[544] HBase and Cassandra

Tyler Caraza-Harter

Learning Objectives

- describe HBase's approach to reliability (HDFS replication, RegionServer failover)
- describe the data models for HBase and Cassandra (wide row and wide partition, respectively)
- select columns Cassandra table to serve as partition keys, cluster keys, and static columns to make specific operations efficient

Hadoop Ecosystem

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

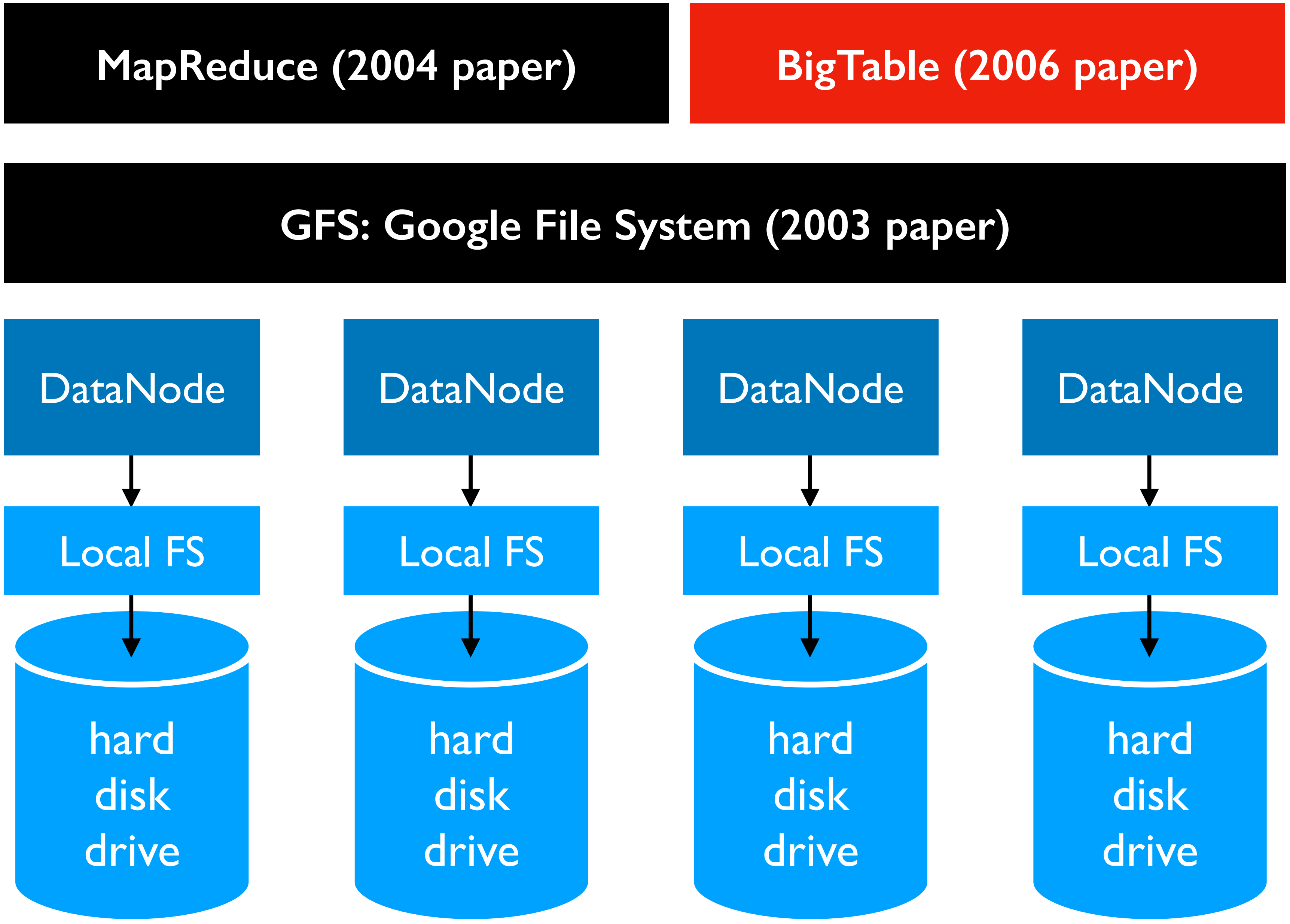
	Google (paper only)	**Hadoop, 1st gen (open source)**	**Modern Hadoop**
Distributed File System	GFS	HDFS	
Distributed Analytics	MapReduce	Hadoop MapReduce	Spark
Distributed Database	BigTable	HBase	Cassandra

Dynamo (Amazon) → Cassandra

Ecosystem: Ambari, Avro, Cassandra, Chukwa, HBase, Hive, Mahout, Ozone, Pig, Spark, Submarine, Tez, ZooKeeper

