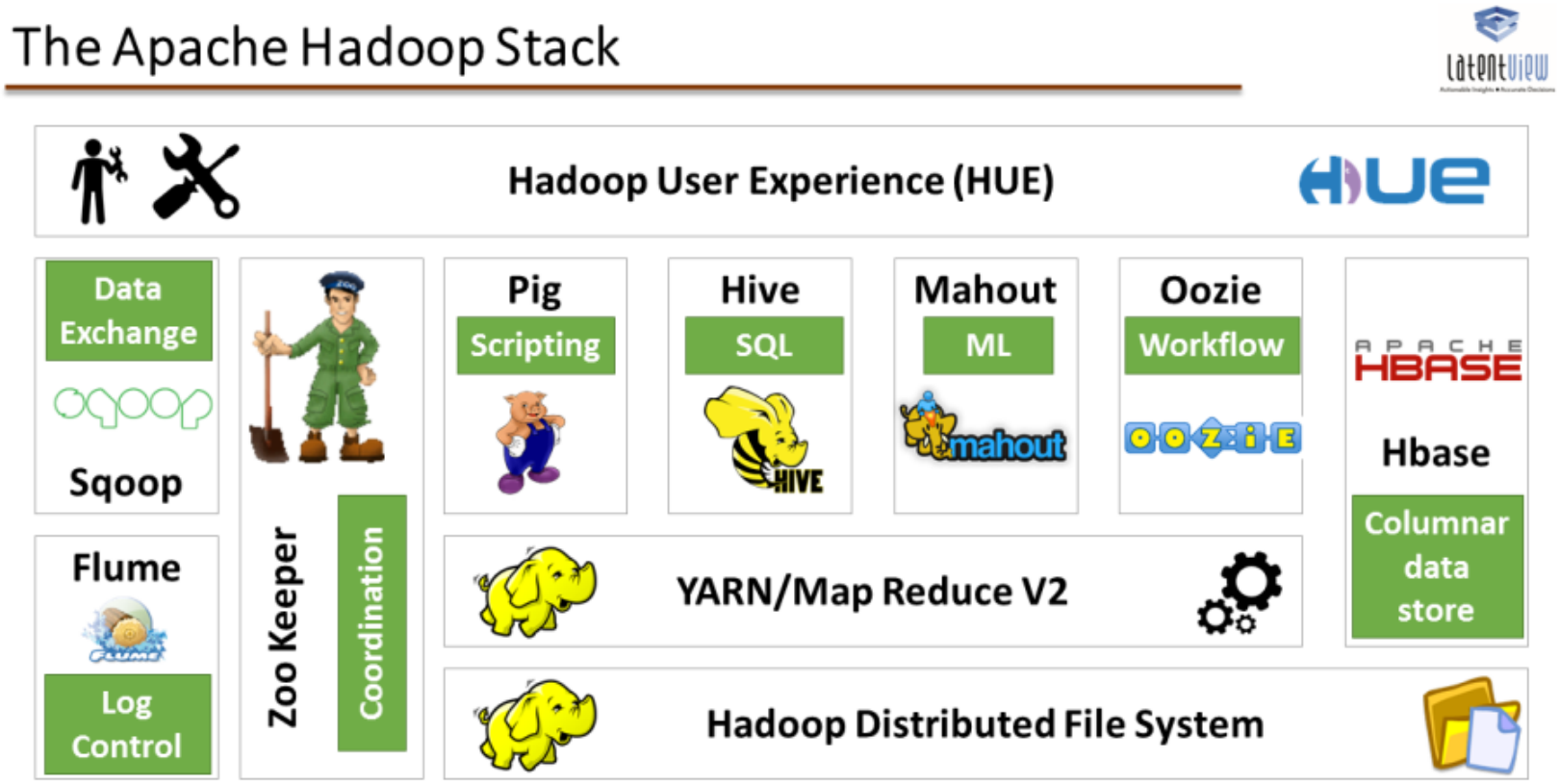
Hadoop Basics

A brief history on Hadoop

- 2003 - Google launches project Nutch to handle billions of searches and indexing millions of web pages.
- Oct 2003 - Google releases papers with GFS (Google File System)
- Dec 2004 - Google releases papers with MapReduce
- 2005 - Nutch used GFS and MapReduce to perform operations
- 2006 - Yahoo! created Hadoop based on GFS and MapReduce (with Doug Cutting and team)
- 2007 - Yahoo started using Hadoop on a 1000 node cluster
- Jan 2008 - Apache took over Hadoop
- Jul 2008 - Tested a 4000 node cluster with Hadoop successfully
- 2009 - Hadoop successfully sorted a petabyte of data in less than 17 hours to handle billions of searches and indexing millions of web pages.
