbeats
- To optimize I/O…
  - Pipeline writes

Discussion: GFS eval (GFS paper)

List your takeaways from "Fig 3: Aggregate Throughput"



1 Gbps / 8 = 125MBps

12.5MBps * 16 / 3 = 67MBps

(a) Reads

(b) Writes