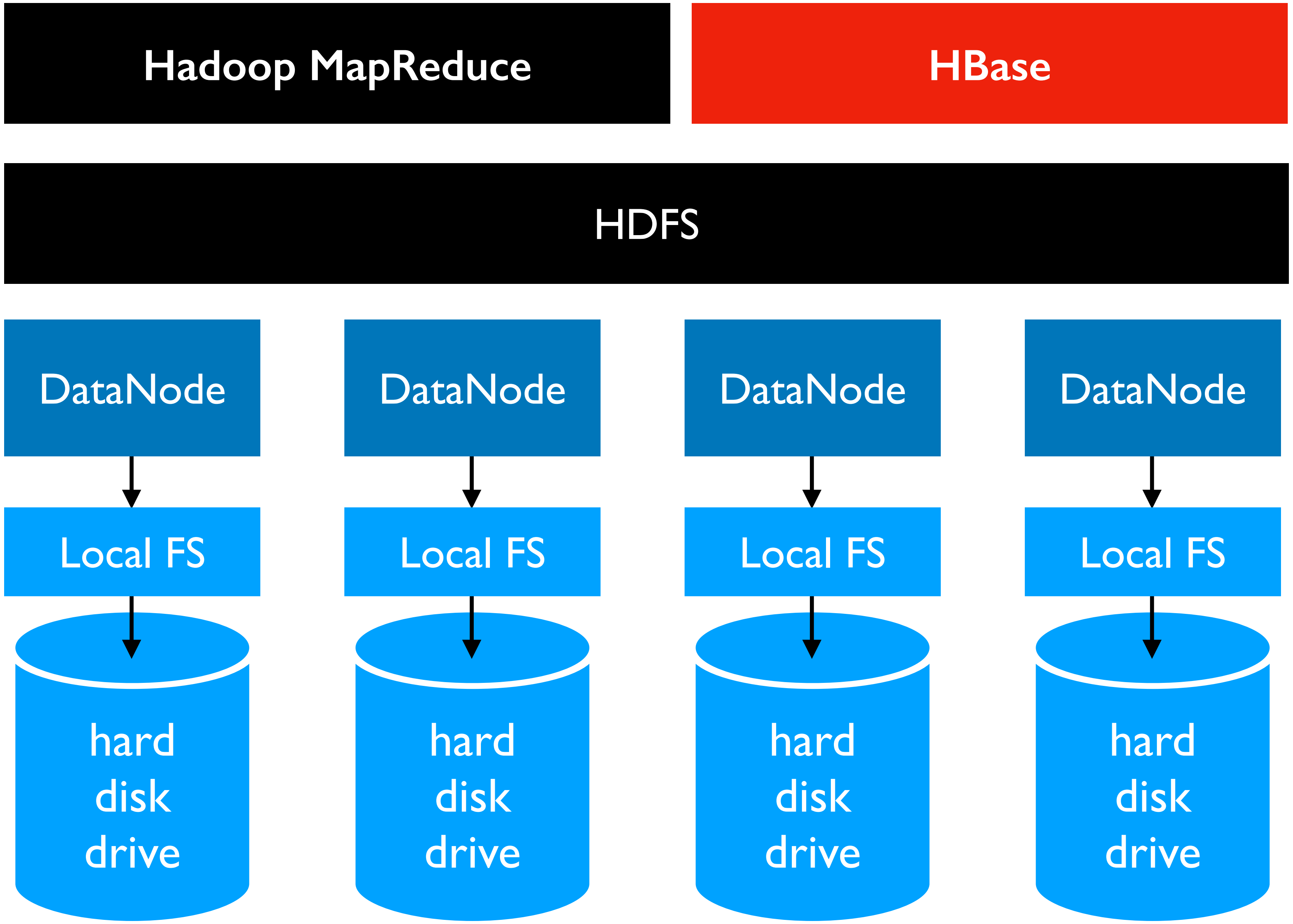
https://hadoop.apache.org/

Google Architecture

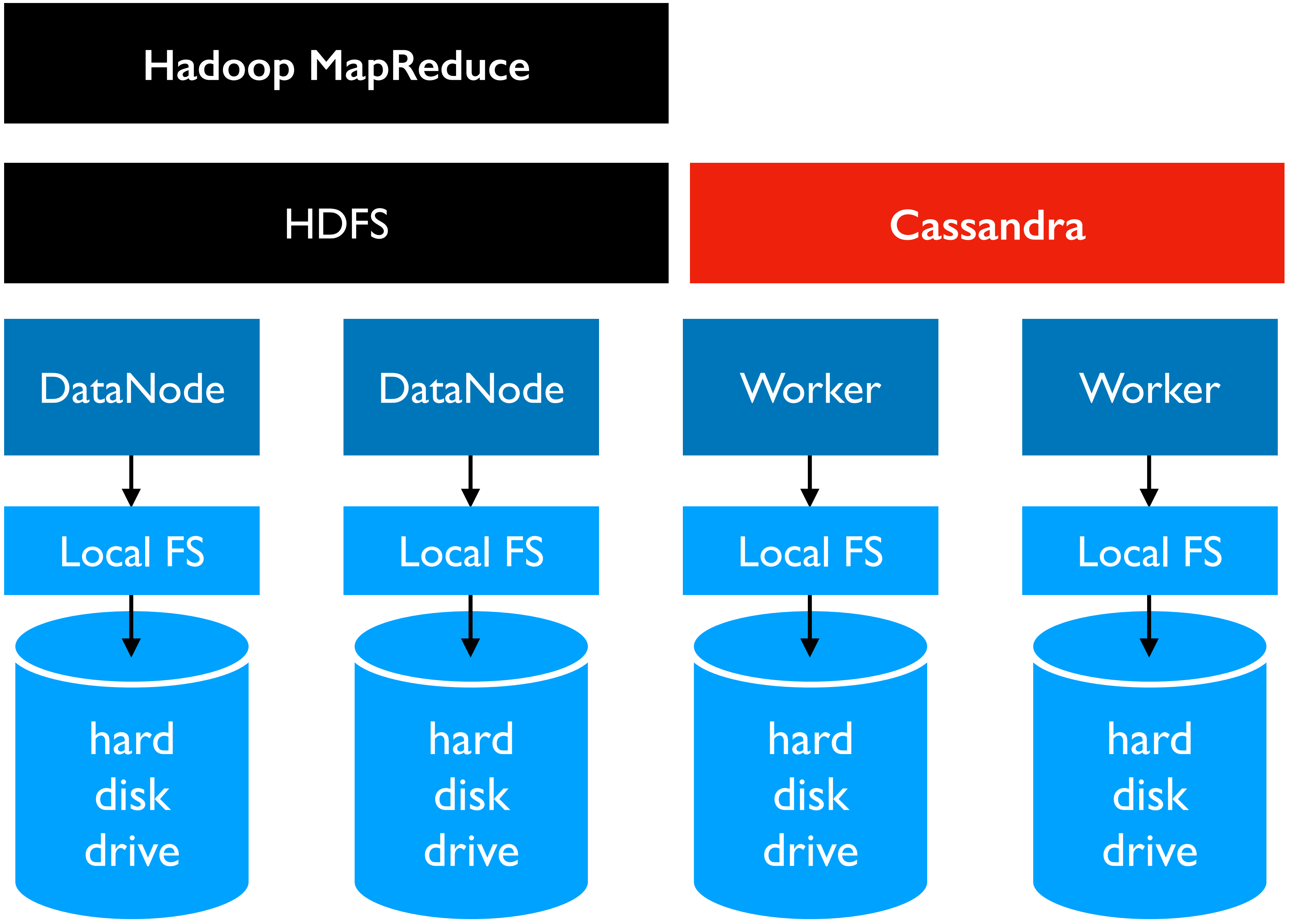


radical idea: base everything on lots of cheap, commodity hardware

Hadoop Ecosystem



Hadoop Ecosystem



Hadoop MapReduce

HDFS

Cassandra

DataNode

DataNode

Worker

Worker

Local FS

Local FS

Local FS

Local FS

hard disk drive

hard disk drive

hard disk drive

hard disk drive

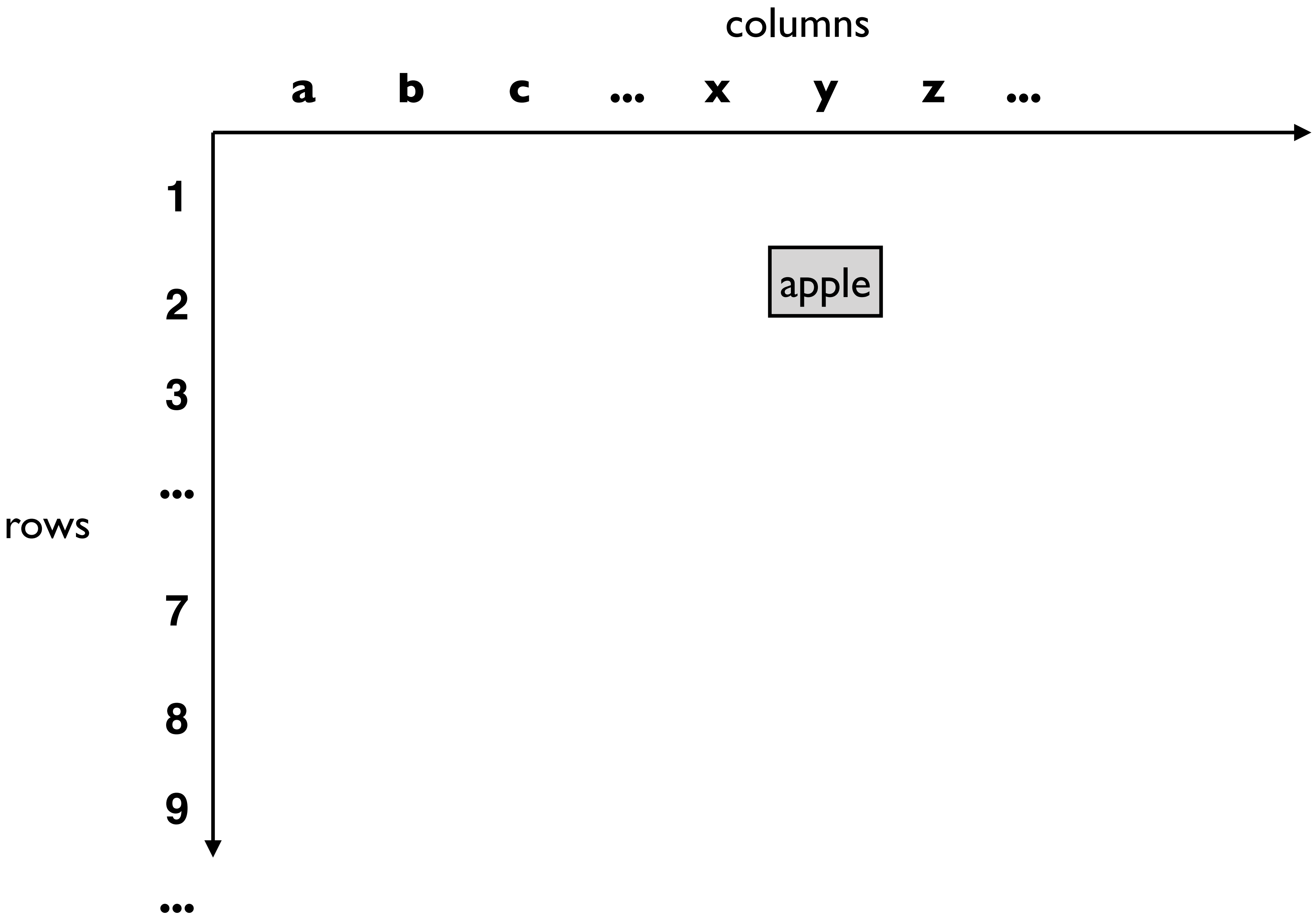
Outline: HBase and Cassandra

HBase

Cassandra Data Model

Demos

HBase Data Model: Versioned Sparse Tables



columns

a b c … x y z …

rows

1

2

3

…

7

8

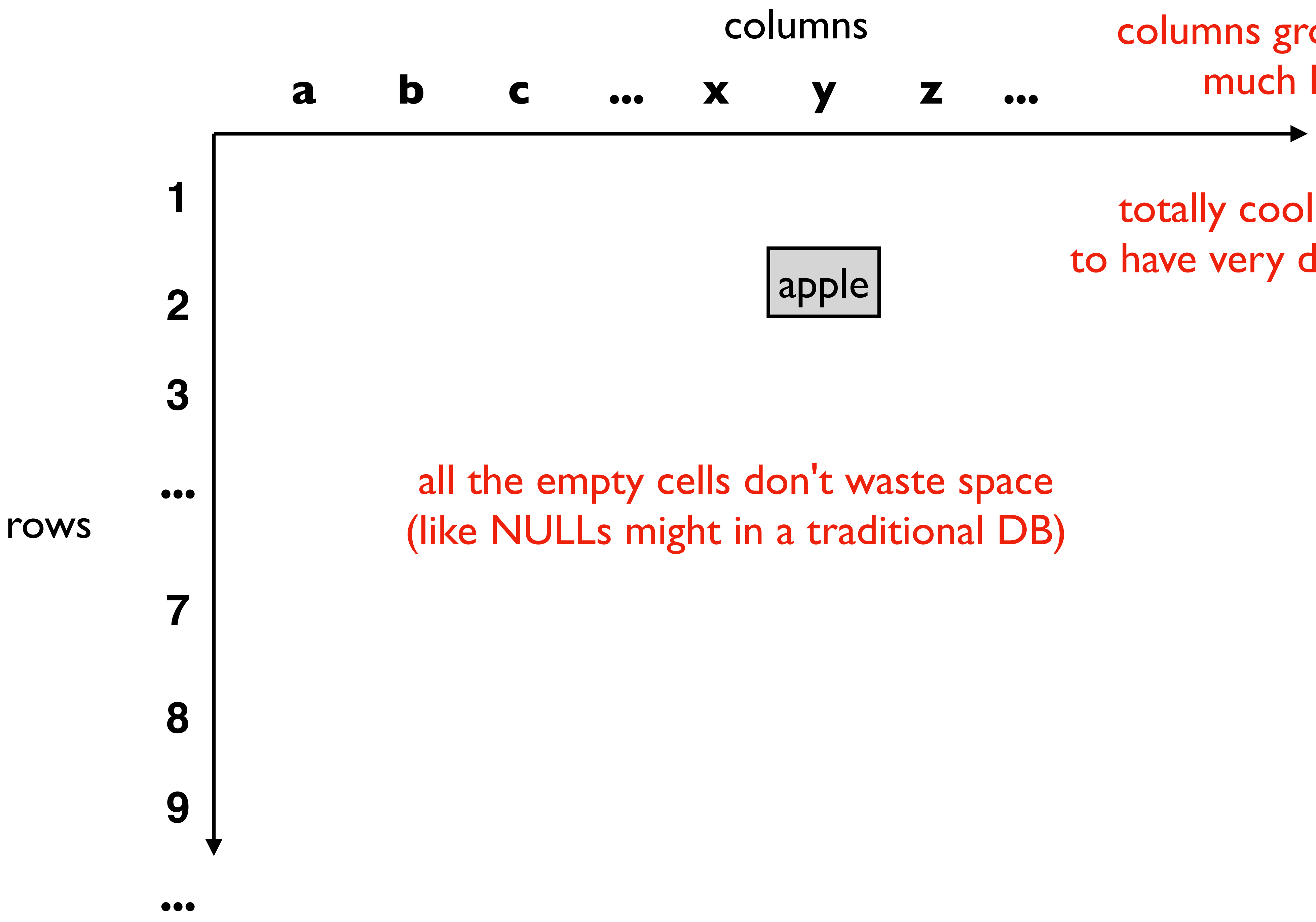
9

…

apple

```
table["2:y"] is "apple"
```

HBase Data Model: Versioned Sparse Tables



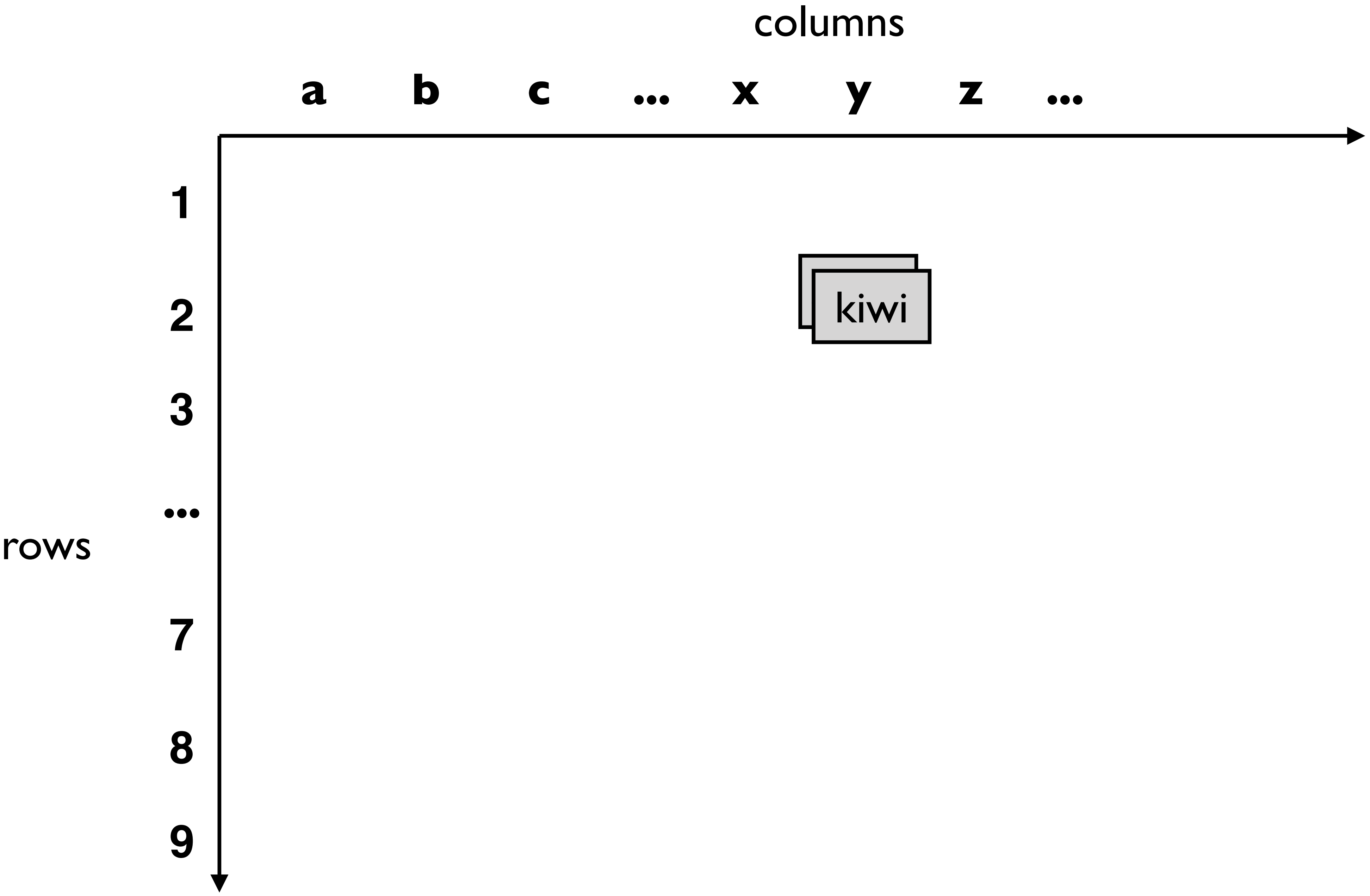
columns grow over time,
much like rows

totally cool for each row
to have very different columns

all the empty cells don't waste space
(like NULLs might in a traditional DB)

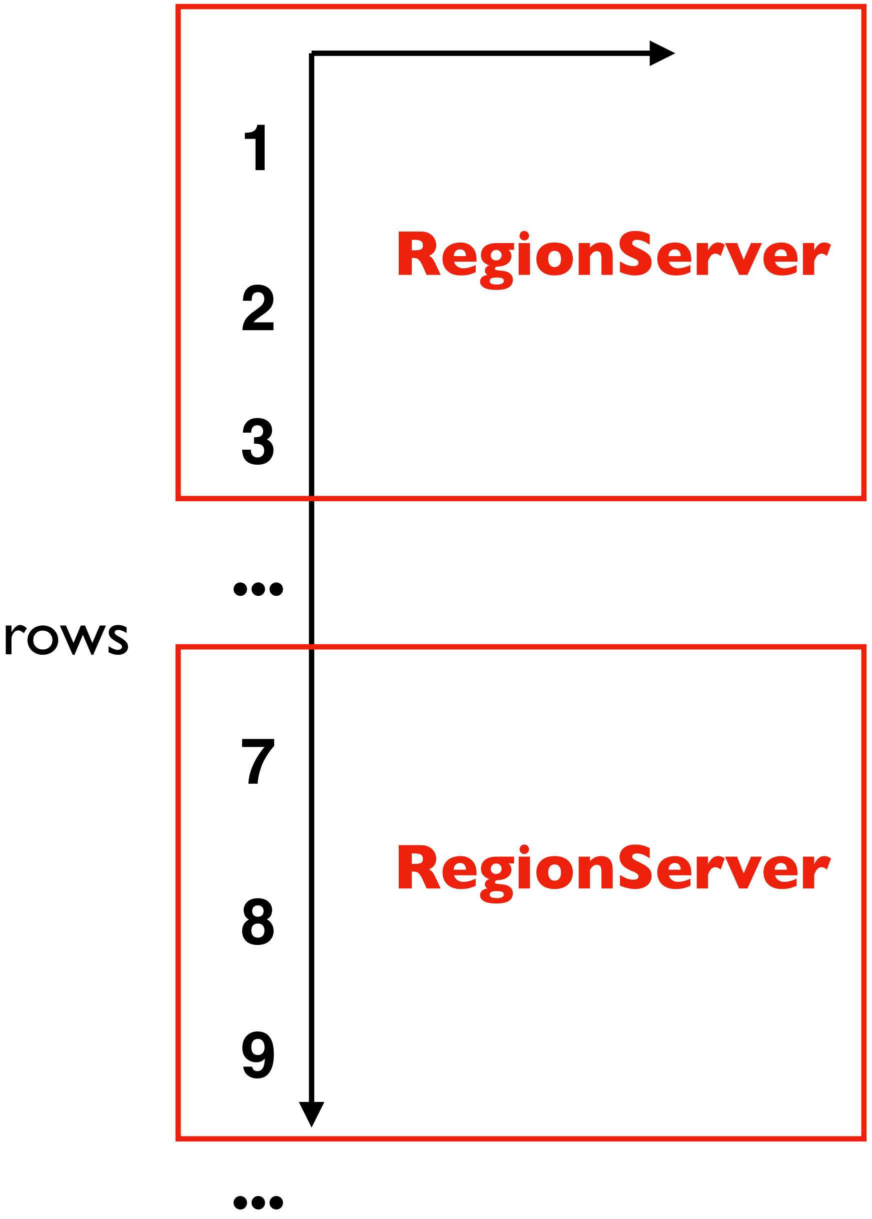
```
table["2:y"] is "apple"
```

HBase Data Model: Versioned Sparse Tables



```
table["2:y:v1"] is "apple"
table["2:y:v2"] is "kiwi"
```

Partitioning the Row Space



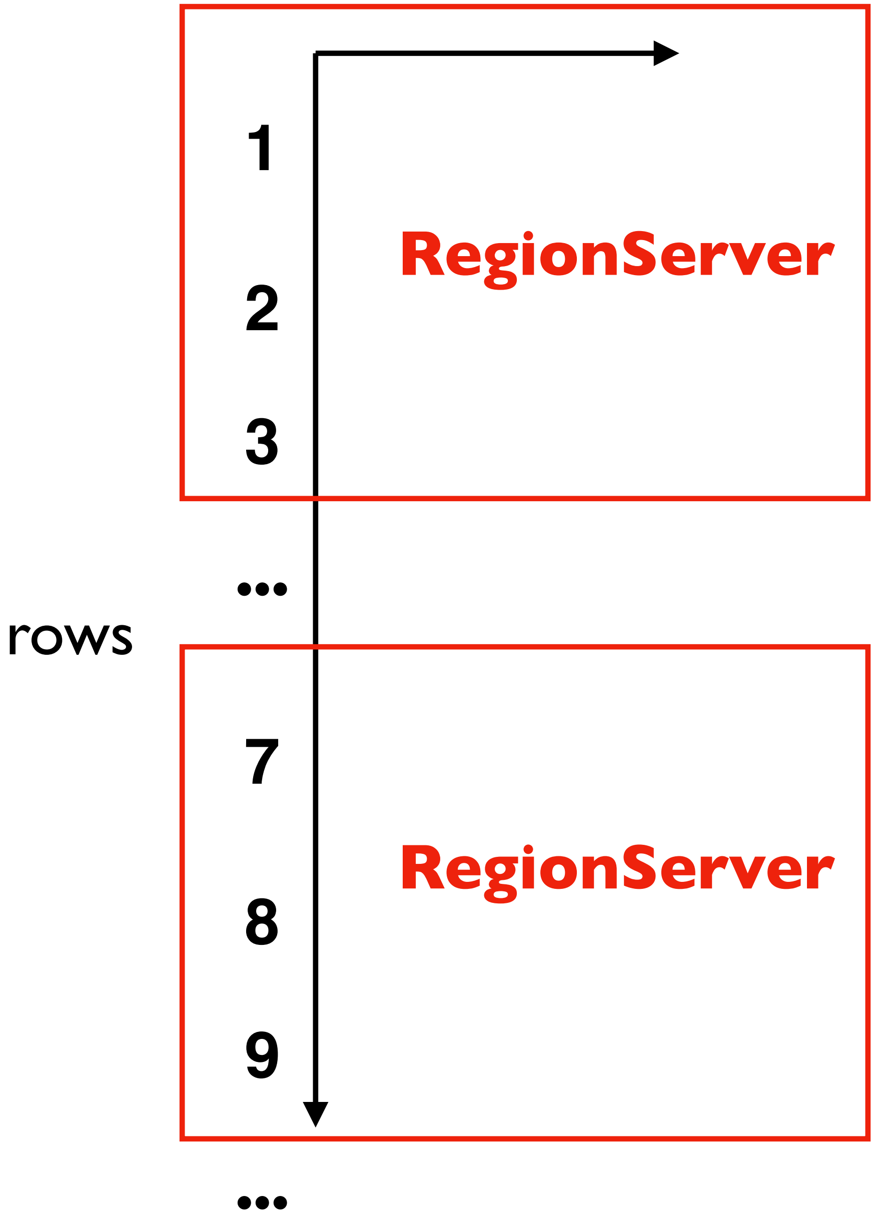
row ranges are called "regions"

