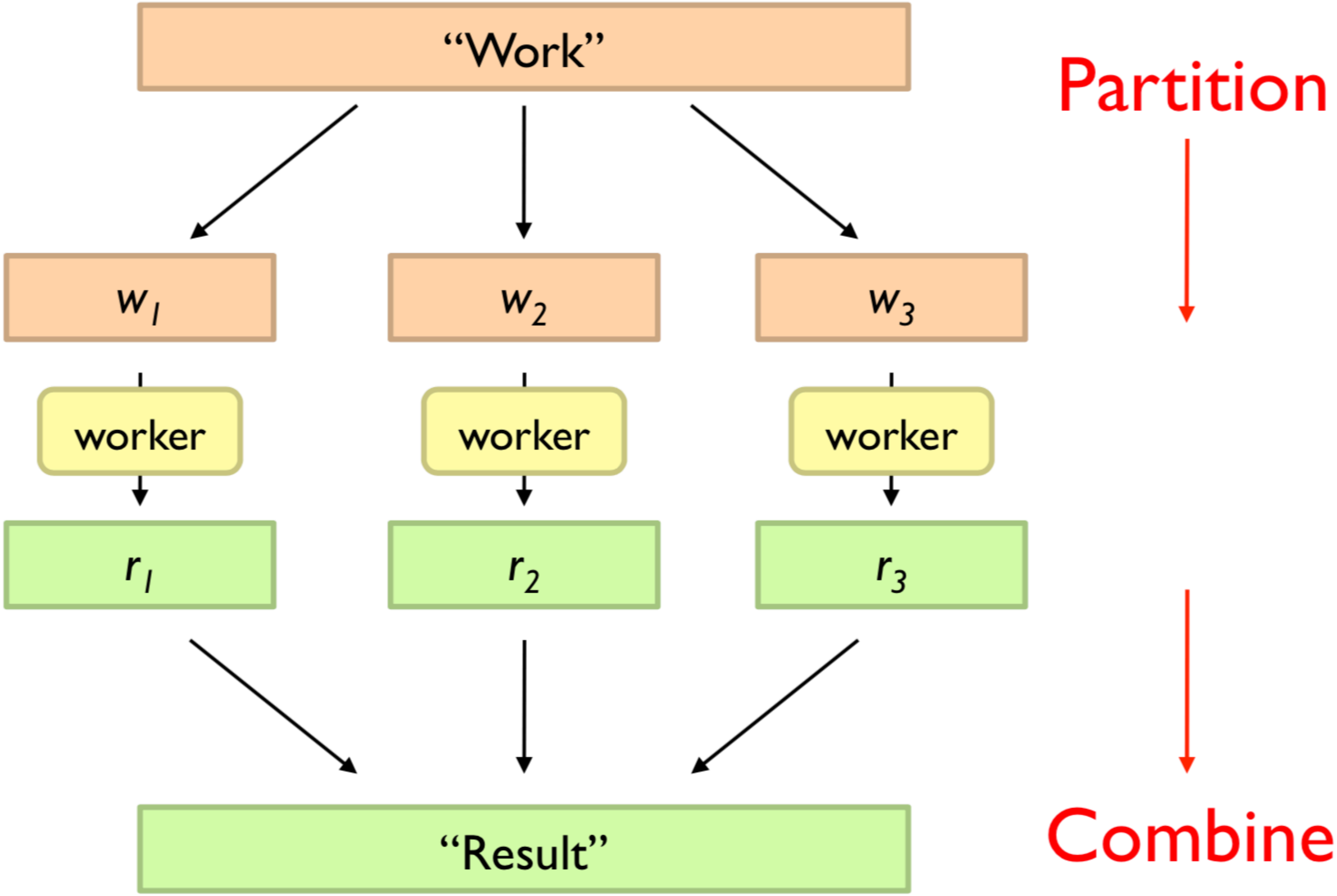
Introduction to MapReduce

Large Scale Data Processing

- We want to process large amount of data (Terabytes/Petabytes)
- We want to parallelize across hundred/thousands of CPUs
- Do that in an easy way

Divide & Conquer



Parallelization Challenge

- How do we partition data into units?
- How do we assign data units to workers?
- What to do if number of units > number of workers?
- What if workers need to share partial results?
- How do we know when workers finish their jobs?
- How do we aggregate results from all workers?
- What if a worker die? what happened to the data it was processing? how do we continue what it already processed?

