

# Cloud Computing with Map-Reduce and Hadoop

Mentors: Wenjun Zeng, Qia Wang



key-value pairs

(word, final-count)

~ SQL aggregation

#### Objectives

- Cloud Computing
- Parallel processing
- Map-Reduce algorithm
- Hadoop environment

#### Map-Reduce

- \*Simple data-parallel programming model designed for scalability and fault-tolerance
- Pioneered by Google

Intermediate

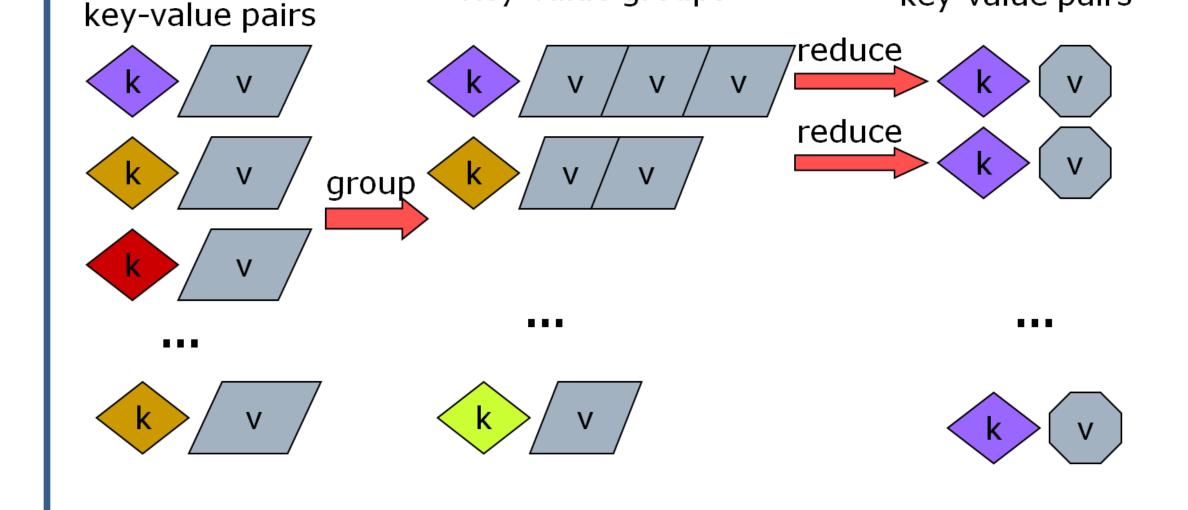
Popularized by Hadoop project

#### The Map and Reduce Step

 $map(k,v) \rightarrow list(k1,v1)$ reduce(k1, list(v1))  $\rightarrow$  v2

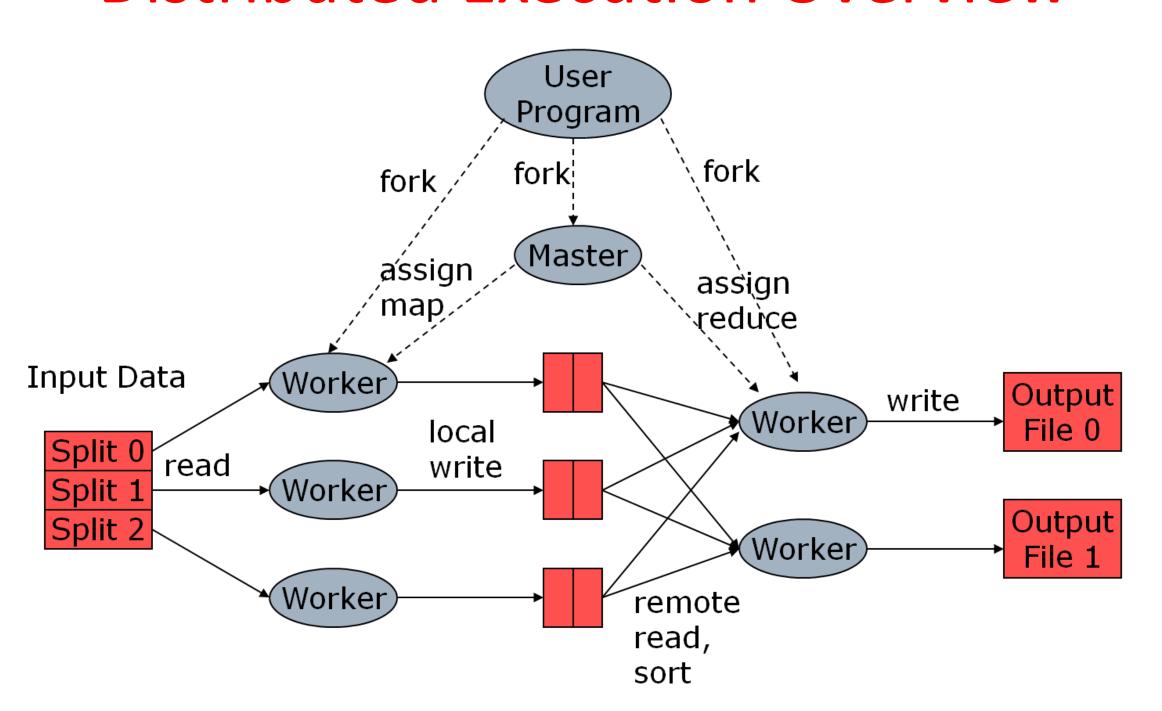
Output

key-value pairs



Key-value groups

#### Distributed Execution Overview



#### The Hadoop Environment

- Distributed, highly fault-tolerant file system
- Handling large data sets

Author: Dwight D. Anderson

- Communication protocols are build on the TCP/IP model.
- \* HDFS: Hadoop Distributed File System

## HDFS Architecture 2.Blckid, DataNodes NameNode Client 3.Read data DataNodes Metadata (Name, replicas, ...): /home/foo/data, 3, ... Namenode Metadata ops Block ops Datanodes Read Datanodes Replication Blocks Write Rack 2 Rack 1

NameNode: Maps a file to a file-id and list of

MapNodes

DataNode: Maps a block-id to a physical location on

disk

### Implementation: Word Count The Map Step Input Intermediate key-value pairs key-value pairs E.g. (doc—id, doc-content) E.g. (word, wordcount-in-a-doc) The Reduce Step Output Intermediate Key-value groups

#### Verification

key-value pairs

(word, wordcount-in-a-doc)

❖ All the tasks are simulated within a single thread

(word, list-of-wordcount)

~ SQL Group by

Results are verified correctly

#### Future Work

- Run the task with multi-threads or multi-servers to compare performance
- Apply Map-Reduce to more complicated data-mining algorithms