regions may grow/split

a region is assigned to ONE HBase "RegionServer" at any given time

RegionServers could server multiple regions

Transactions

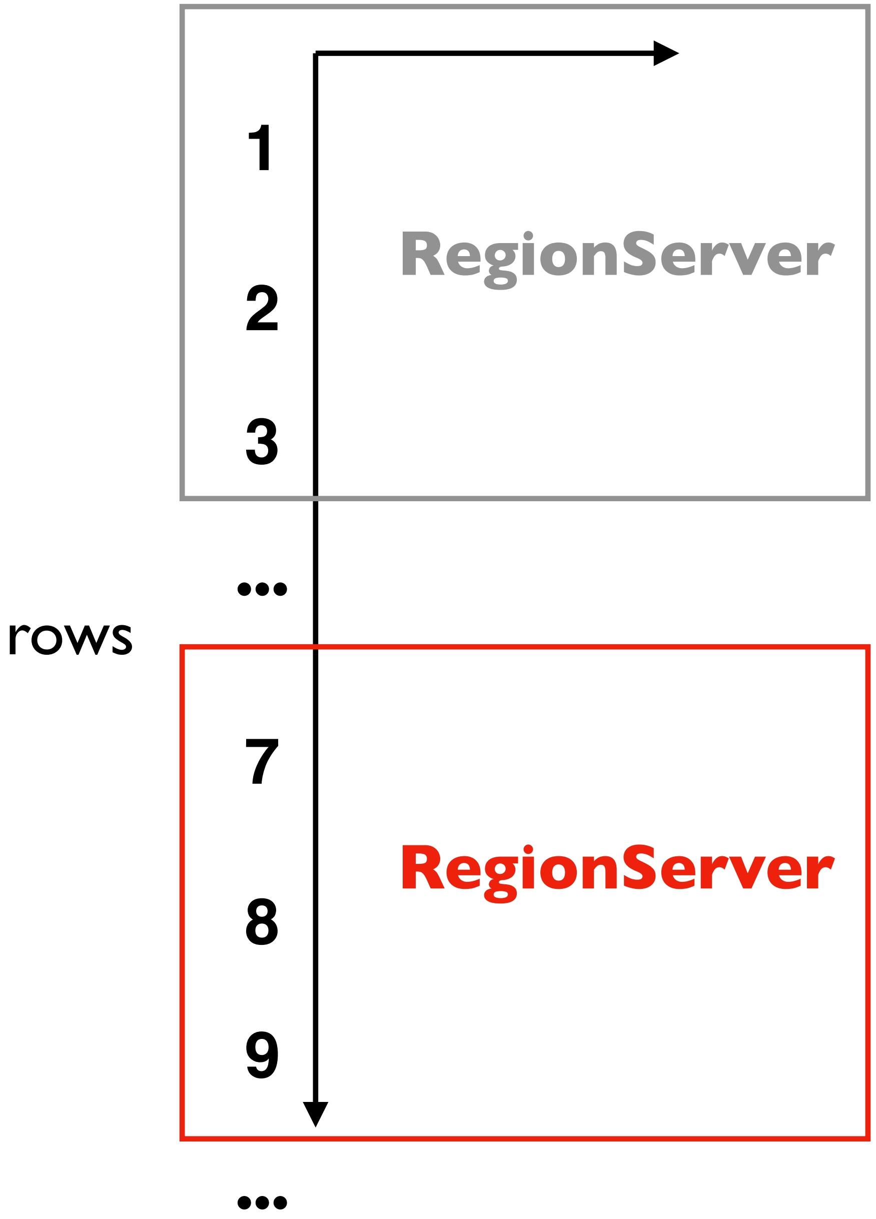


Rows are never split across regions

HBase only support single-row transactions

Design implication: try to keep all of a user's data in ONE row, even if it means millions of columns

Fault Tolerance: what if a RegionServer dies?



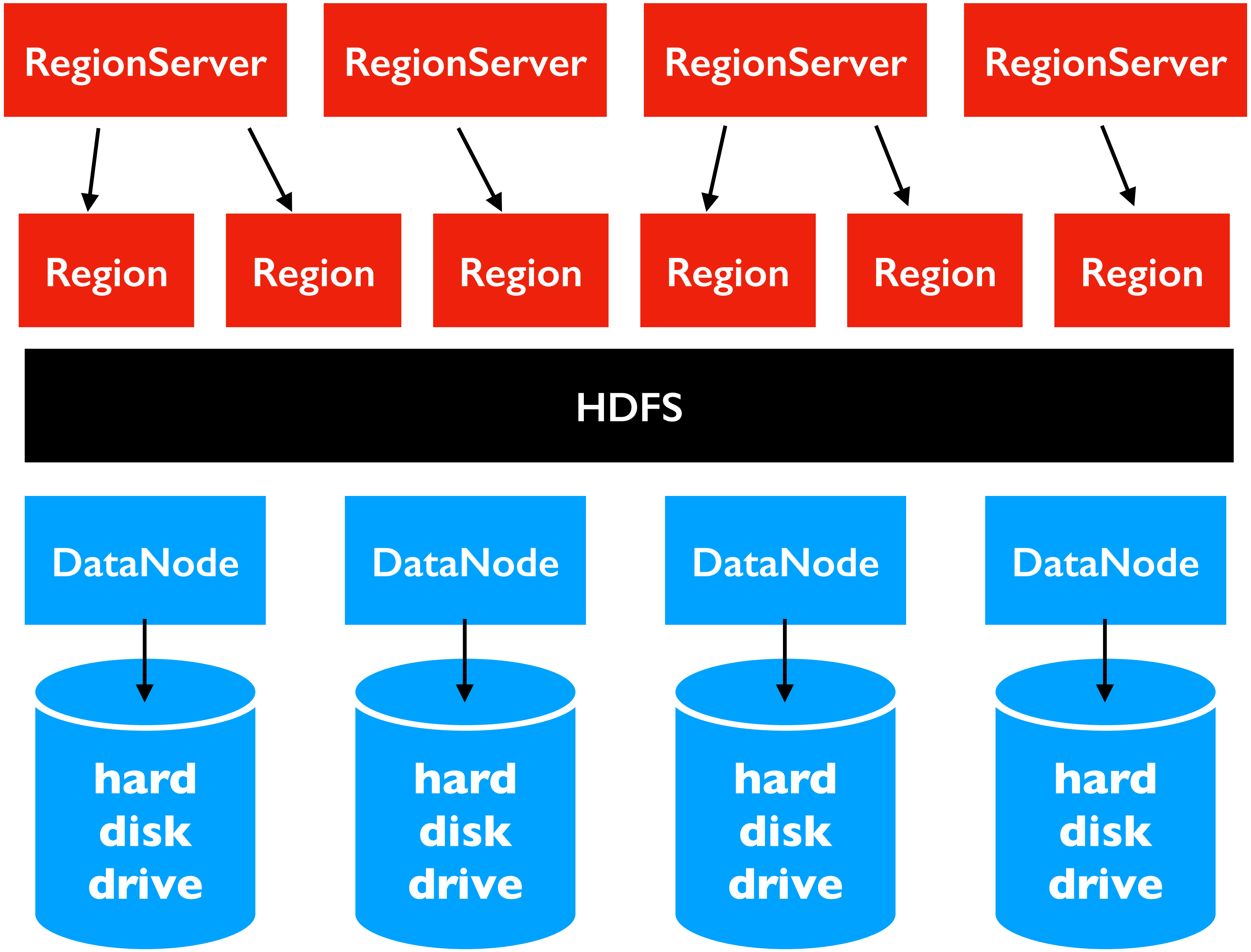
Rows are never split across regions

HBase only support single-row transactions

Design implication: try to keep all of a user's data in ONE row, even if it means millions of columns

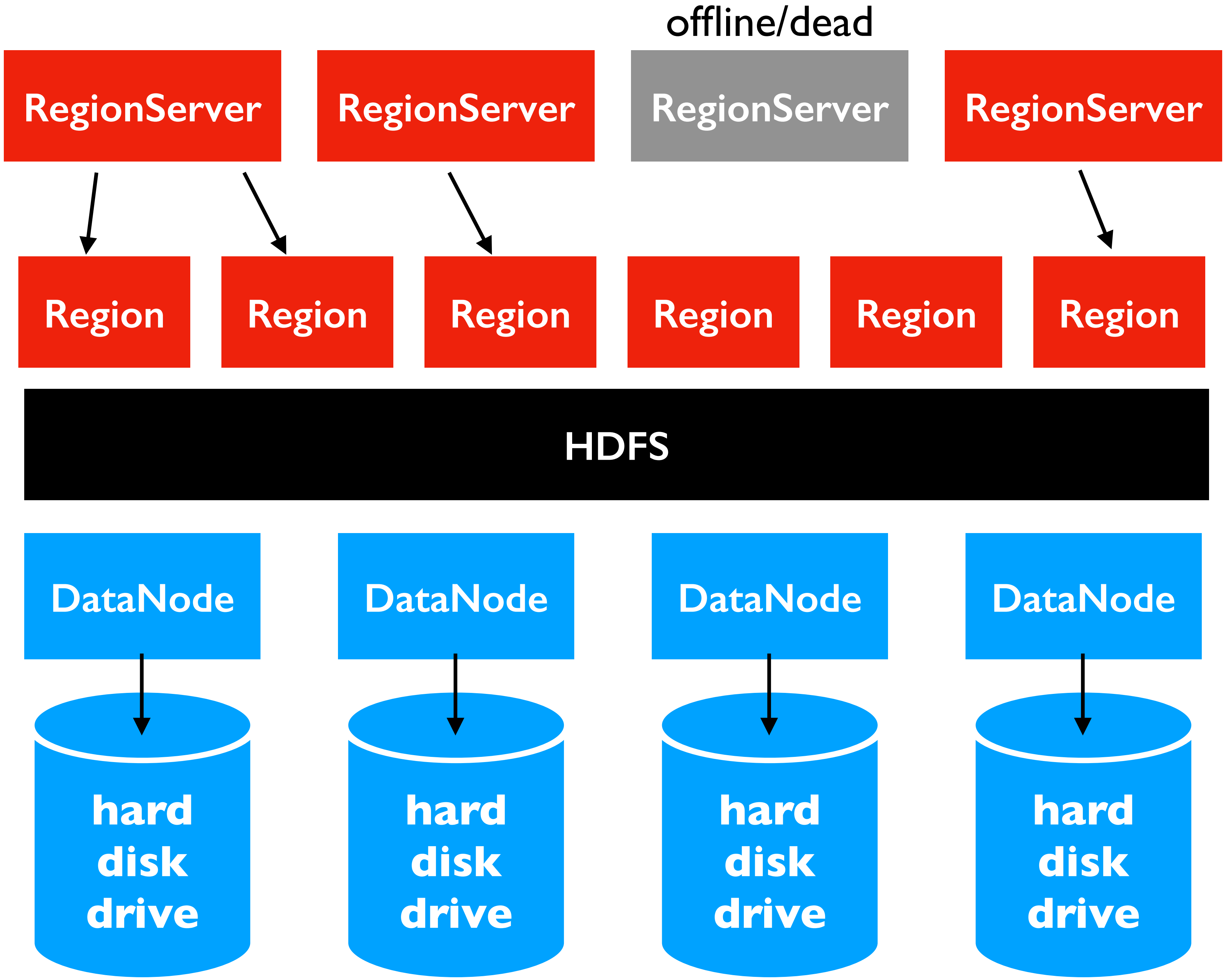
Fault Tolerance: what if a RegionServer dies?

RegionServers store region data inside HDFS files



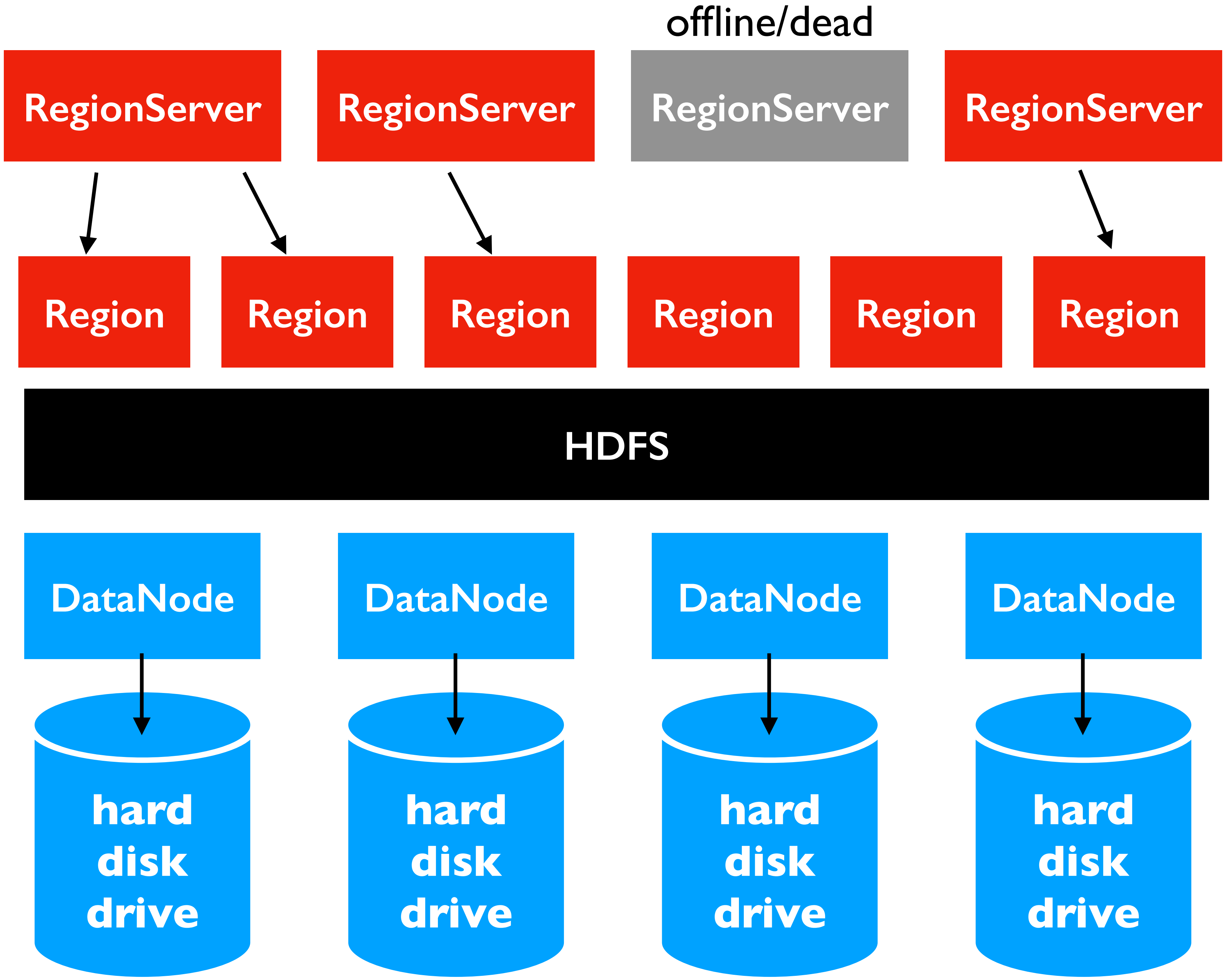
ideally a RegionServer is placed on the same machine as a DataNode holding most of its data

Fault Tolerance: what if a RegionServer dies?