- Dec 2011 - Hadoop releases version 1.0
- Aug 2013 - Version 2.0.6 is available

Hadoop Ecosystem



- The two major components of Hadoop
  - Hadoop Distributed File System (HDFS)
  - MapReduce Framework

HDFS

- HDFS is a filesystem designed for storing very large files running on clusters of commodity hardware.

  - Very large file: some hadoop clusters stores petabytes of data.

  - Commodity hardware: Hadoop doesn't require expensive, highly reliable harware to run on. It is designed to run on clusters of commodity hardware.

- Blocks

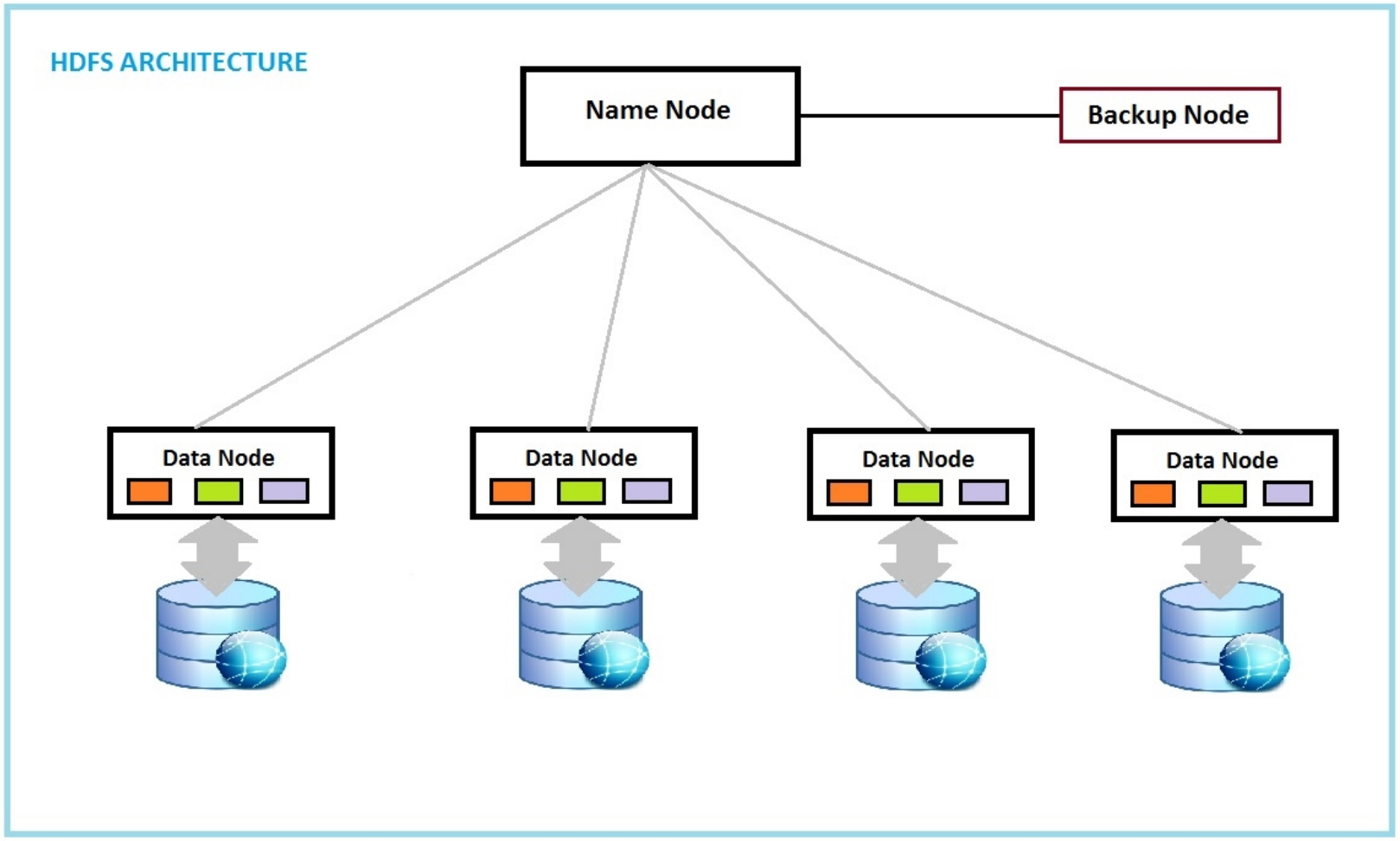
  - Files in HDFS are broken into block-sized chunks. Each chunk is stored in an independent unit.

