Introduction to Hadoop

CS 448 - Relational DB Management Systems

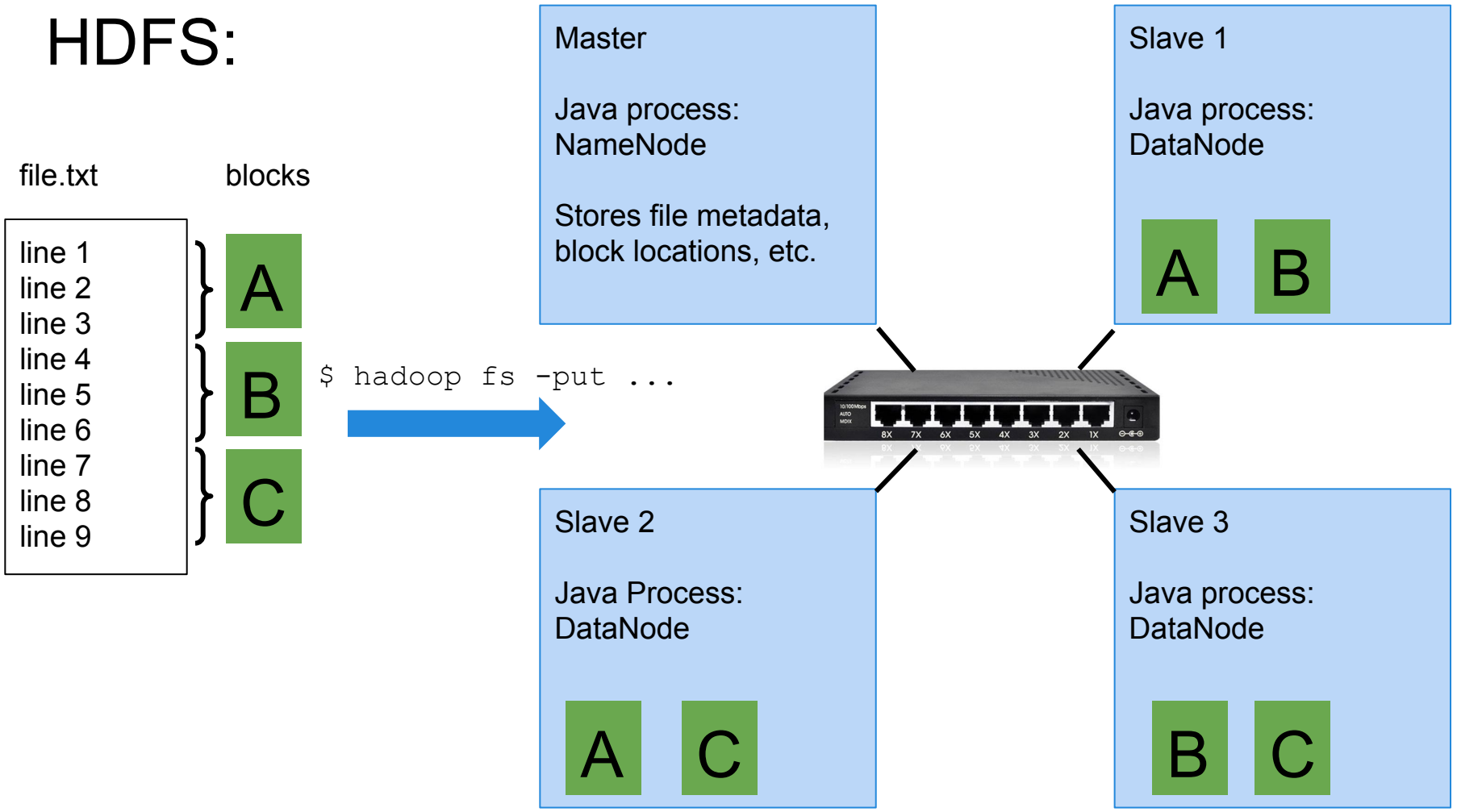
What is Hadoop?

- A collection of tools used to process data that is distributed across a large number of machines (sometimes thousands).
- Written in Java.
- Fault tolerant.
- Two of the most important tools in Hadoop are HDFS and YARN, discussed below.
  - These tools enable MapReduce jobs.
  - A MapReduce job is composed of tasks.

