# Hadoop

- an open source implementation of the MapReduce; parallel processing framework

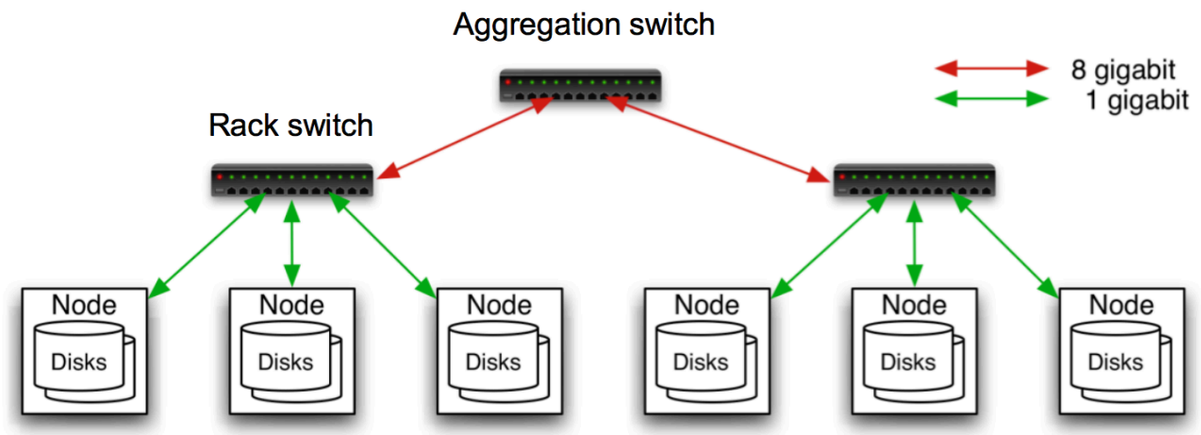
- includes
  - data distribution to processing nodes
  - handle failures; restarting nodes, or reassigning task to another node
  - execution of user programs
  - collecting final result

# Hadoop Components

- Hadoop Distributed File System (HDFS)
  - for data storage distribution
- MapReduce framework
  - for parallel data processing
  - execute user programs written as **map** & **reduce** functions

# Typical Hadoop Cluster

- Hundreds/ thousands of nodes (1000 - 4000)
- 40 nodes /rack
- Node specs (Yahoo)
  - 8 cores, 8 GB memory, 4 disks (=4 TB)