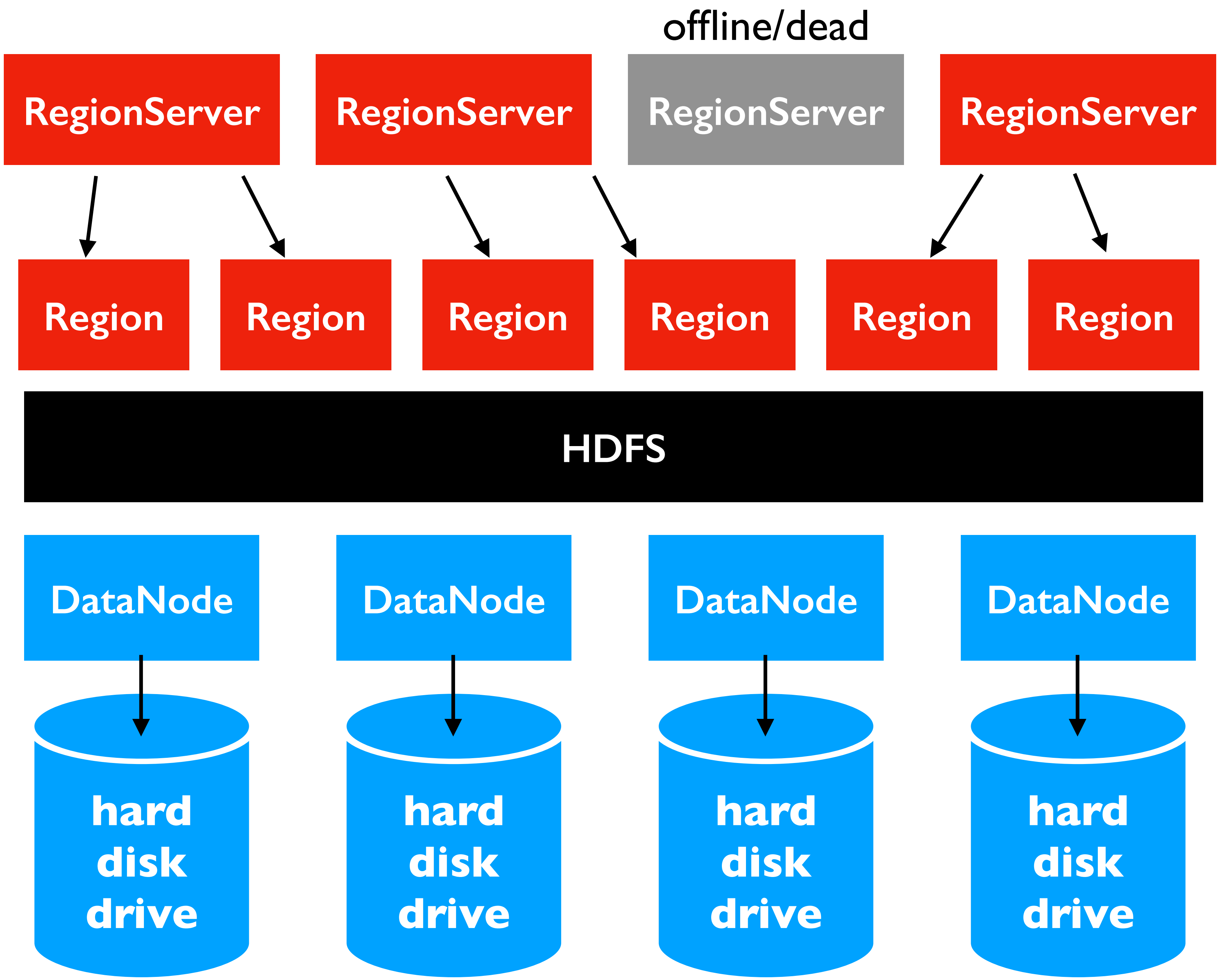
what do do?

Fault Tolerance: what if a RegionServer dies?



region data still safely stored in 3x replicated HDFS files

Fault Tolerance: what if a RegionServer dies?

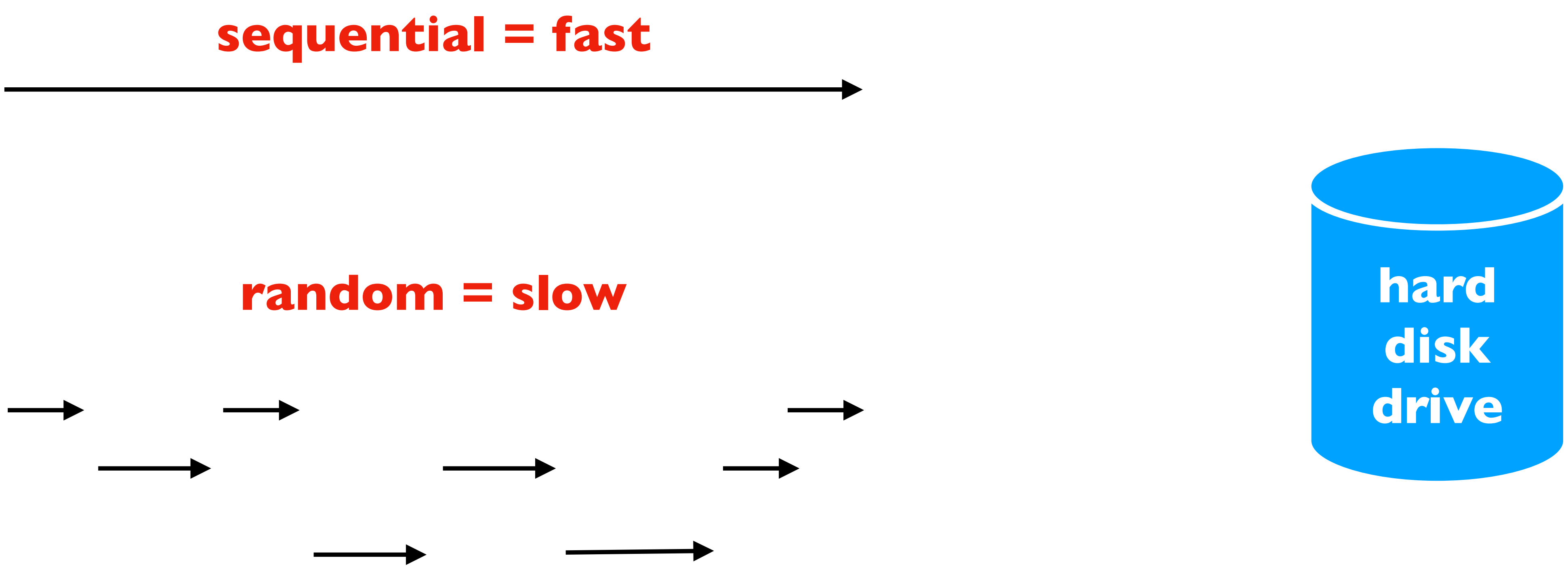


handoff regions to healthy RegionServers

HBase Storage Layout

Observation:

- starting a write operation on disk has a very high fixed cost



sequential = fast

random = slow

hard disk drive

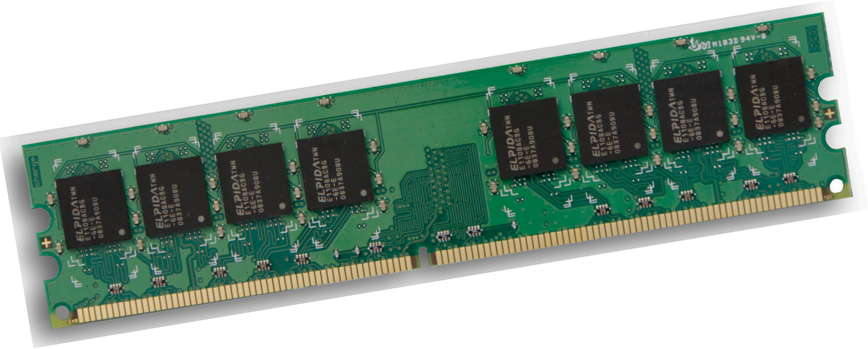
Strategy:

- store new data in memory until we have a lot of data
- then do one big write to disk

HBase Storage Layout

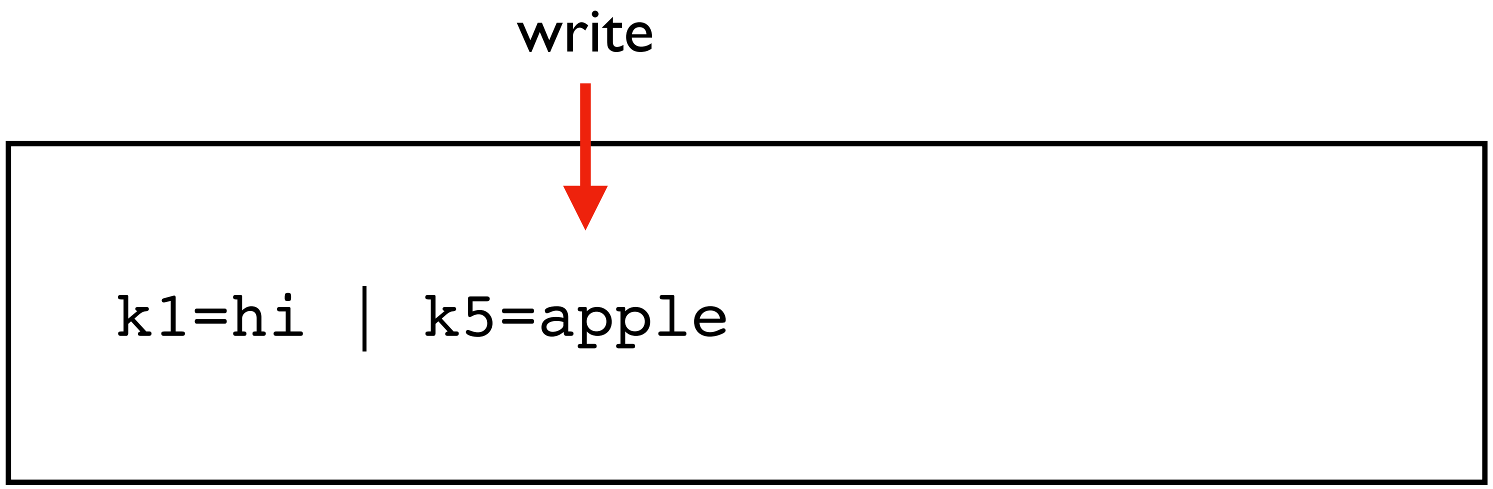
write

k1=hi



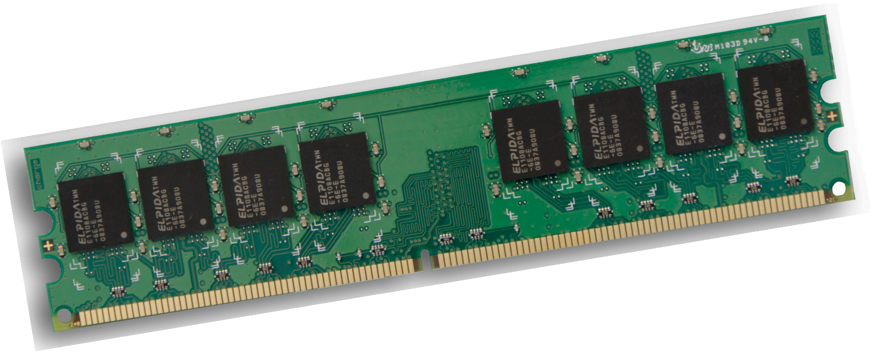
HDFS
(backed by hard drives)

HBase Storage Layout



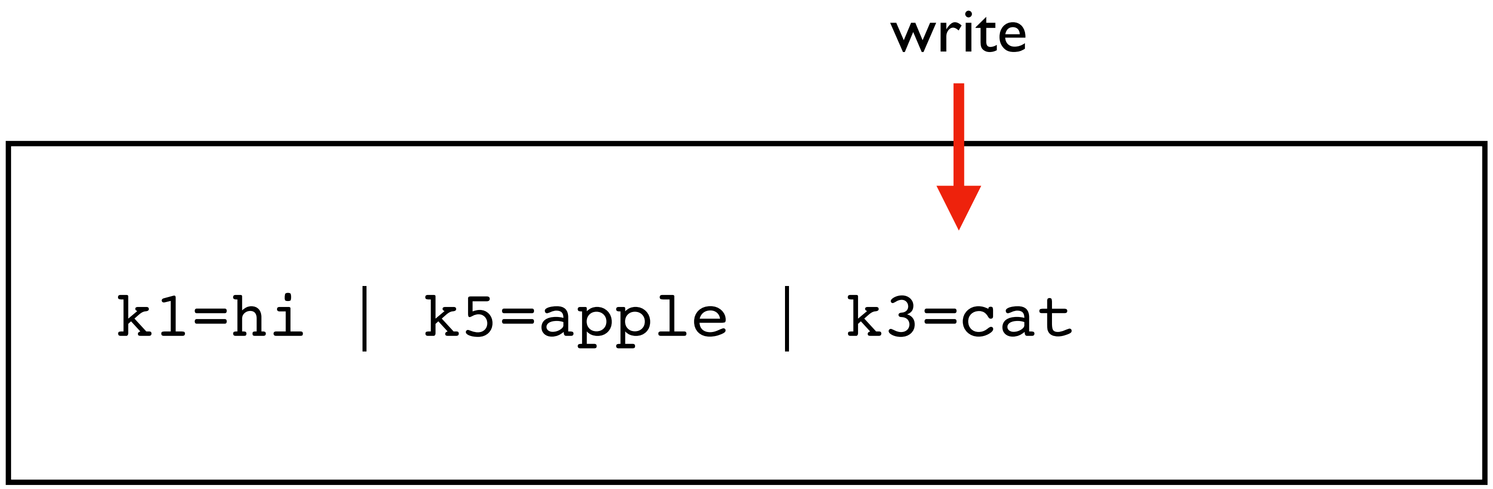
write

```
k1=hi | k5=apple
```



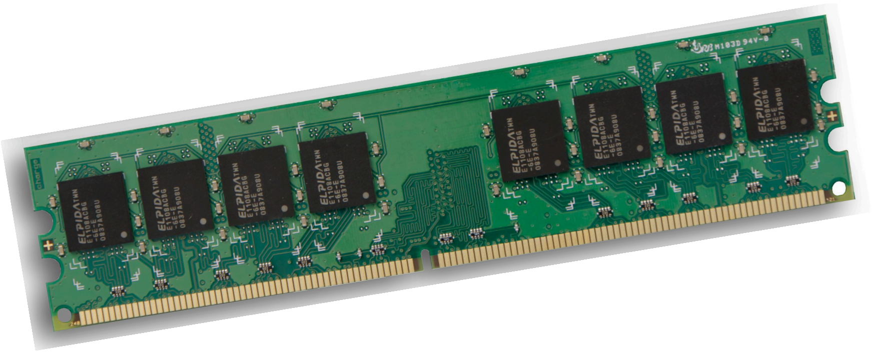
HDFS
(backed by hard drives)