MapReduce jobs run on top of HDFS (Hadoop Distributed File System). What is HDFS?

- A tool for distributing files across a cluster.
- Accepts a file as input. Fragments the file into blocks, duplicates the blocks (for redundancy), and distributes the blocks across the cluster.
- Imitates many of the same functions offered by a local file system: mkdir, rm, cat, etc.

HDFS:


file.txt
blocks
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
A
B
C
$ hadoop fs -put ...
Master
Java process:
NameNode
Stores file metadata,
block locations, etc.
Slave 1
Java process:
DataNode
A
B
Slave 2
Java Process:
DataNode
A
C
Slave 3
Java process:
DataNode
B
C

MapReduce jobs are distributed by YARN. What is YARN (also known as MapReduce version 2)?

- "Yet Another Resource Negotiator."
- Monitors the workload on each cluster node.
- Allocates compute resources.
  - Allocates resources to ApplicationMasters.
  - ApplicationMasters then manage job tasks.
- Supports other workflows in addition to MapReduce (an improvement from MapReduce version 1).

YARN:

MapReduce Job A

MapTask A

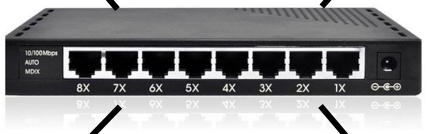
ReduceTask A

MapReduce Job B

MapTask B

ReduceTask B

```
$ hadoop jar ...
```



Master

Java processes:
ResourceManager,
ApplicationMaster1,
ApplicationMaster2

Slave 1

Java process:
NodeManager

MapTask A

ReduceTask A

Slave 2

Java Process:
NodeManager

MapTask A

ReduceTask A

Slave 3

Java process:
NodeManager

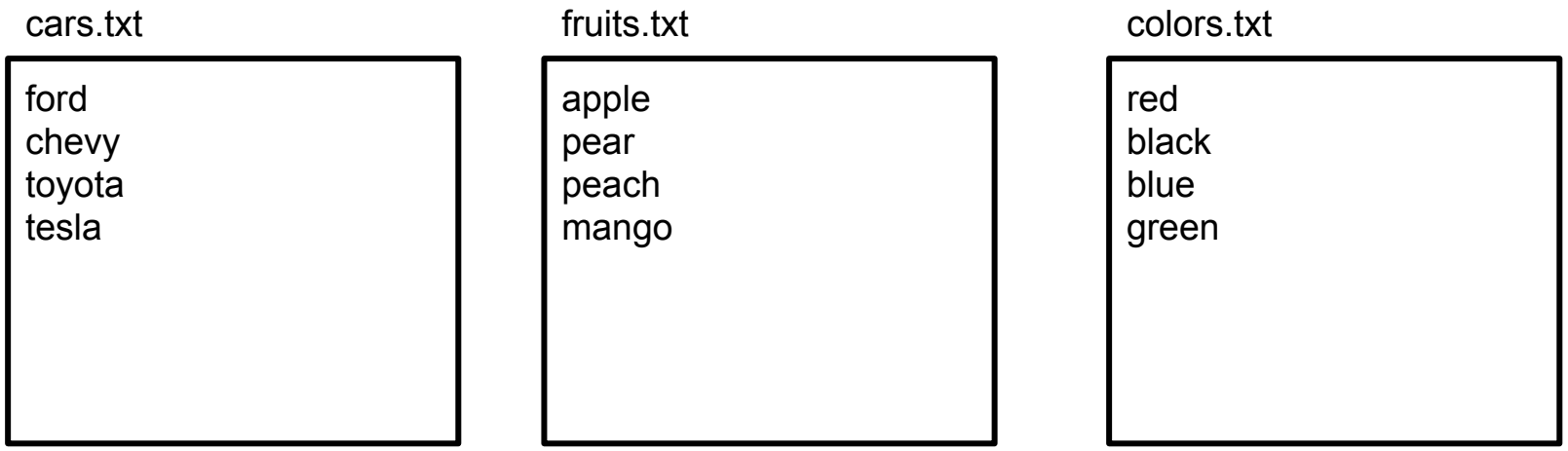
MapTask B

ReduceTask B

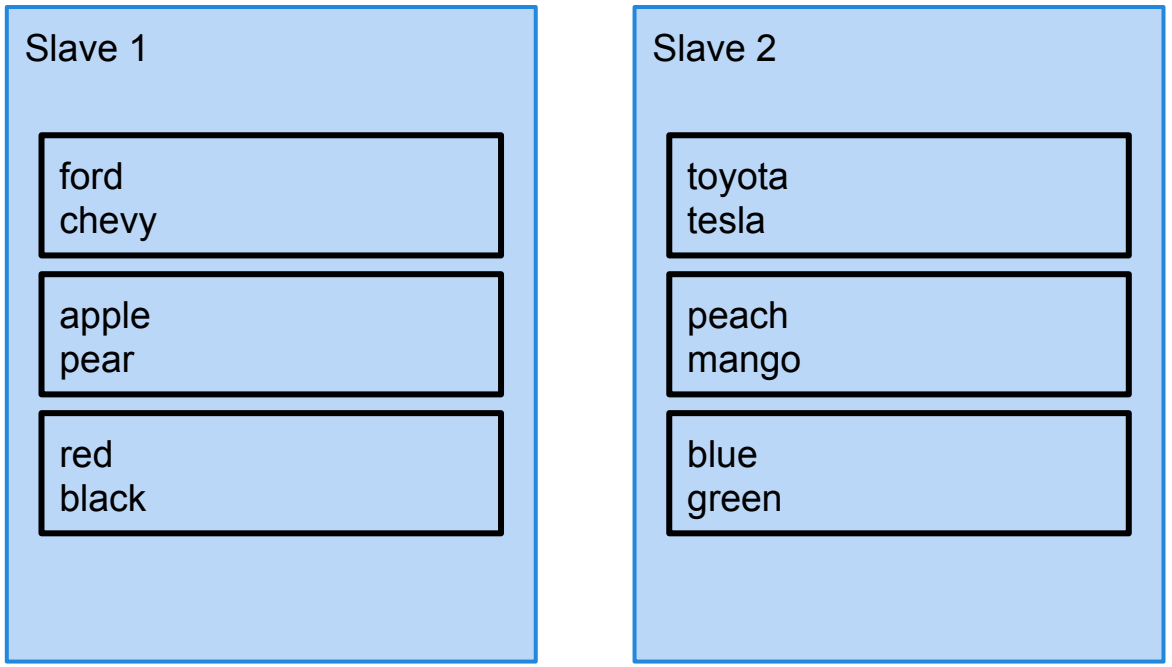
What is MapReduce?

- A workflow for processing distributed data.
- Four phases: Map, Combine, Shuffle, Reduce
- (Note: Hadoop is not the only platform that implements the MapReduce workflow. See MPI, Spark.)

Example: Suppose we have three text files and that we wish to count the number of times each letter of the alphabet appears in these files:



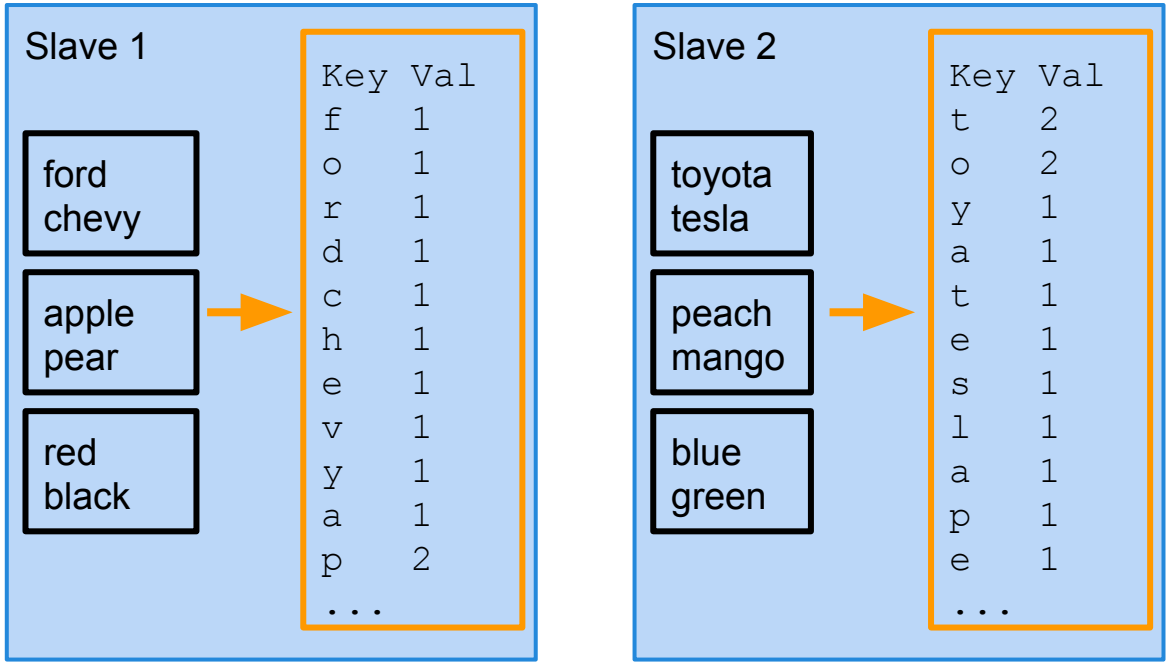
Suppose further that we have two slave nodes in our Hadoop cluster. Let's load our files into HDFS to distribute them across the cluster:*



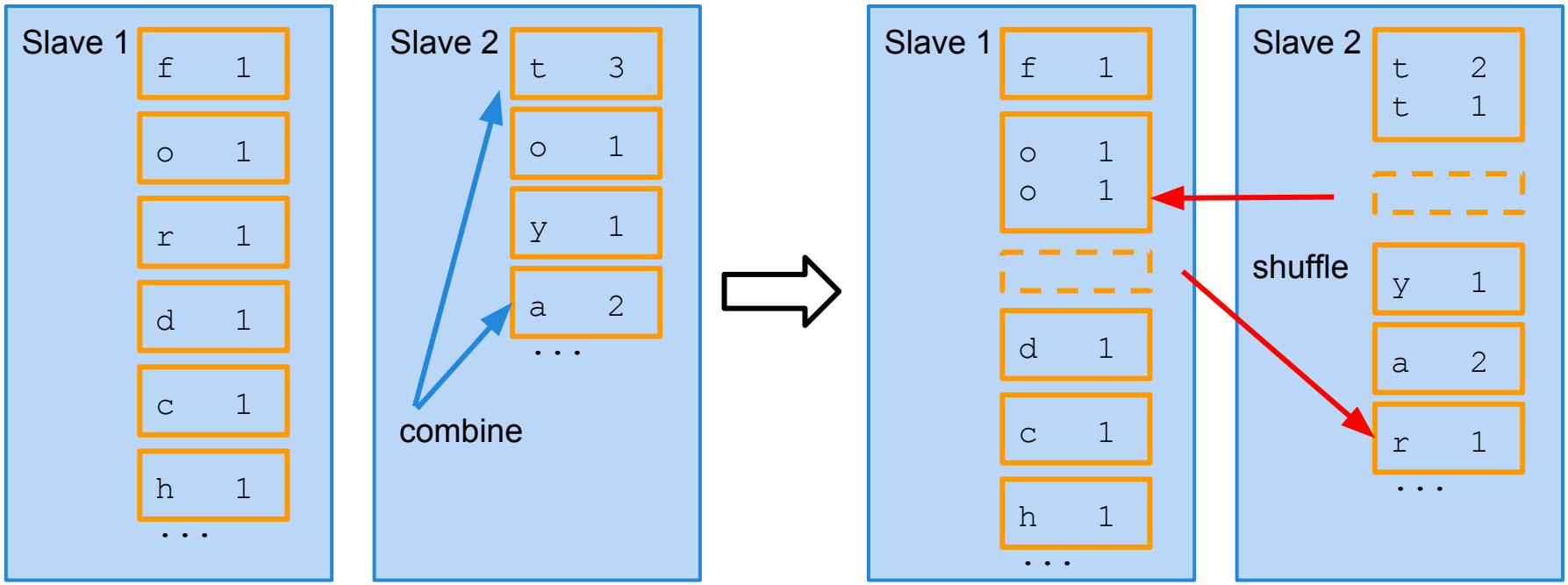
*duplicate blocks not shown

Now we begin the map phase:

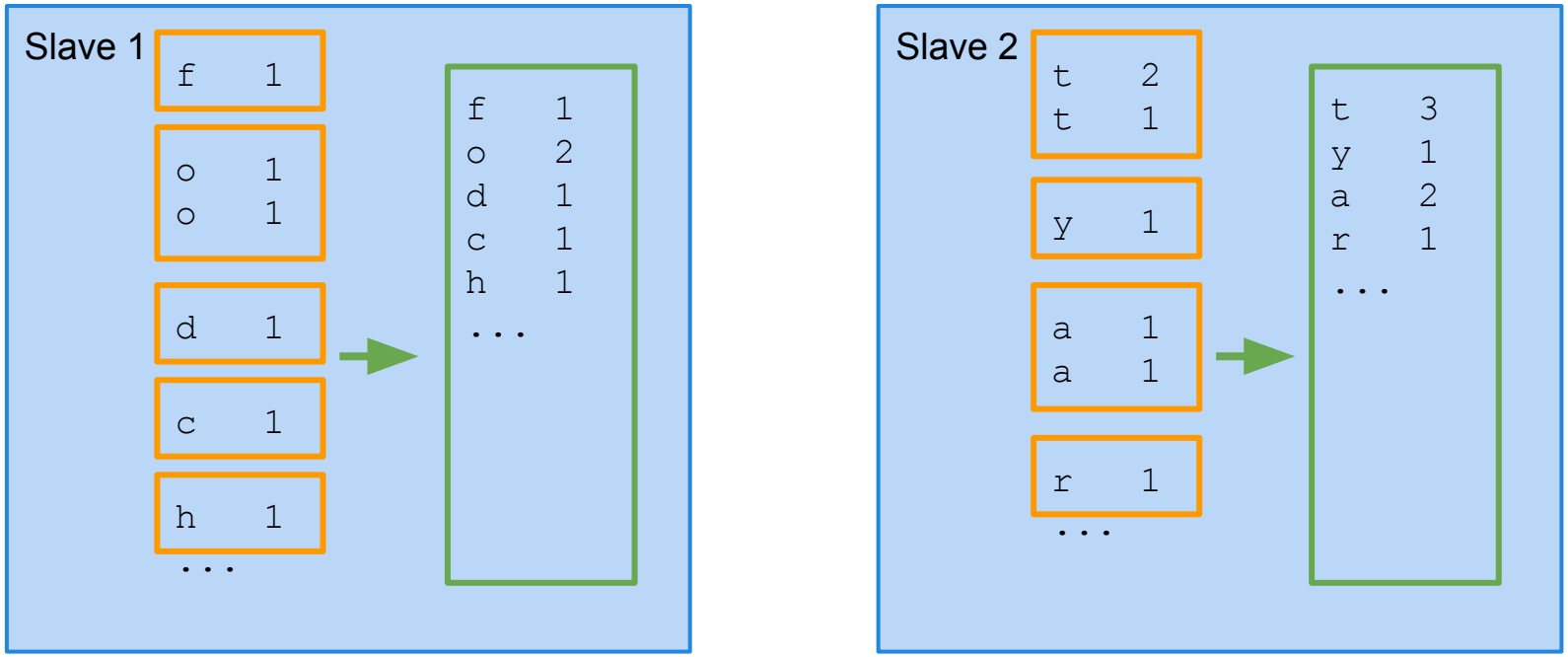
- We write a java map task and run the task across our cluster. We run multiple map tasks on each node.
- Each map task takes a local block as input (processing it one line at a time) and generates key-value pairs as output. For example, letter counts:



Next, K-V pairs with matching keys are grouped locally (combine phase). These groups are then redistributed across the cluster (shuffle phase) such that any K-V pairs with the same key will appear on the same node. The destination node for each key is determined by a hash.