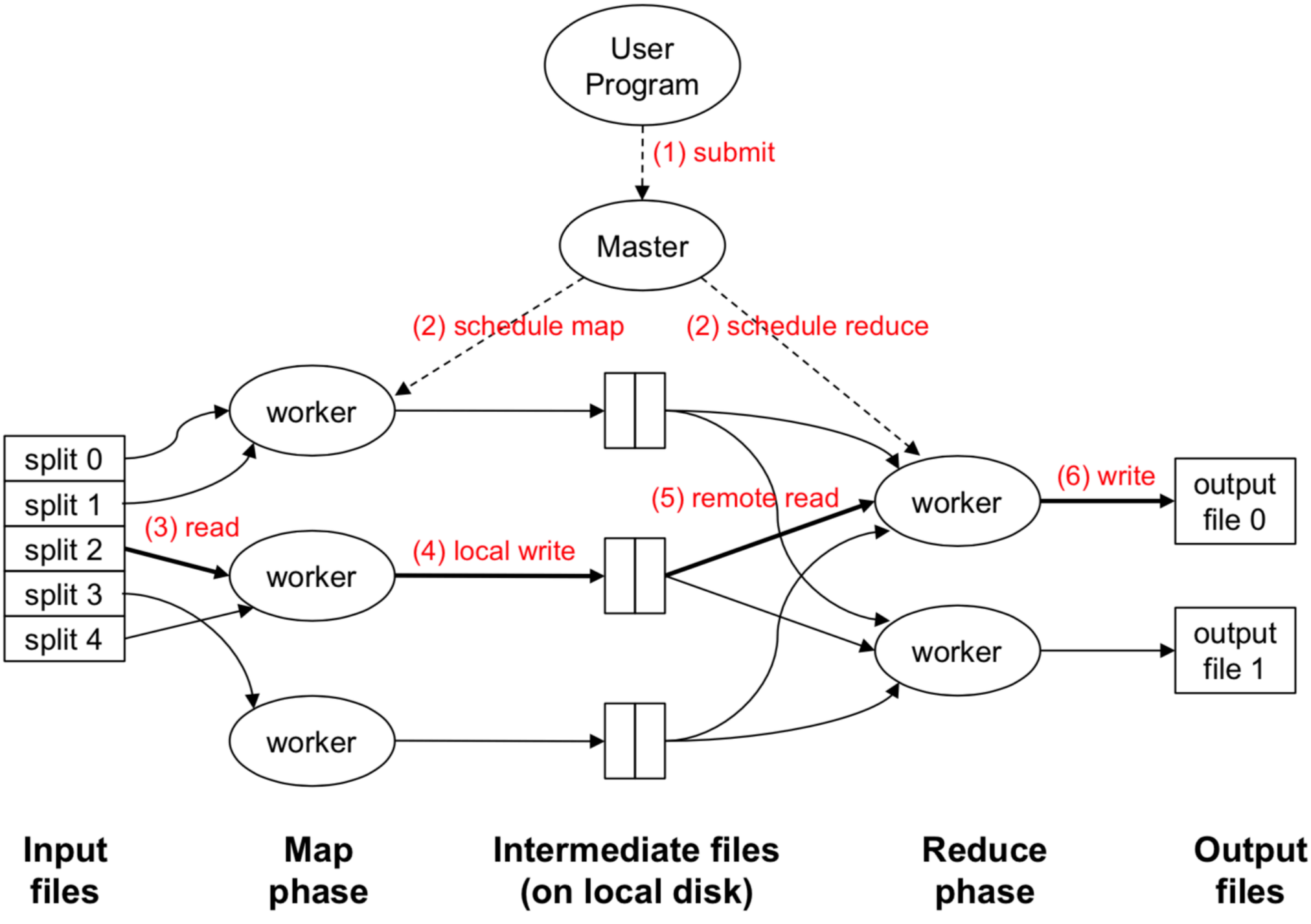
# MapReduce

- Automatic parallelization & distribution
- Provides status & monitoring tools
- clean abstraction for the programmers
  - only need to define the map & reduce functions

Framework

Programming Model

# Physical View



# MapReduce Workflow

- Master node is the JobTracker
  - the main functionality is to manage resources
    - keeps track of the running tasks
    - re-assign task to another node in case of failure
- All other nodes are TaskTracker

# Hadoop Distributed File System (HDFS)

- An open-source implementation of Google File System (GFS)