  - By default, the size of each block is 64 MB.

- Some benefits of splitting files into blocks.

    -- a file can be larger than any single disk in the network.

    -- Blocks fit well with replication for providing fault tolerance and availability. To insure against corrupted blocks and disk/machine failure, each block is replicated to a small number of physically separate machines.


HDFS ARCHITECTURE
Name Node
Backup Node
Data Node
Data Node
Data Node
Data Node

- Namenodes

  -- The namenode manages the filesystem namespace.

  -- It maintains the filesystem tree and the metadata for all the files and directories.

  -- It also contains the information on the locations of blocks for a given file.

- Datanodes

  - datanodes: stores blocks of files. They report back to the namenodes periodically

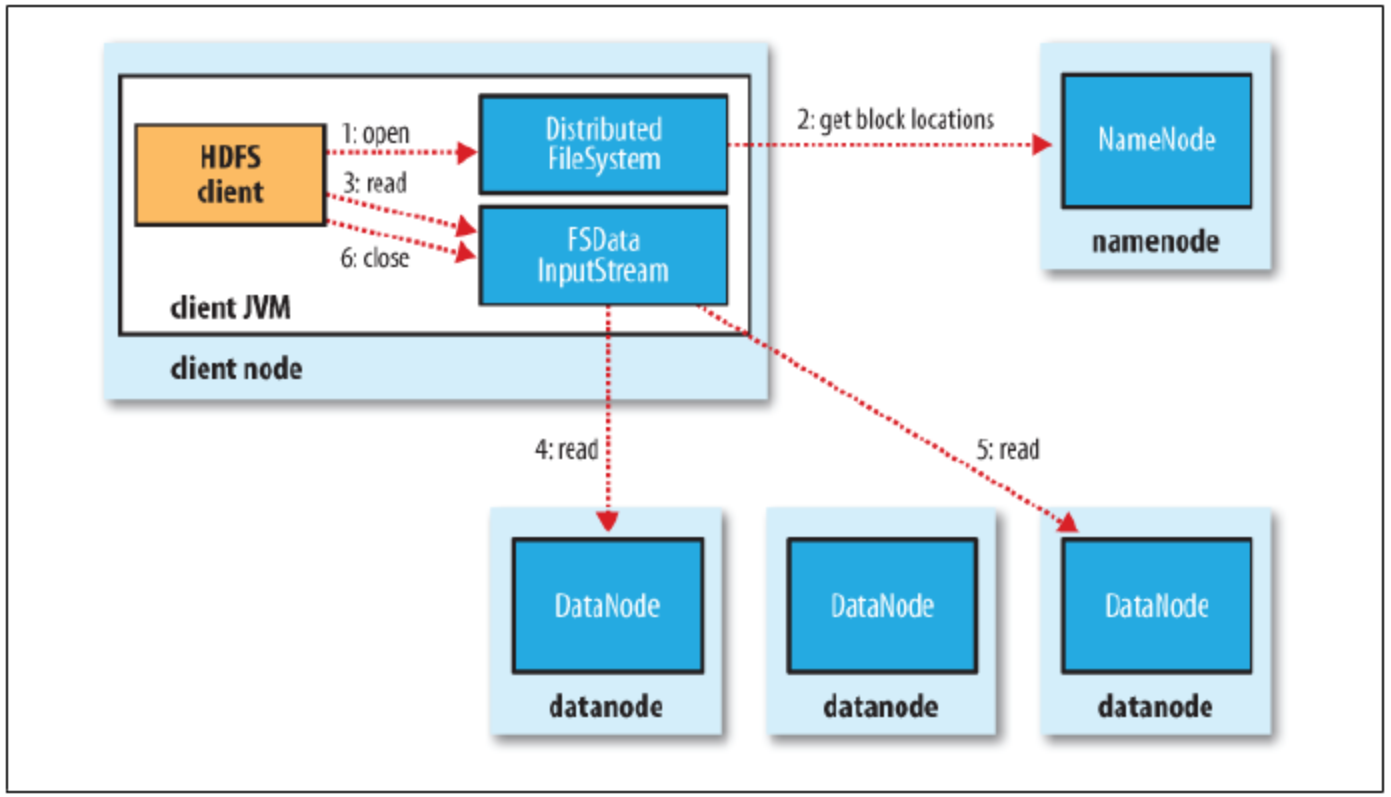


Figure 3-1. A client reading data from HDFS

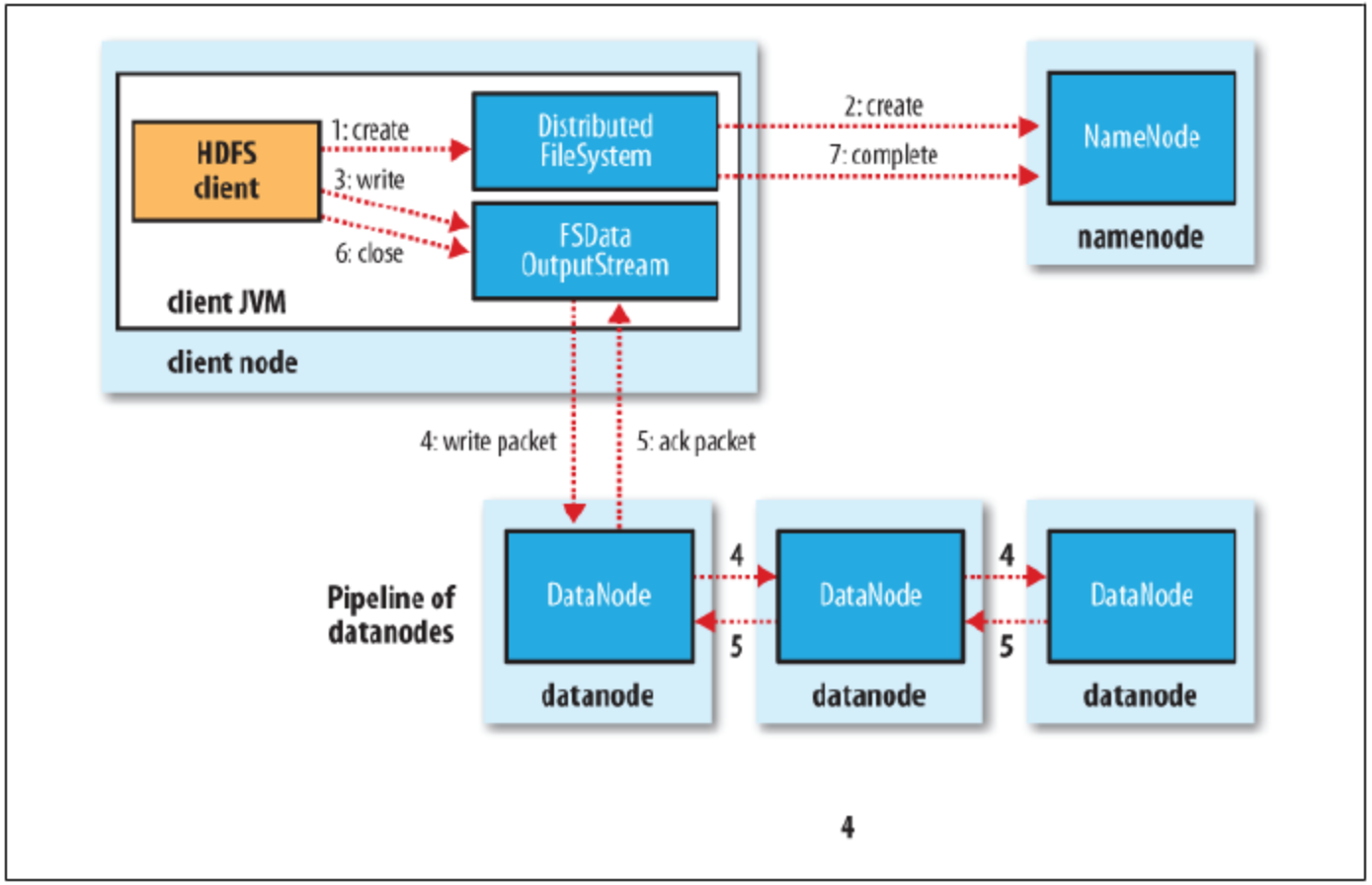


Figure 3-3. A client writing data to HDFS

MapReduce Programming Model – Mappers and Reducers

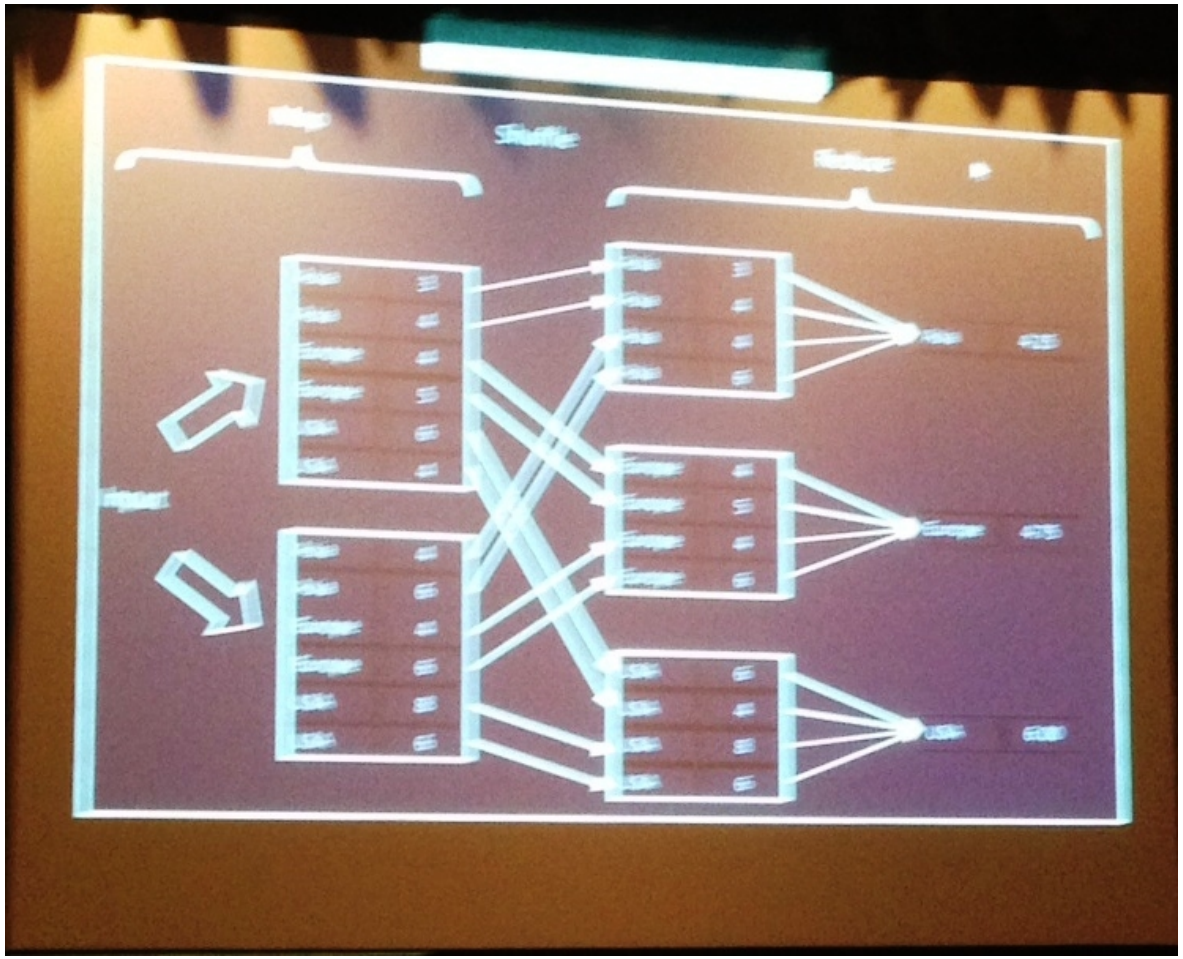
- In MapReduce, the programmer defines a mapper and a reducer with the following signatures:

$$\text{map: } (k_1, v_1) \rightarrow [(k_2, v_2)]$$
$$\text{reduce: } (k_2, [v_2]) \rightarrow [(k_3, v_3)]$$

- Implicit between the map and reduce phases is shuffle, sort, and group-by operation on intermediate keys.

- Output key-value pairs from each reducer are written persistently back onto the distributed file system.

MapReduce Schematic



Word Count- Schematic

In	Mappers		Shuffle	Reducers		Out	
	Key	freq		Key	freq	key	freq
	Word1-Book1	n1		Word1-Book1	n1		
	Word1-Book2	n2		Word1-Book2	n2	Word1	n13
Book1	Word2-Book1	n3		Word1-Book3	n7		
Book2	Word2-Book2	n4		Word1-Book4	n8		
	Word3-Book1	n5					
	Word3-Book2	n6		Word2-Book1	n3		
				Word2-Book2	n4	Word2	n14
				Word2-Book3	n9		
	Word1-Book3	n7		Word2-Book4	n10		
	Word1-Book4	n8					
Book3	Word2-Book3	n9					
Book 4	Word2-Book4	n10		Word3-Book1	n5		
	Word3-Book3	n11		Word3-Book2	n6	Word3	n15
	Word3-Book4	n12		Word3-Book3	n11		
				Word3-Book4	n12		

- **Computation:**
  n13 = (n1 + n1 + n7 + n8)
- n14 = (n3 + n4 + n9 + n10)
- n15 = (n5 + n6 + n11 + n12)