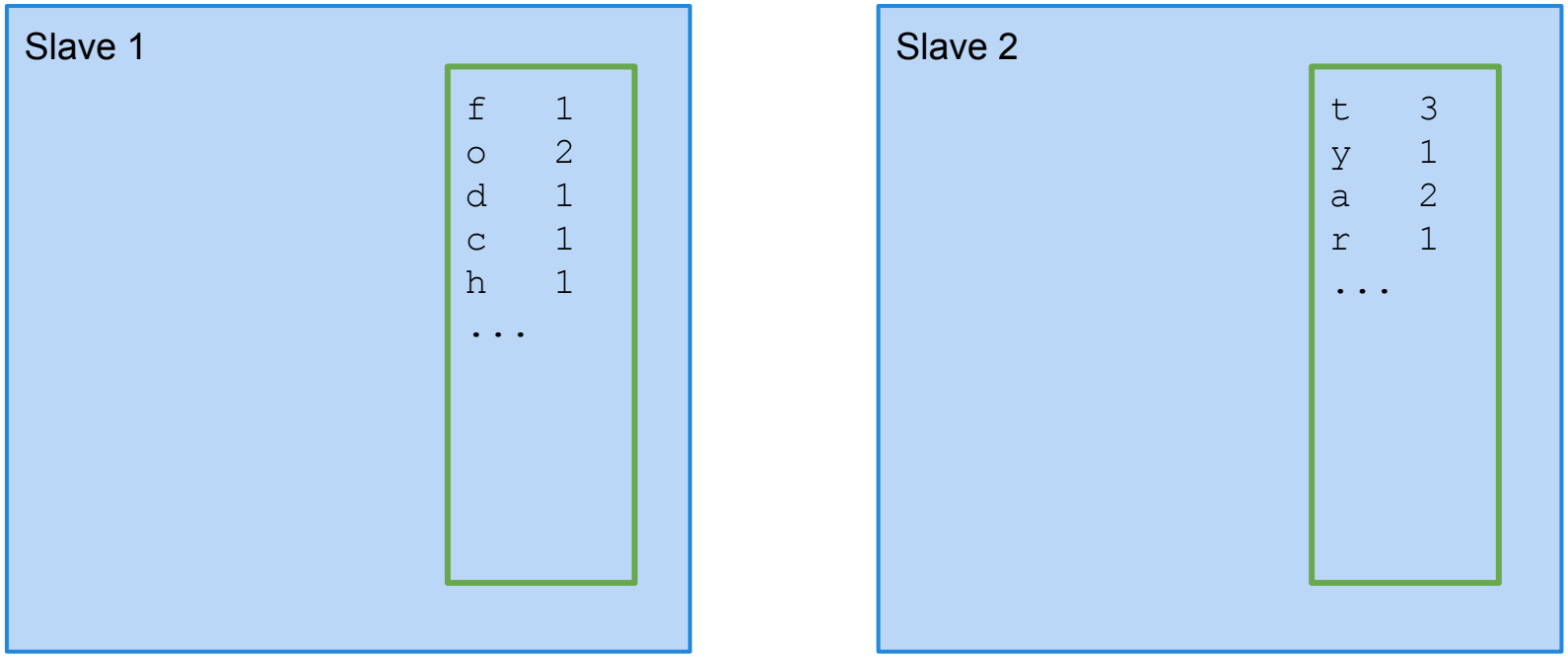


Finally, the reduce phase: We write a java reduce task and run multiple instances of this task on each node in our cluster. Each reduce task will take as input a number of K-V pair groups, iterate over the K-V pairs in those groups, and compute output K-V pairs (for example, total letter counts):*



*The terminology is subtle. Do not confuse a reduce task (processes K-V pairs for multiple keys) with a single call to the reduce() function in the job .java file (processes K-V pairs for only a single key). A single reduce task makes multiple calls to reduce().

The end result is that we have total letter counts for all 3 of our input files. Note that our two output files are distributed across our cluster, stored in HDFS. We can merge them into a single file on our local file system, if we wish.



Further Resources:

- The Hadoop materials on the Yahoo Developer's Network are quite informative: https://developer.yahoo.com/hadoop/tutorial/
- The Apache Hadoop website also has materials: https://hadoop.apache.org/
- For a good overview of YARN: http://www.ibm.com/developerworks/library/bd-hadoopyarn/