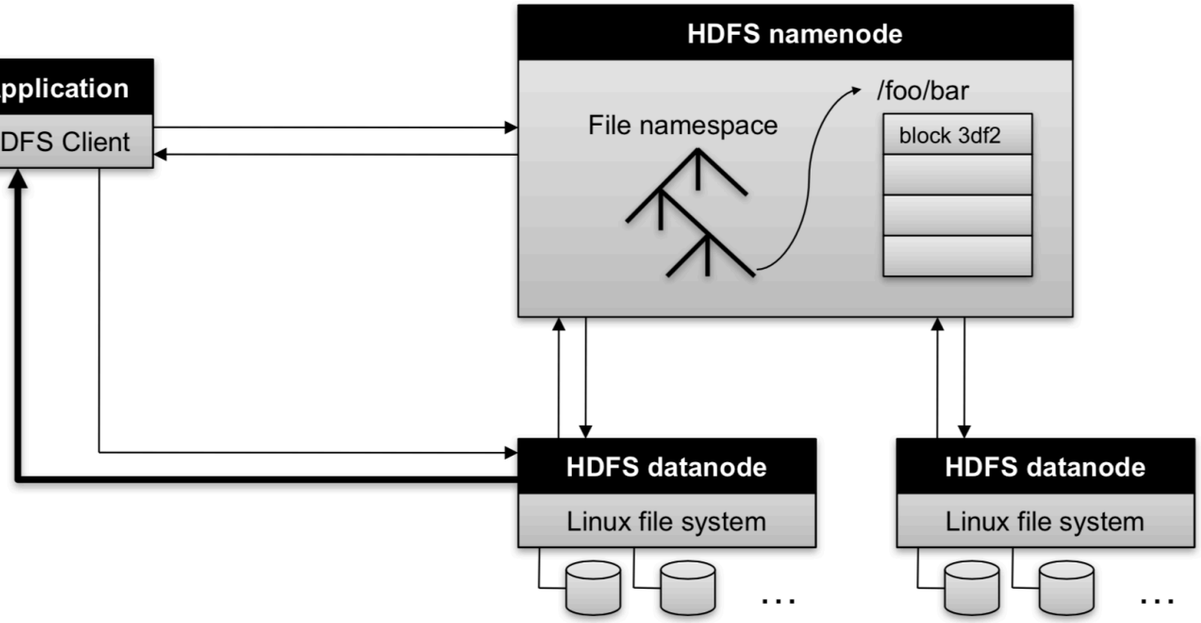
- A filesystem designed to store large files, running on cluster of commodity machines

- Files in HDFS are split into block called chunks, these chunks are stored independently

  - example, chunk size is 64 MB

# HDFS Architecture

- The NameNode (master node)
  - holds file/directory structure
  - mapping chunks to files
    - metadata; ownership, permissions
    - directs clients to where to the DataNode that contains the data
    - takes care of distributing data on the DataNodes
- DataNodes (workers) store the data