HBase Storage Layout



write

k1=hi | k5=apple | k3=cat

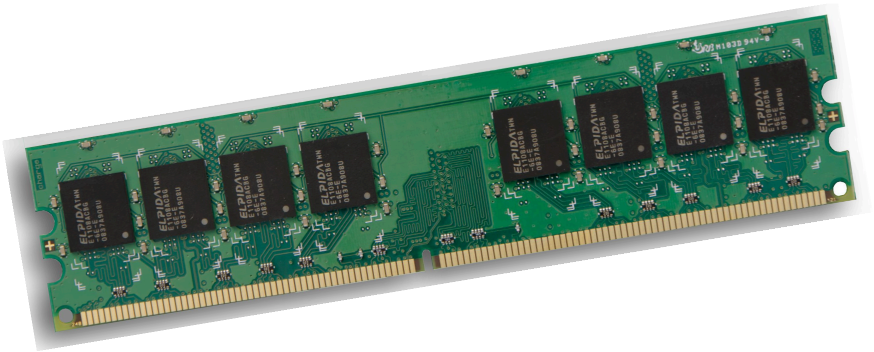


HDFS
(backed by hard drives)

HBase Storage Layout

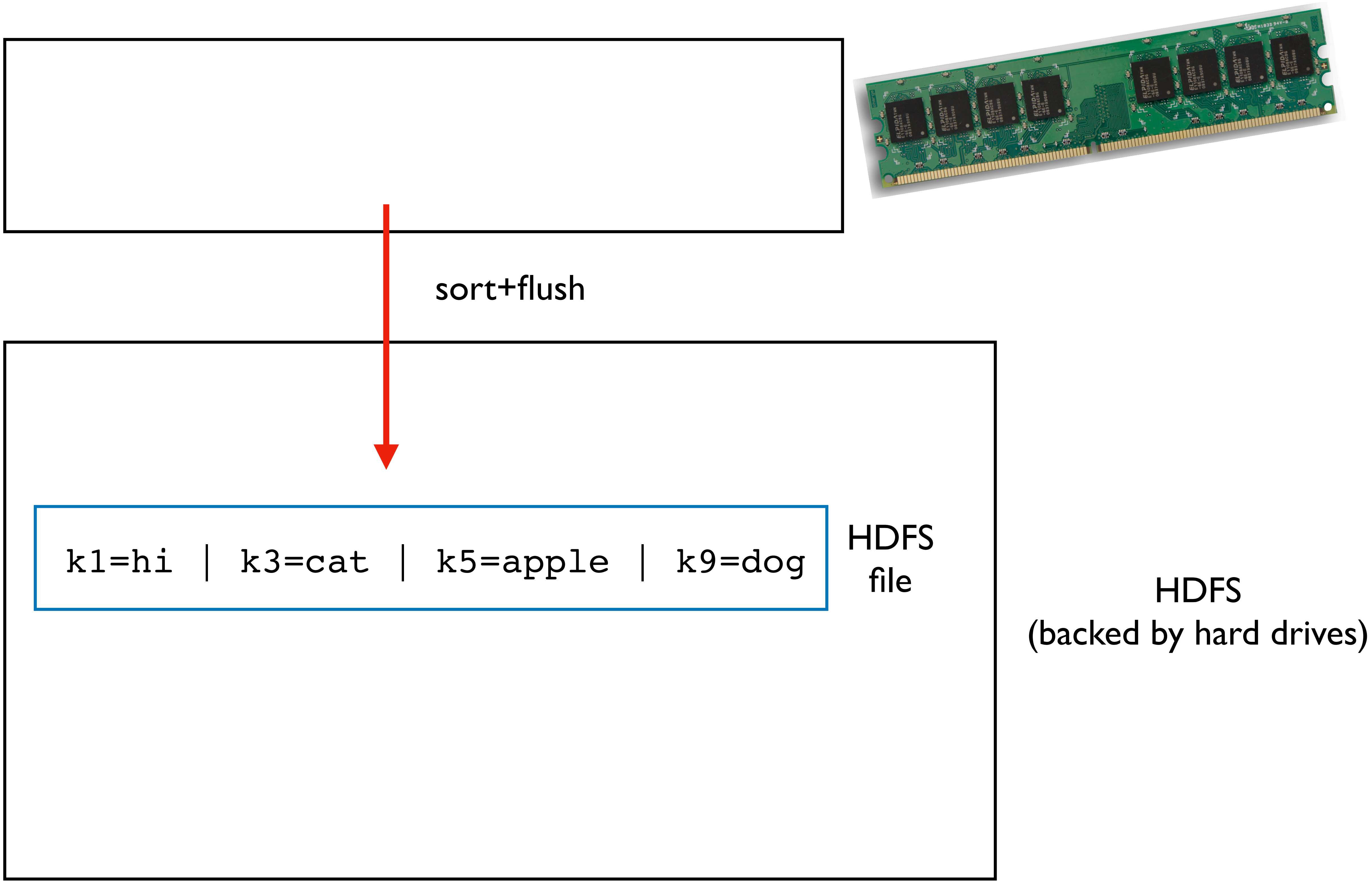
write

k1=hi | k5=apple | k3=cat | k9=dog



HDFS
(backed by hard drives)

HBase Storage Layout



sort+flush

k1=hi | k3=cat | k5=apple | k9=dog

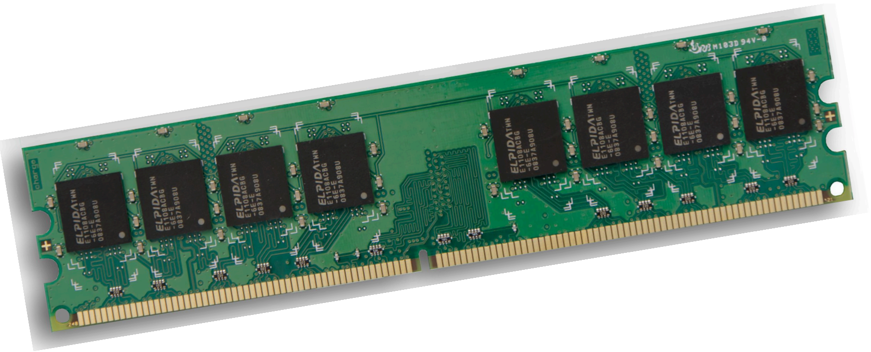
HDFS
file

HDFS
(backed by hard drives)

HBase Storage Layout

write

k6=moon | ...

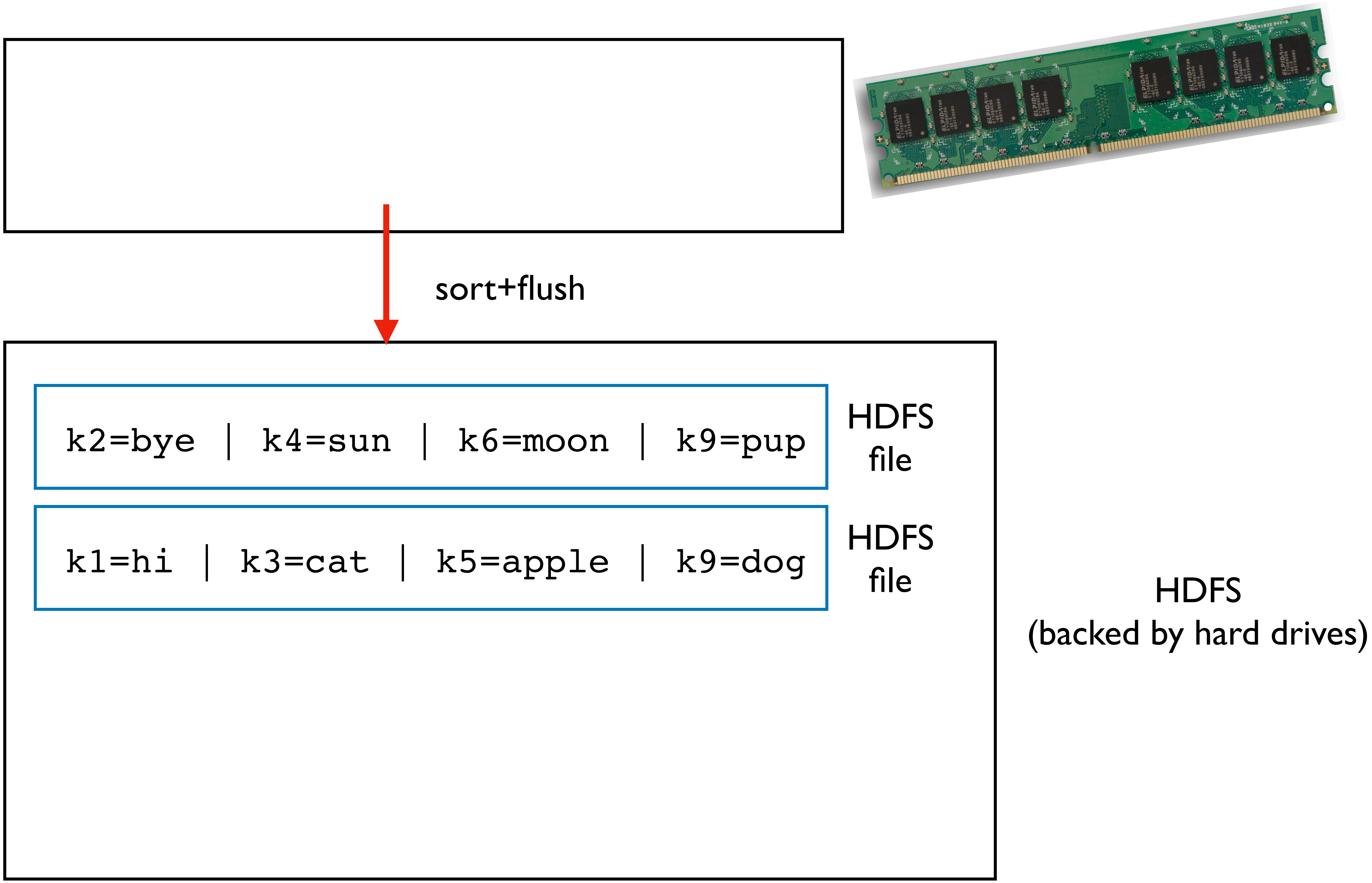


k1=hi | k3=cat | k5=apple | k9=dog

HDFS
file

HDFS
(backed by hard drives)

HBase Storage Layout



sort+flush

k2=bye | k4=sun | k6=moon | k9=pup

HDFS file

k1=hi | k3=cat | k5=apple | k9=dog

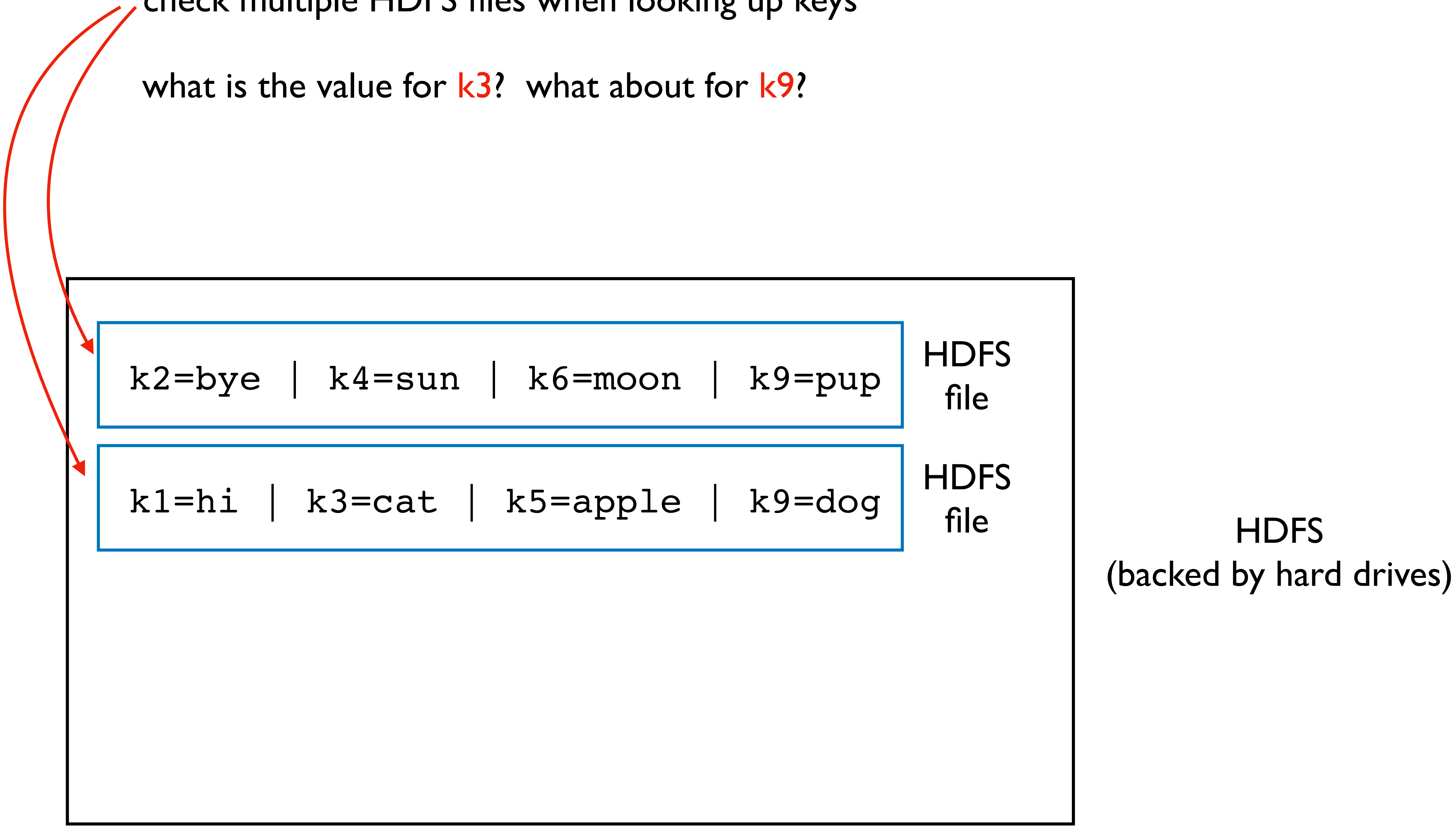
HDFS file

HDFS
(backed by hard drives)