WordCount Example

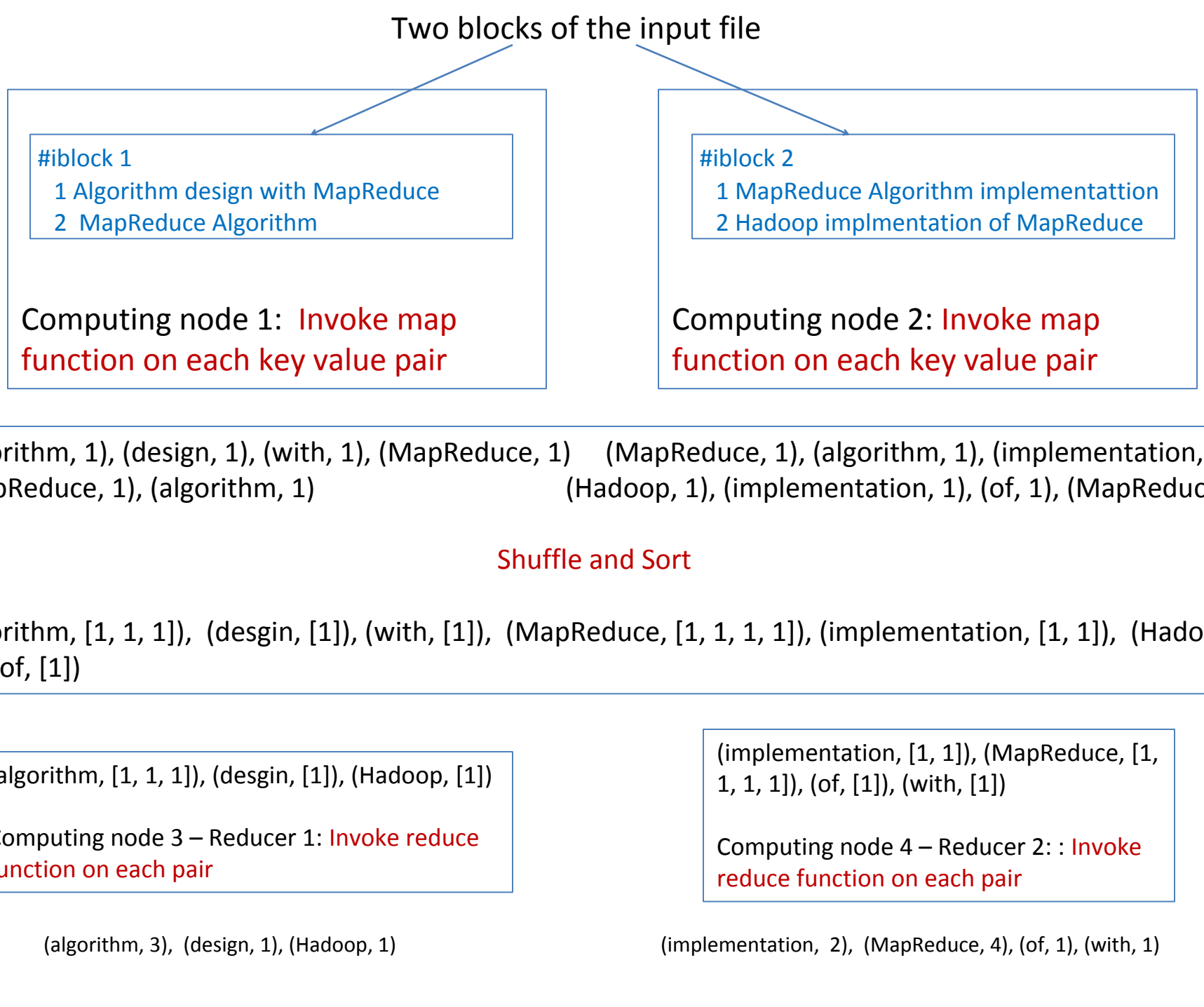
- Given the following file that contains four documents

```
#input file
1	Algorithm design with MapReduce
2	MapReduce Algorithm
3	MapReduce Algorithm Implementation
4	Hadoop Implementation of Hadoop
```

- We would like to count the frequency of each unique word in this file.

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term t ∈ doc d do
4:             Emit(term t, count 1)

1: class Reducer
2:     method Reduce(term t, counts [c1, c2, ...])
3:         sum ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             sum ← sum + c
6:         Emit(term t, count sum)
```


Two blocks of the input file
#iblock 1
1 Algorithm design with MapReduce
2 MapReduce Algorithm
Computing node 1: Invoke map function on each key value pair
#iblock 2
1 MapReduce Algorithm implementattion
2 Hadoop implmentation of MapReduce
Computing node 2: Invoke map function on each key value pair
(algorithm, 1), (design, 1), (with, 1), (MapReduce, 1)
(MapReduce, 1), (algorithm, 1)
(MapReduce, 1), (algorithm, 1), (implementation, 1)
(Hadoop, 1), (implementation, 1), (of, 1), (MapReduce, 1)
Shuffle and Sort
(algorithm, [1, 1, 1]), (desgin, [1]), (with, [1]), (MapReduce, [1, 1, 1, 1]), (implementation, [1, 1]), (Hadoop, [1], (of, [1])
(algorithm, [1, 1, 1]), (desgin, [1]), (Hadoop, [1])
Computing node 3 – Reducer 1: Invoke reduce function on each pair
(implementation, [1, 1]), (MapReduce, [1, 1, 1, 1]), (of, [1]), (with, [1])
Computing node 4 – Reducer 2: : Invoke reduce function on each pair
(algorithm, 3), (design, 1), (Hadoop, 1)
(implementation, 2), (MapReduce, 4), (of, 1), (with, 1)