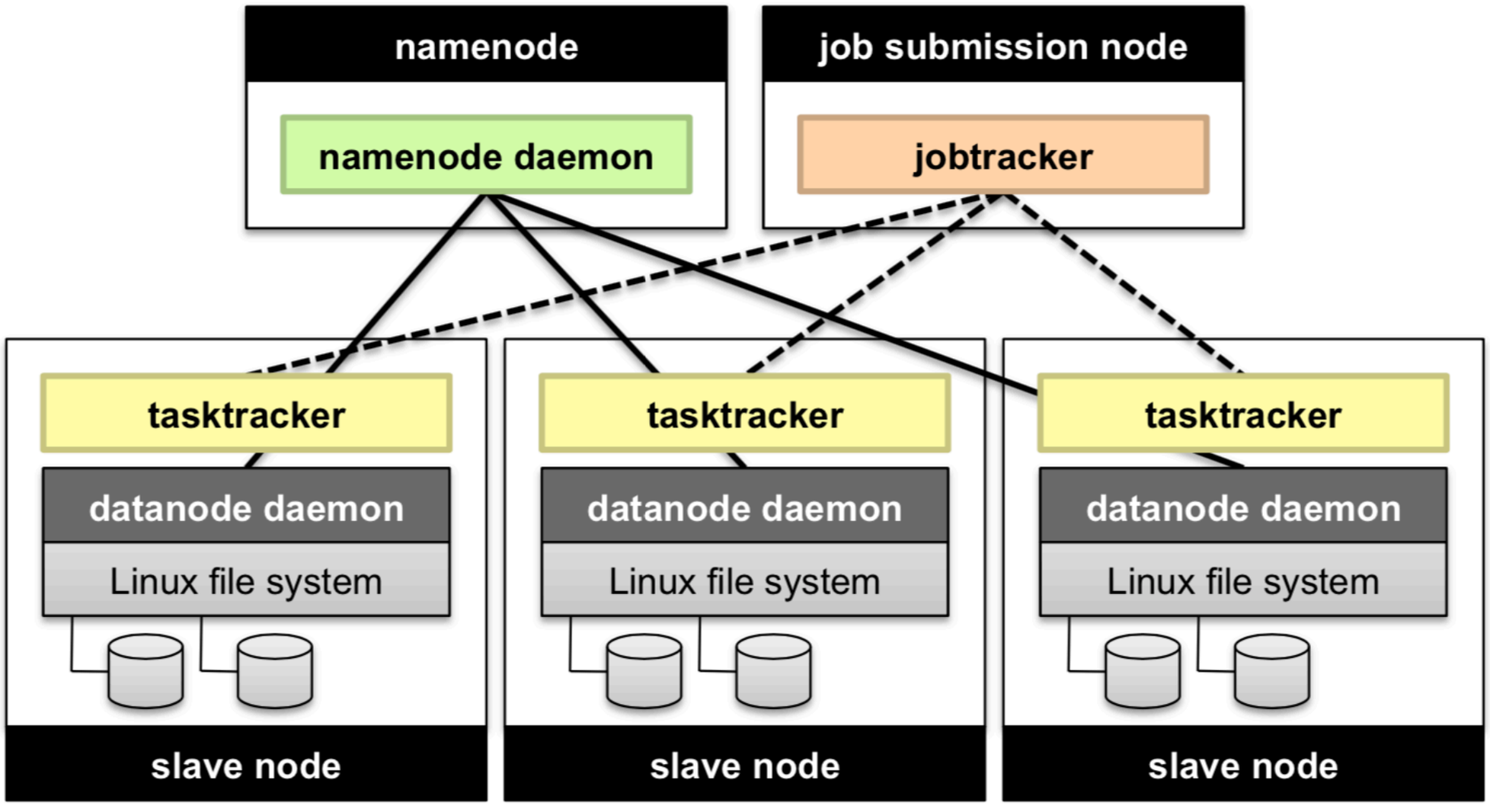
# HDFS Design

- Files are stored as chunks
  - fixed size
- Reliability
  - no data loss because chunks are replicated (default is 3 replicas)
- Single master node to manage access, keep metadata about the data

# HDFS commands

- Unix shell-like commands (ls, rm, mkdir)
- Every command starts with $ bin/hadoop dfs
- Examples:
  - *bin/hadoop dfs -ls /user/hduser* (list the content of /user/hduser)
  - *bin/hadoop dfs -copyFromLocal* ~/test.txt /user/hduser/input (copy file from local computer to the HDFS )

# Putting Everything Together



# MapReduce Job

- Refers to the unit of work that the client wants to be performed
- It consists of:
  - MapReduce program,
  - and job configuration; such as type of the input, where to find data, where to store data, types of the key-value pairs

# Hadoop Running Mode

- Standalone (local) mode
    - this mode is good running MapReduce during development; easy to test & debug
        - Hadoop daemons; JobTracker, TaskTracker, NameNode, DataNode
- Pseudo-distributed mode
    - Hadoop daemons run on single machine, thus simulate the cluster at small scale utilizing cores available on the machine
- Full-distributed mode
    - runs on cluster of machines