HBase Reads

check multiple HDFS files when looking up keys

what is the value for k3? what about for k9?



k2=bye | k4=sun | k6=moon | k9=pup

HDFS file

k1=hi | k3=cat | k5=apple | k9=dog

HDFS file

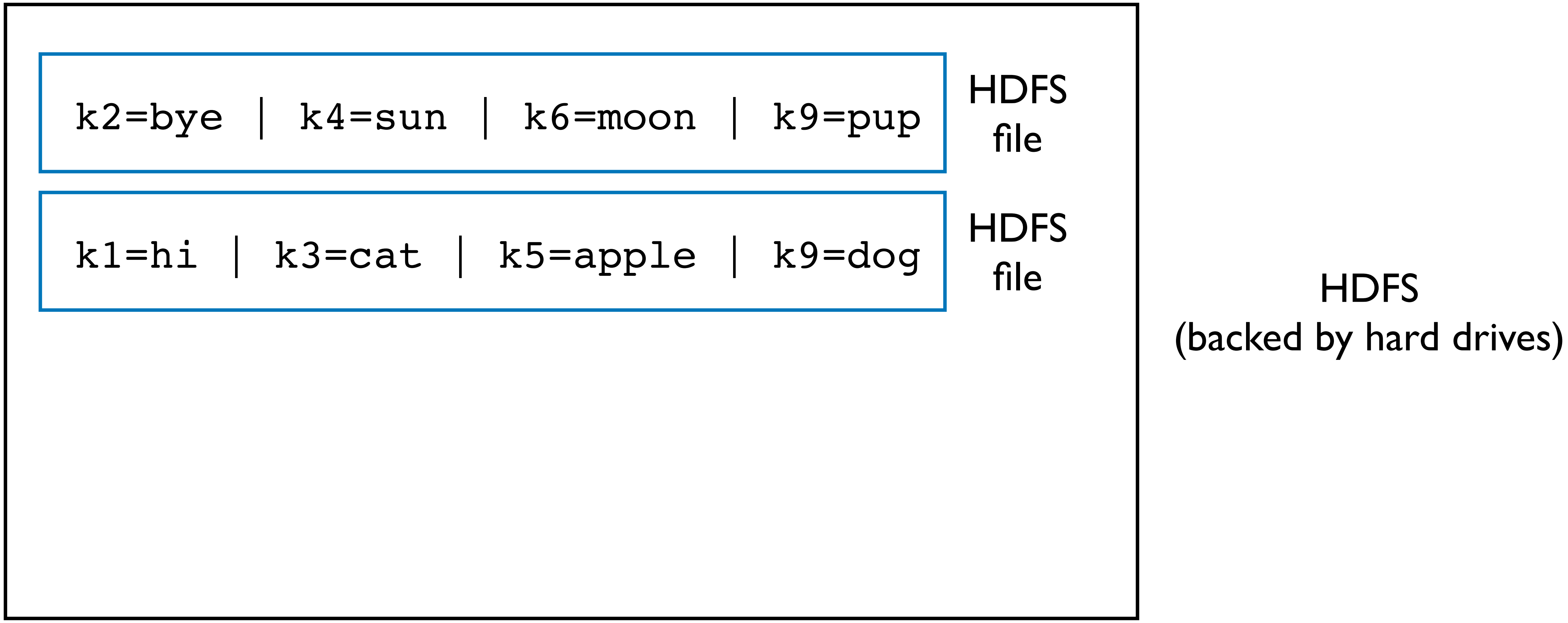
HDFS (backed by hard drives)

HBase Reads

check multiple HDFS files when looking up keys

what is the value for k3? what about for k9?

"tombstones" are used when we delete data
(need to write something -- can't erase old version in finalized file)



k2=bye | k4=sun | k6=moon | k9=pup

HDFS file

k1=hi | k3=cat | k5=apple | k9=dog

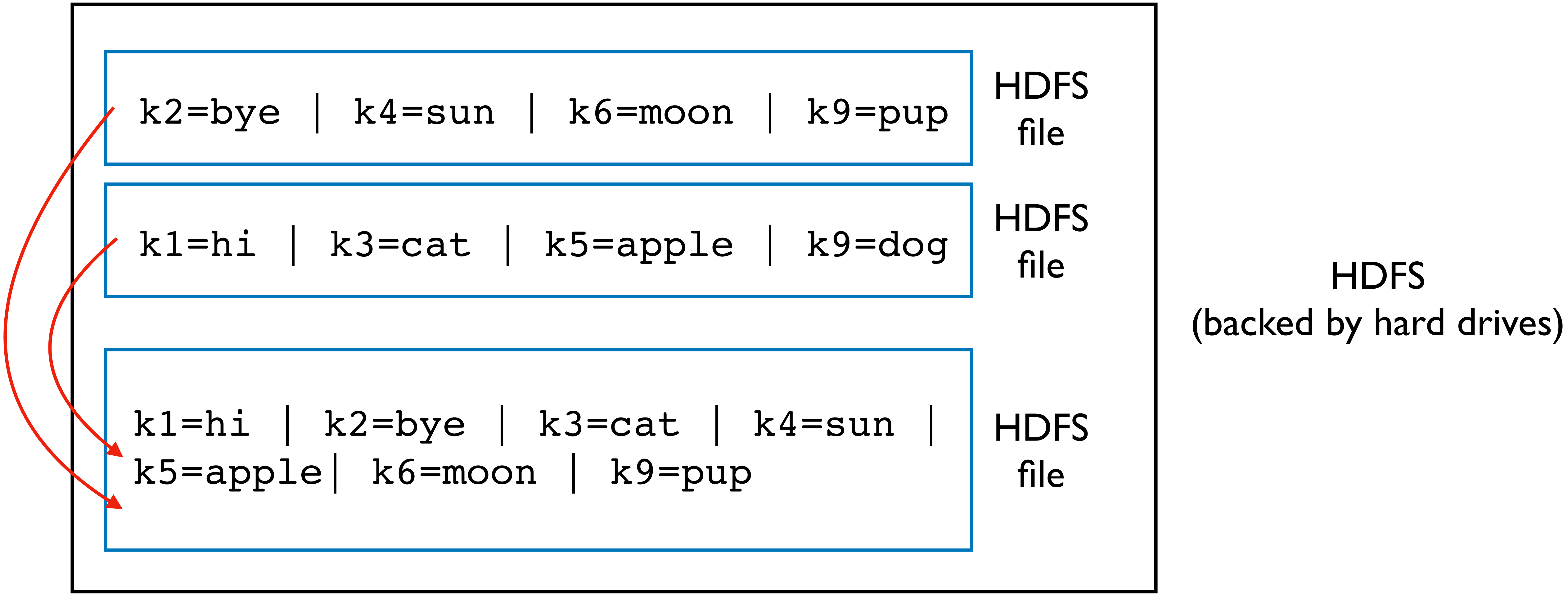
HDFS file

HDFS
(backed by hard drives)

Compaction: Many Small Files => Few Big Files

If there are too many files, reads become too slow.

Solution: compact/combine smaller files into bigger files



```
k2=bye | k4=sun | k6=moon | k9=pup
```

HDFS file

```
k1=hi | k3=cat | k5=apple | k9=dog
```

HDFS file

```
k1=hi | k2=bye | k3=cat | k4=sun |
k5=apple| k6=moon | k9=pup
```

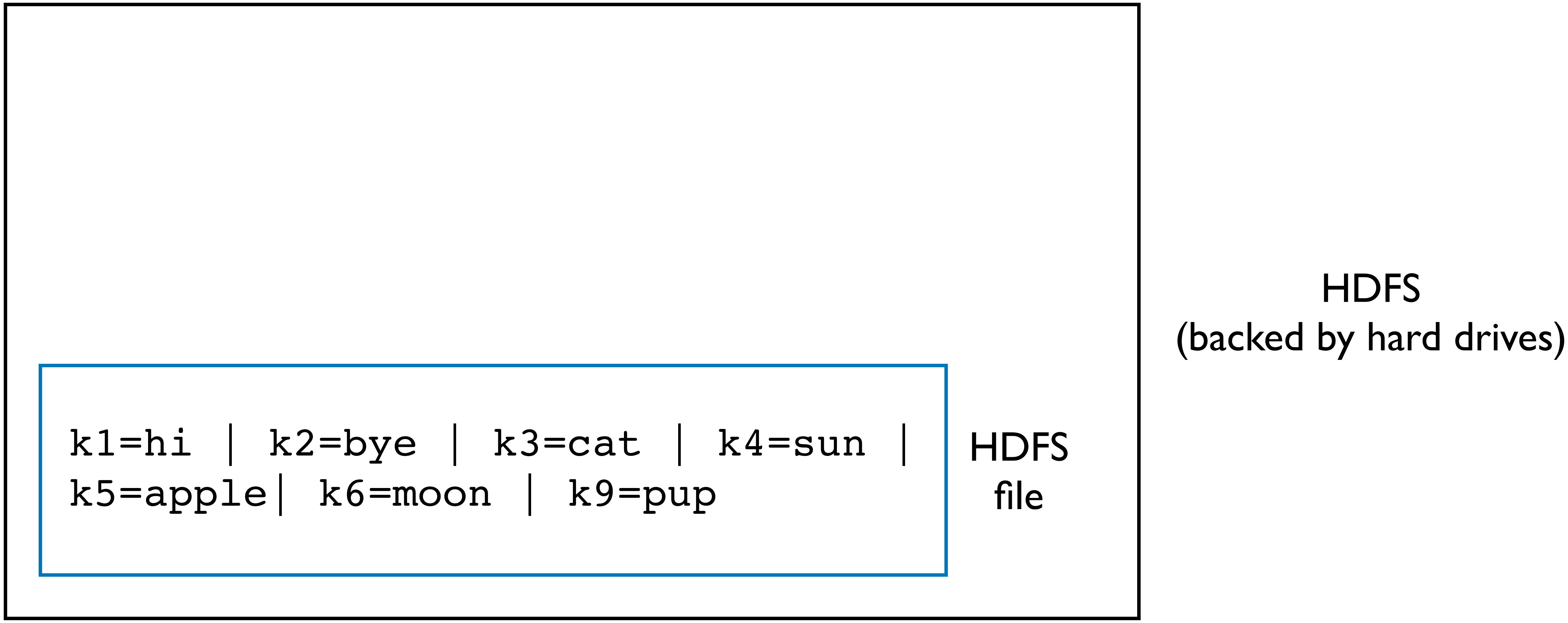
HDFS file

HDFS
(backed by hard drives)

Compaction: Many Small Files => Few Big Files

If there are too many files, reads become too slow.

Solution: compact/combine smaller files into bigger files



```
k1=hi | k2=bye | k3=cat | k4=sun |
k5=apple| k6=moon | k9=pup
```

HDFS file

HDFS (backed by hard drives)

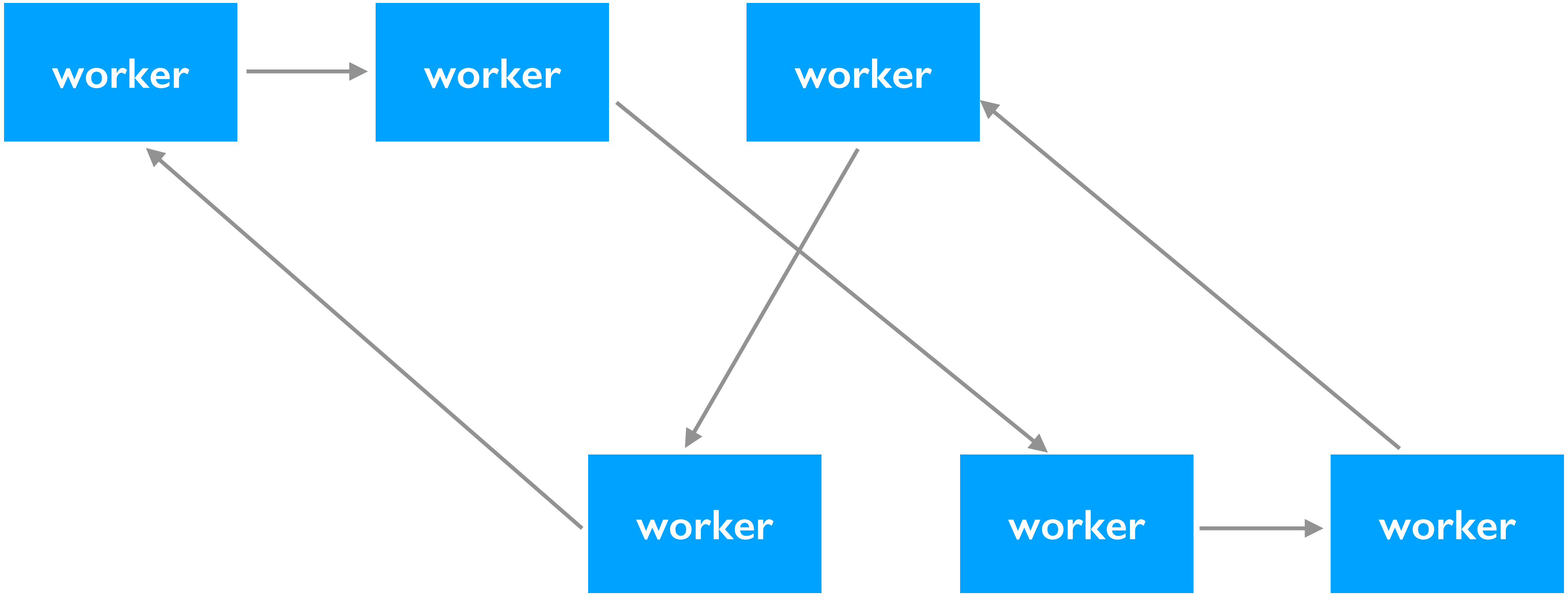
Outline: HBase and Cassandra

HBase

Cassandra Data Model

Demos

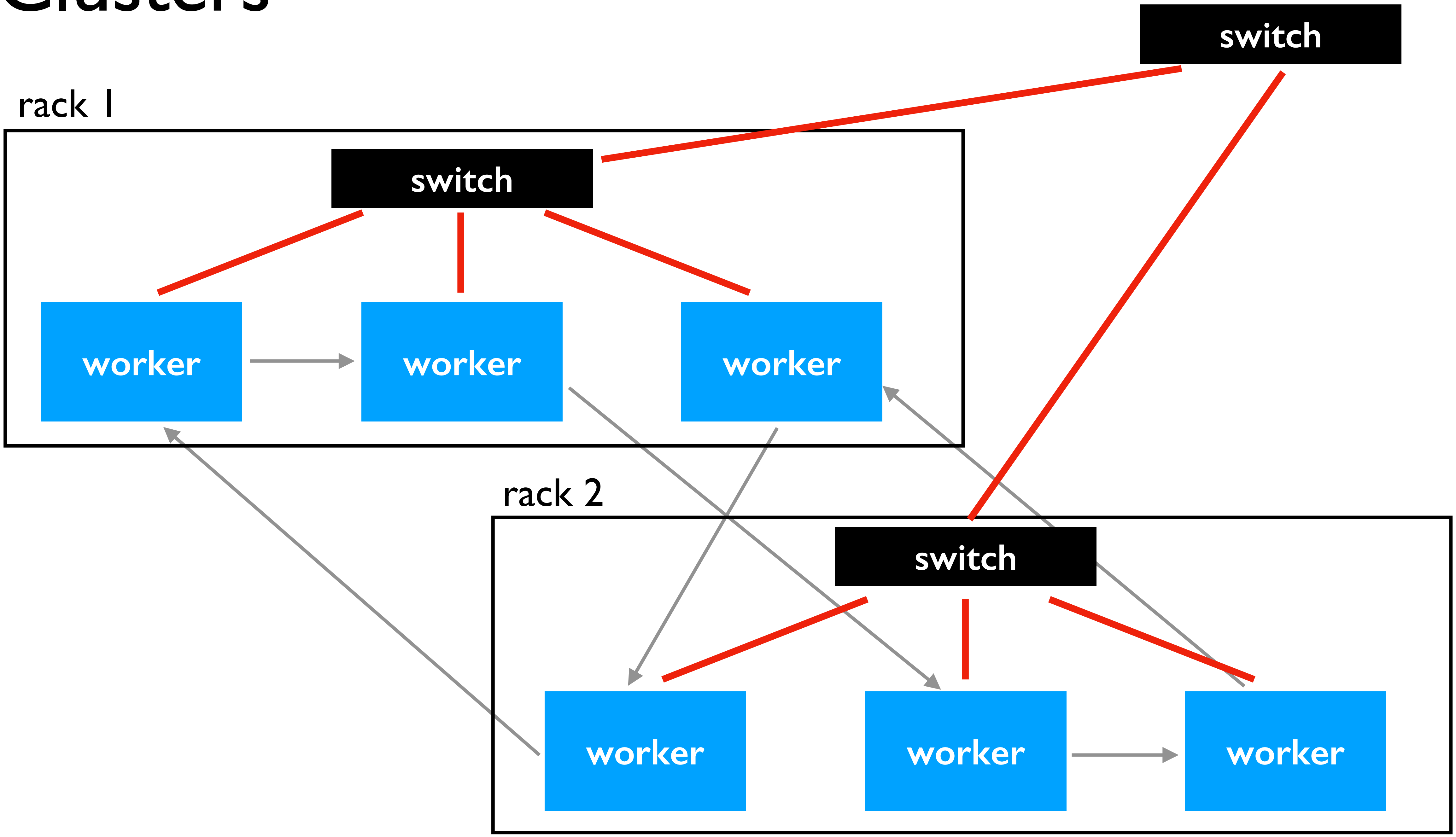
Clusters



Cassandra clusters have many worker nodes

- **No** centralized boss node (unlike HDFS, Spark)
- Not necessarily same data center (could be geographically distributed)
- Clusters are called "rings" because some nodes are defined to be "adjacent"

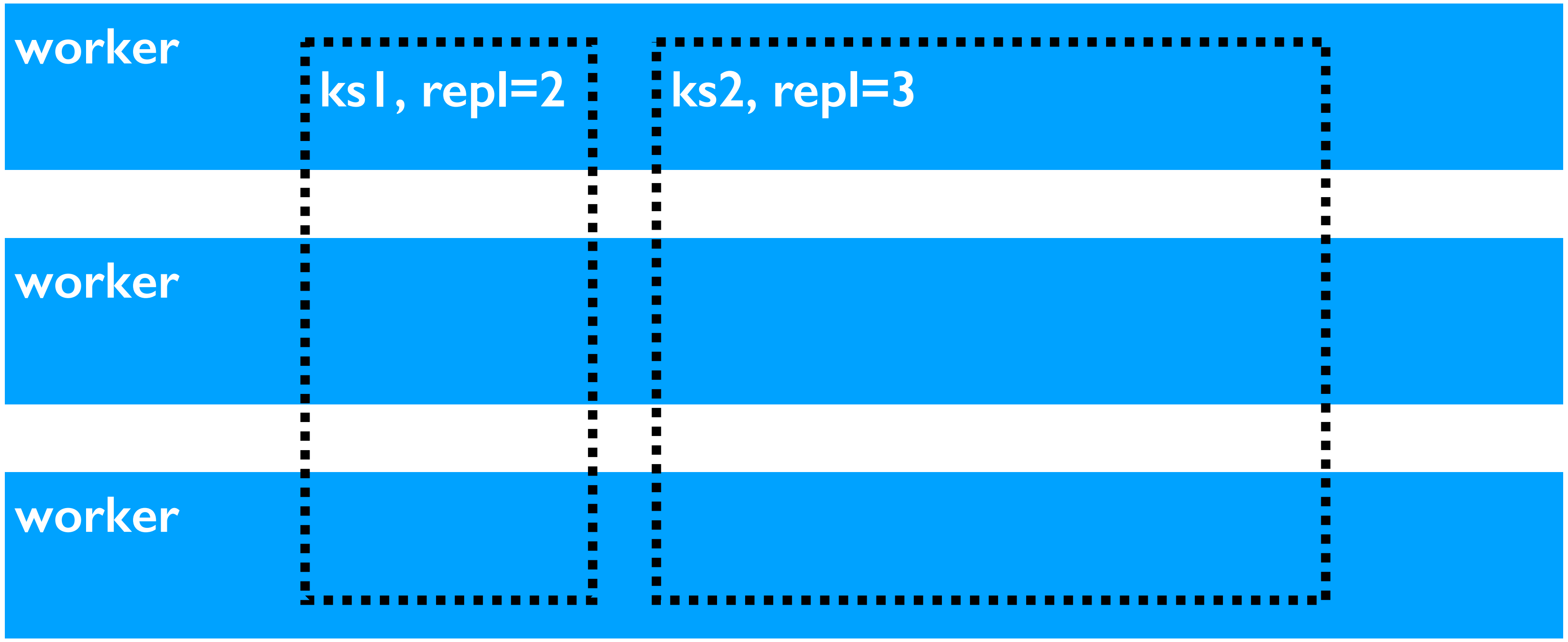
Clusters



Cassandra clusters have many worker nodes

- **No** centralized boss node (unlike HDFS, Spark)
- Not necessarily same data center (could be geographically distributed)
- Clusters are called "rings" because some nodes are defined to be "adjacent"
- Ring organization doesn't necessarily correspond to network topology

Keyspaces



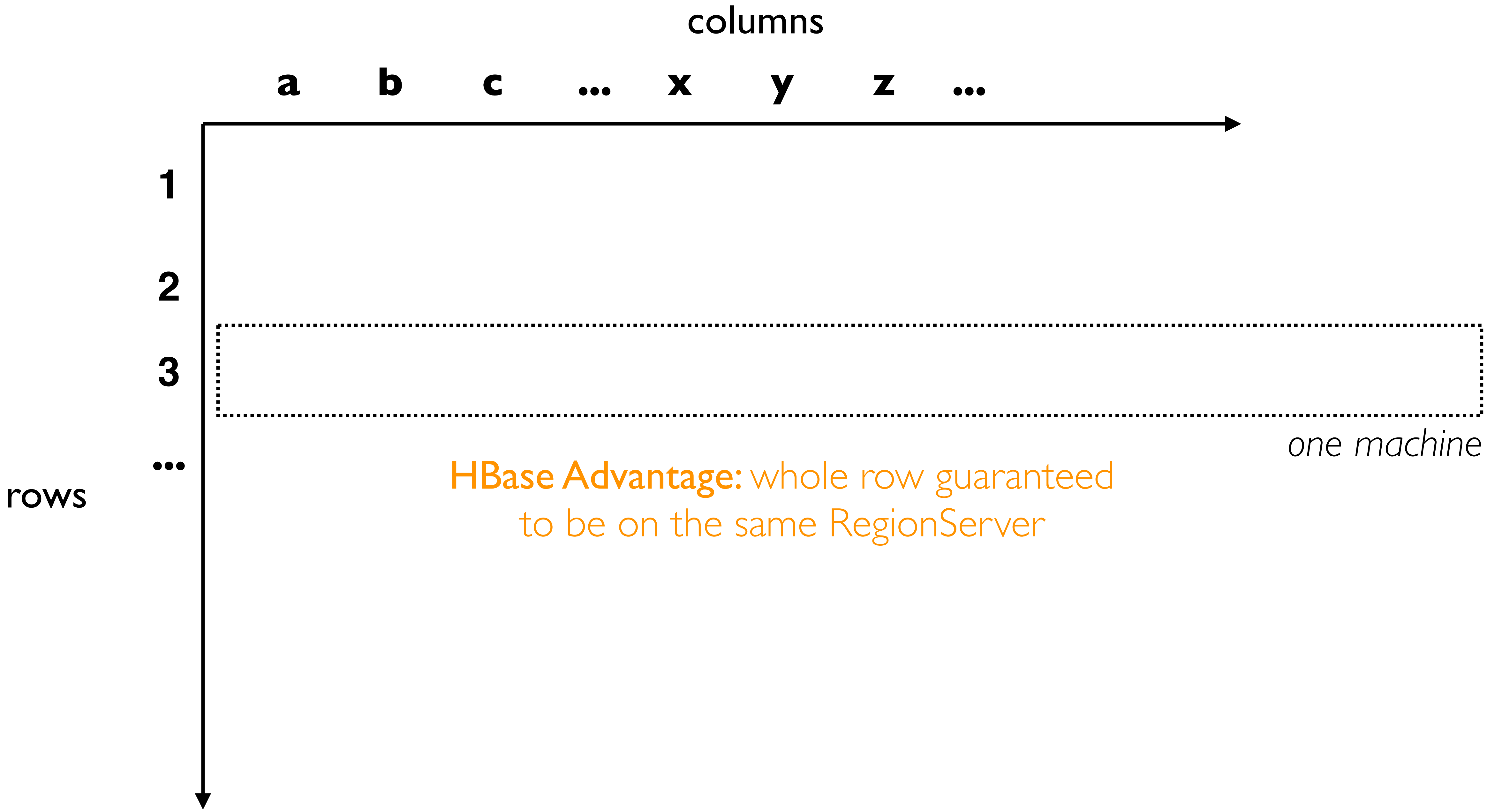
Keyspaces

- similar to databases on database servers
- keyspaces store data across many workers
- different keyspaces can have different replication settings

Each keyspace might contain many tables.

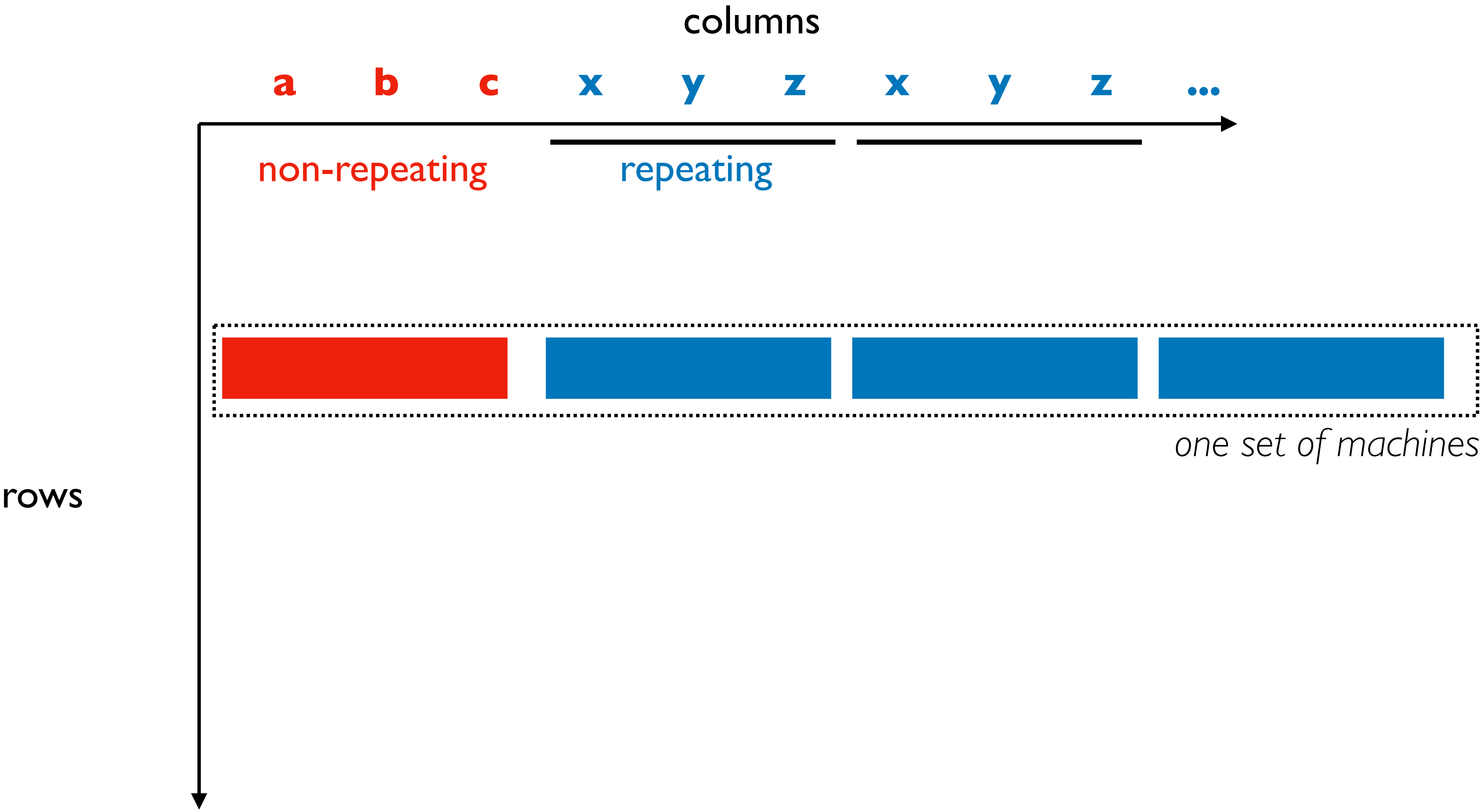
HBase: "Wide Row" design

HBase Disadvantage: no efficient way to even know about all the columns (only about "column families"). SQL-like query languages not easily implemented