http://hadoop.apache.org/

# MapReduce Complete Data Flow

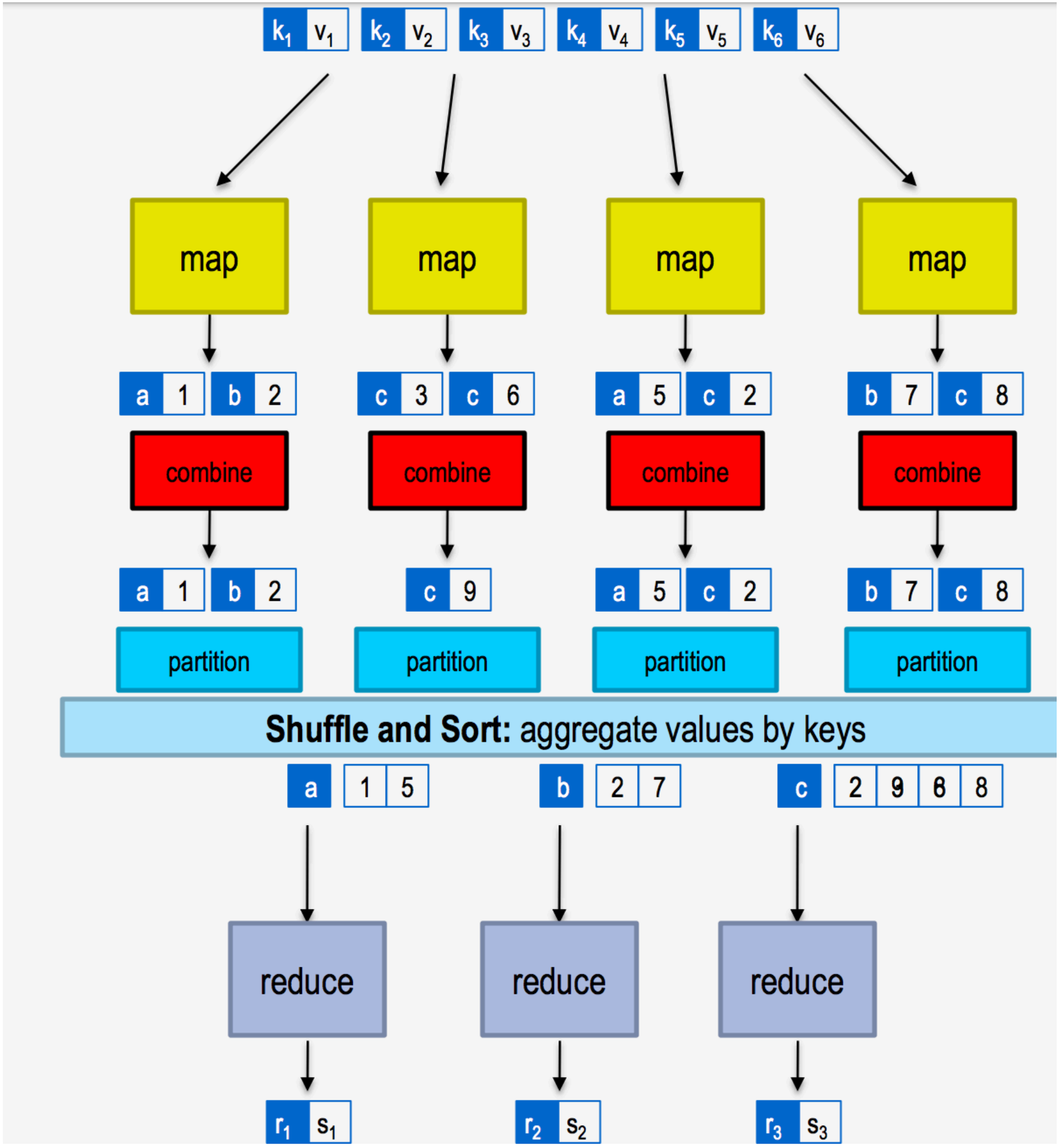
- Input
- Map
- Combine
- Intermediate output
- Partition
- Shuffle & Sort (aggregate key-value pairs by key)
- Reduce
- Output

# Input Format

- Defines how to read data from the input file into the map
- Based on the input data, you choose the input format
- Hadoop comes with several implementation for input format
  - For example, TextInputFormat
    - key: line offset
    - value: line content

# Combiner

- It is optional
- It is a local mini-reducer
- The goal is to minimize data transfer between map and reduce tasks
- Combiners compact the map’s output,
  - so there is less data to be written on local disk, and to be transferred to reduce
- The output from the map can be reduced by using compression
  - this can be configured in the code; Boolean to allow compression, and which compression method to be used


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
combine
combine
combine
combine
a 1
b 2
c 9
a 5
c 2
b 7
c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 6 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3