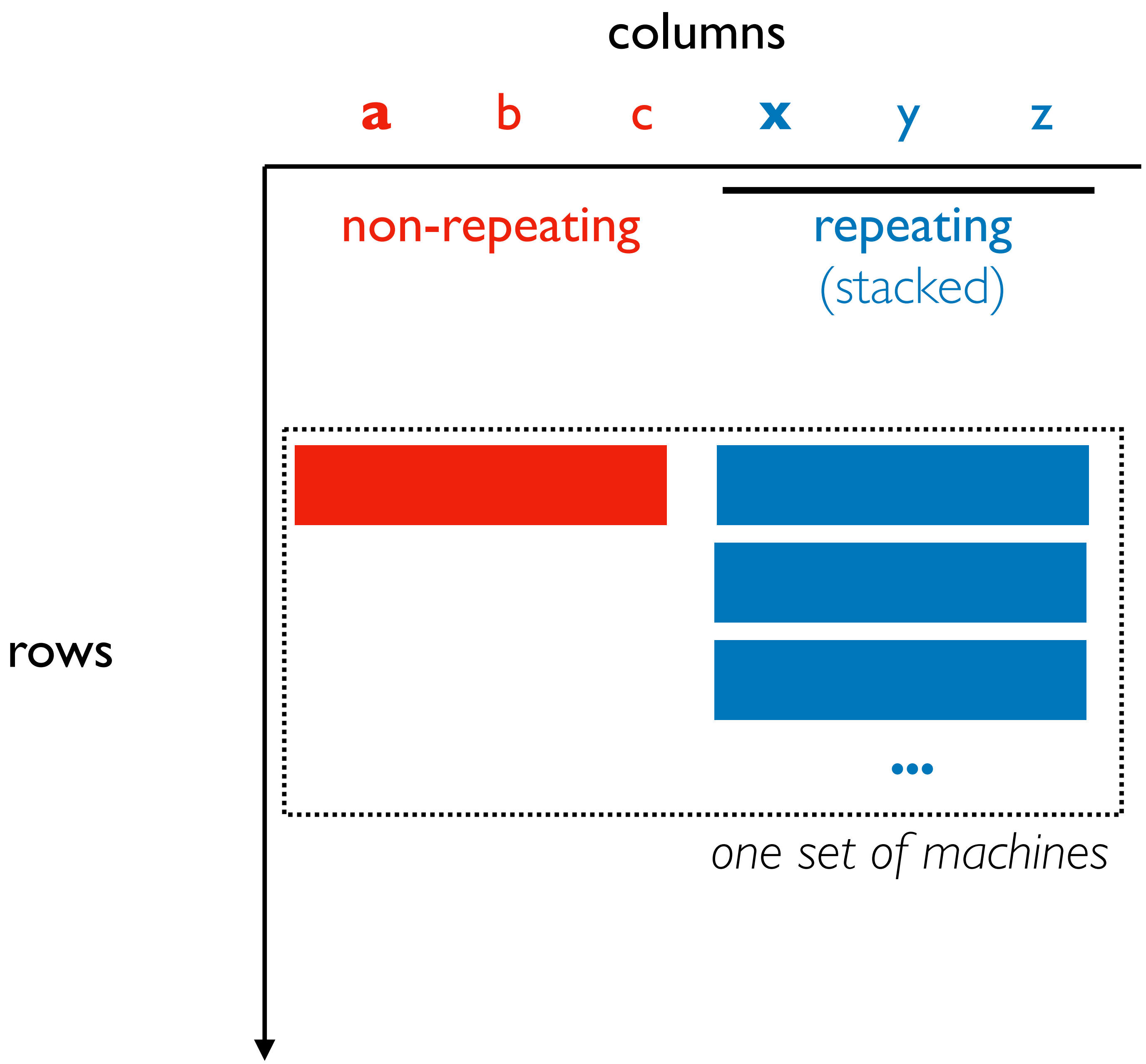


HBase Advantage: whole row guaranteed to be on the same RegionServer

Cassandra: "Wide Partition" design



Cassandra: "Wide Partition" design



Advantages

- finite columns, so can use SQL-like queries: Cassandra Query Language (CQL)
- can keep related data on same machines

Disadvantages

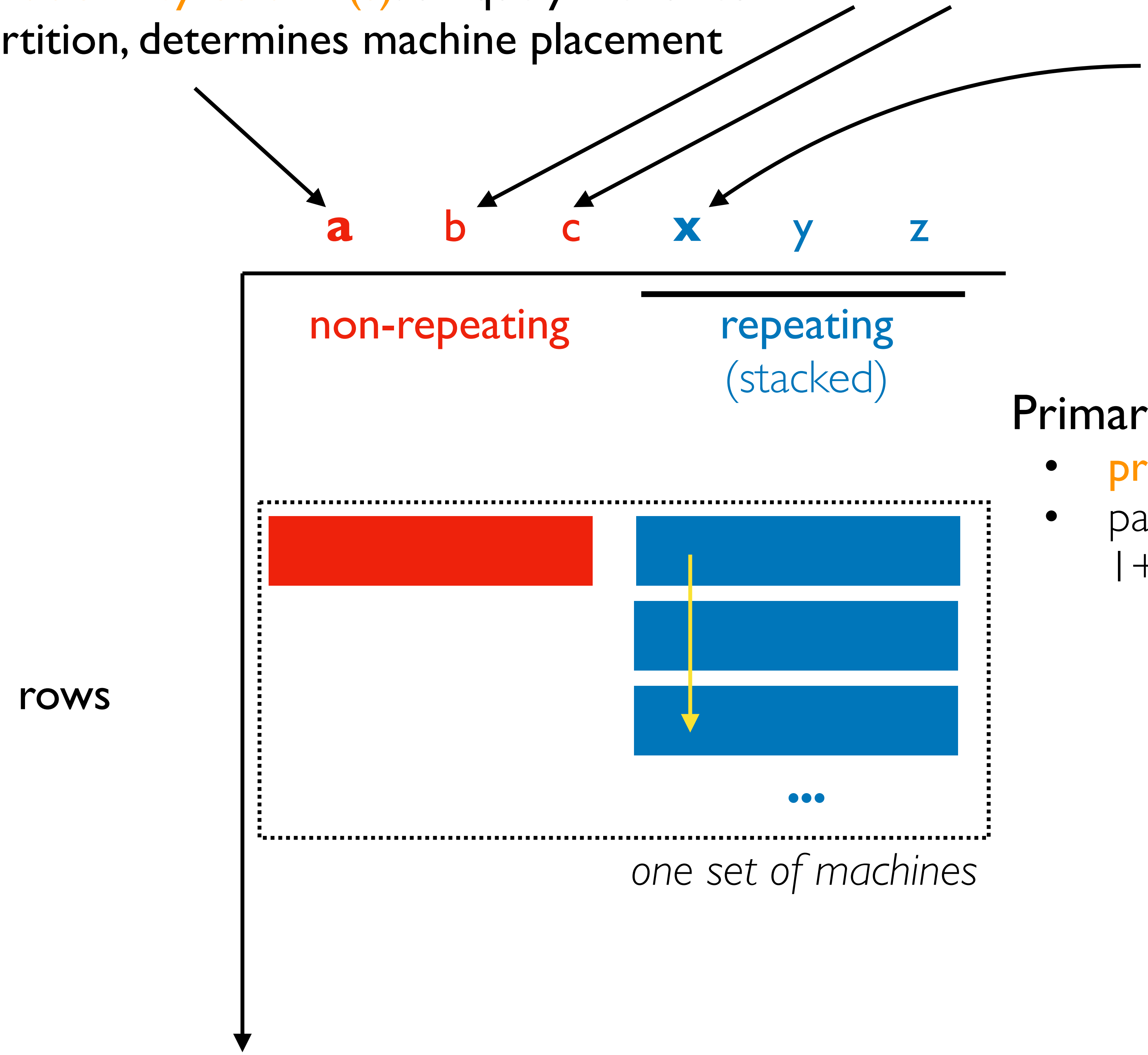
- big partitions: imbalanced storage
- hot partitions: other imbalance

Cassandra: "Wide Partition" design

partition key column(s): uniquely identifies partition, determines machine placement

static columns: one value per partition key

cluster column(s): determines sort order within partition

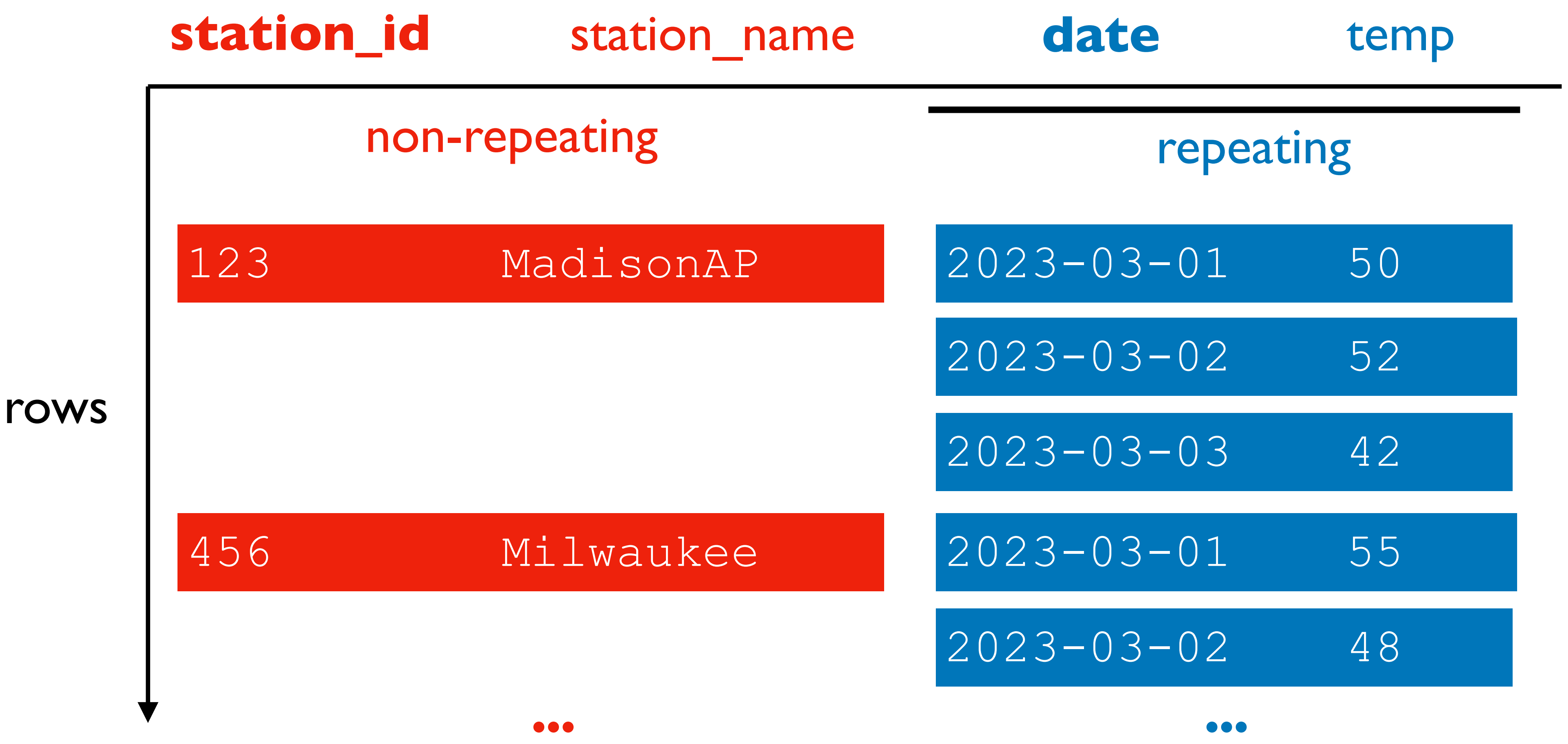


Primary Key Details

- primary key uniquely identifies row
- parts:
  1+ partition keys and 0+ cluster keys

Shema Example: Weather Data

primary key: (station_id, date)



station_id	station_name	date	temp
123	MadisonAP	2023-03-01	50
		2023-03-02	52
		2023-03-03	42
456	Milwaukee	2023-03-01	55
		2023-03-02	48
...		...	

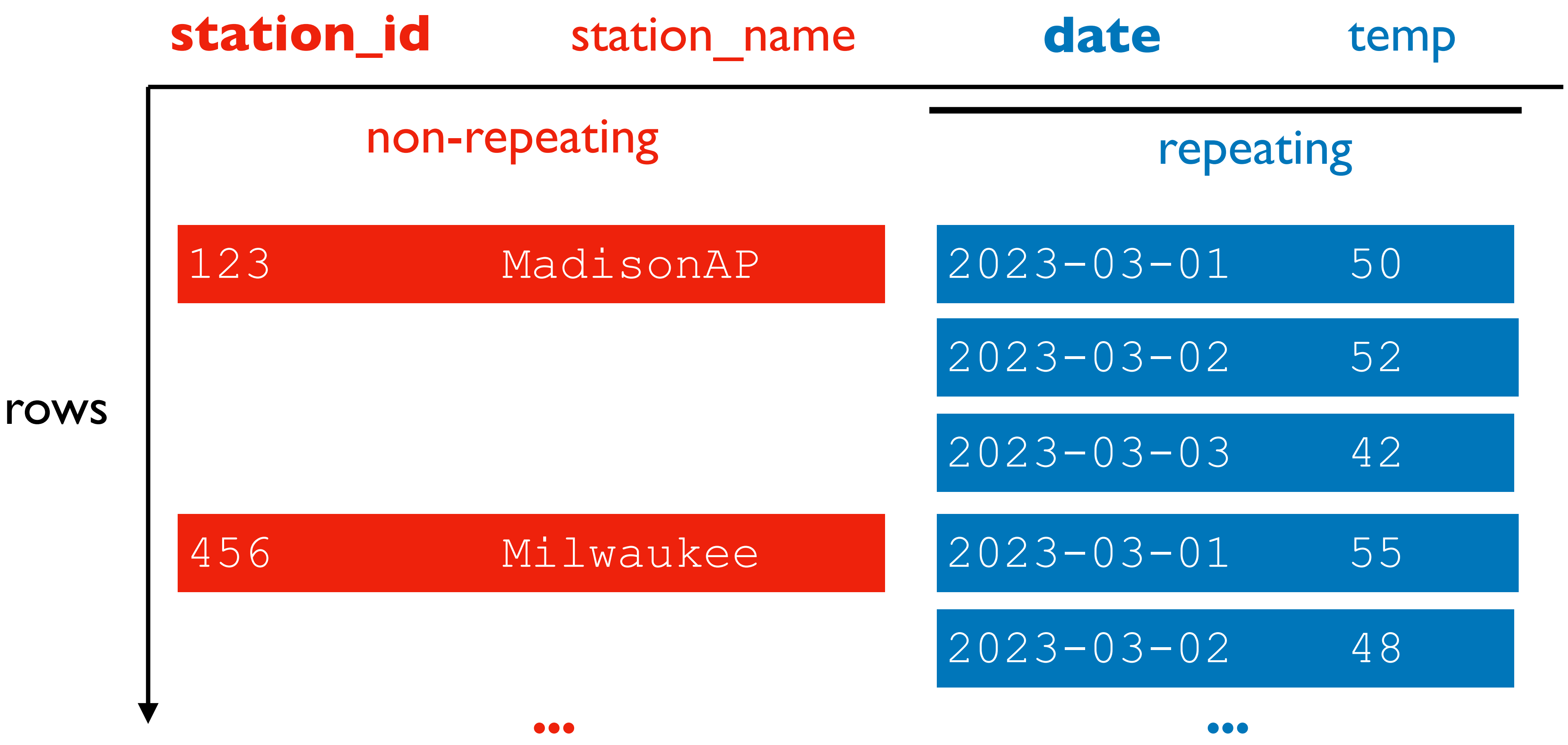
Advantages

- can get all data for one station without scanning the whole cluster (100s of machines)
- looking up dates in a range for a station is fast (pre-sorted)

Challenges

- need to anticipate common queries
- carefully choose partion keys and cluster columns
- too many partitions: queries hit many nodes
- too few partitions: imbalance

Schema Example: Weather Data



non-repeating

repeating

rows

station_id	station_name	date	temp
123	MadisonAP	2023-03-01	50
		2023-03-02	52
		2023-03-03	42
456	Milwaukee	2023-03-01	55
		2023-03-02	48
...		...	

TopHat

Outline: HBase and Cassandra

HBase

Cassandra Data Model

Demos

- Deployment
- cqlsh
- Python (cassandra-driver package)
- Spark (external data source)