Ideas behind MapReduce

- Scale “out” not “up”
  - using large number of commodity computers (scale out) is preferred over small number of high-end server
- Hardware abstraction
    - From the user point of view dealing with Data center is as one computer
- Hide System-level details
    - like data partitioning, communication between workers, coordination, handling error & failure
        - The framework takes care of all the challenges listed in the previous slide
- Moving process to data
  - run program on the node that has the data

Typical Large Data Problem

- Iterate over large number of records (e.g., lines in text or rows is a DB)

map

- Extract something

- Shuffle & sort intermediate results

execution framework

- Aggregate intermediate results

reduce

- Generate Final output

MapReduce Implementation

- It was developed by Google
  - written in C++
  - published as a research paper in 2004 “MapReduce: Simplified Data Processing on Large Clusters ”
- Hadoop MapReduce is an open-source implementation in Java
  - initially was done by Yahoo
  - then became an Apache project

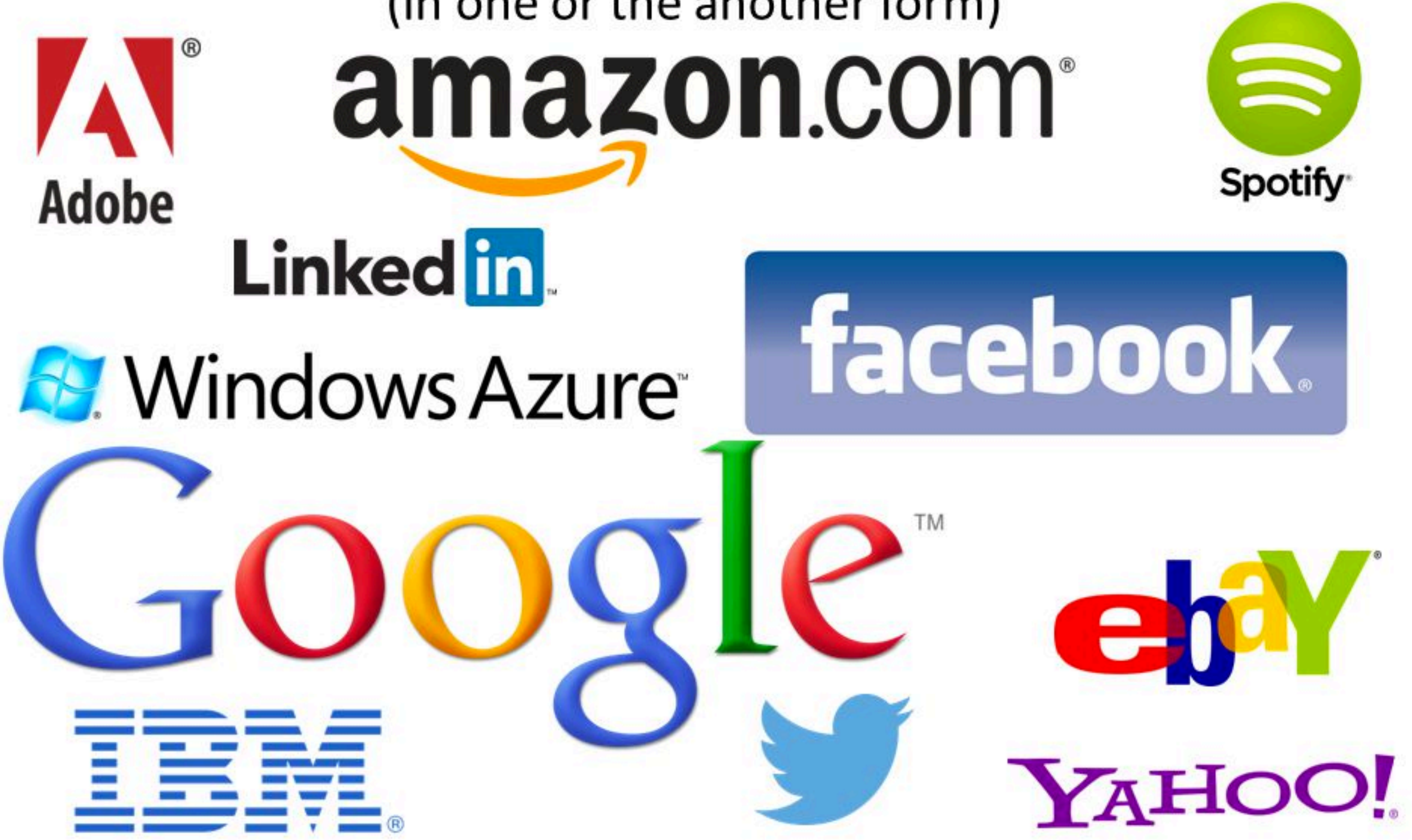
hadoop

- Apache Hadoop develops open-source tools for **reliable**, **scalable** and **distributed** processing of a large-scale data

- Two main components in Apache Hadoop project:
  - Hadoop distributed File System (HDFS) (Storage)
  - MapReduce (Processing)

Who uses Hadoop?

(in one or the another form)



MapReduce refers to

- Programming model
    - Two main functions: map and reduce
- Execution framework
- Specific implementation (the code/program)

Usage usually clear from the context

Programming Model

- Processing large datasets in parallel on cluster, by dividing work into set of independent tasks
- Programmer specifies two functions
  - map (k1 , v1) —> List [(k2 , v2)]
  - reduce (k2, List[v2]) —> List [(k3 , v3)]
- All values of same key are sent to the same reducer

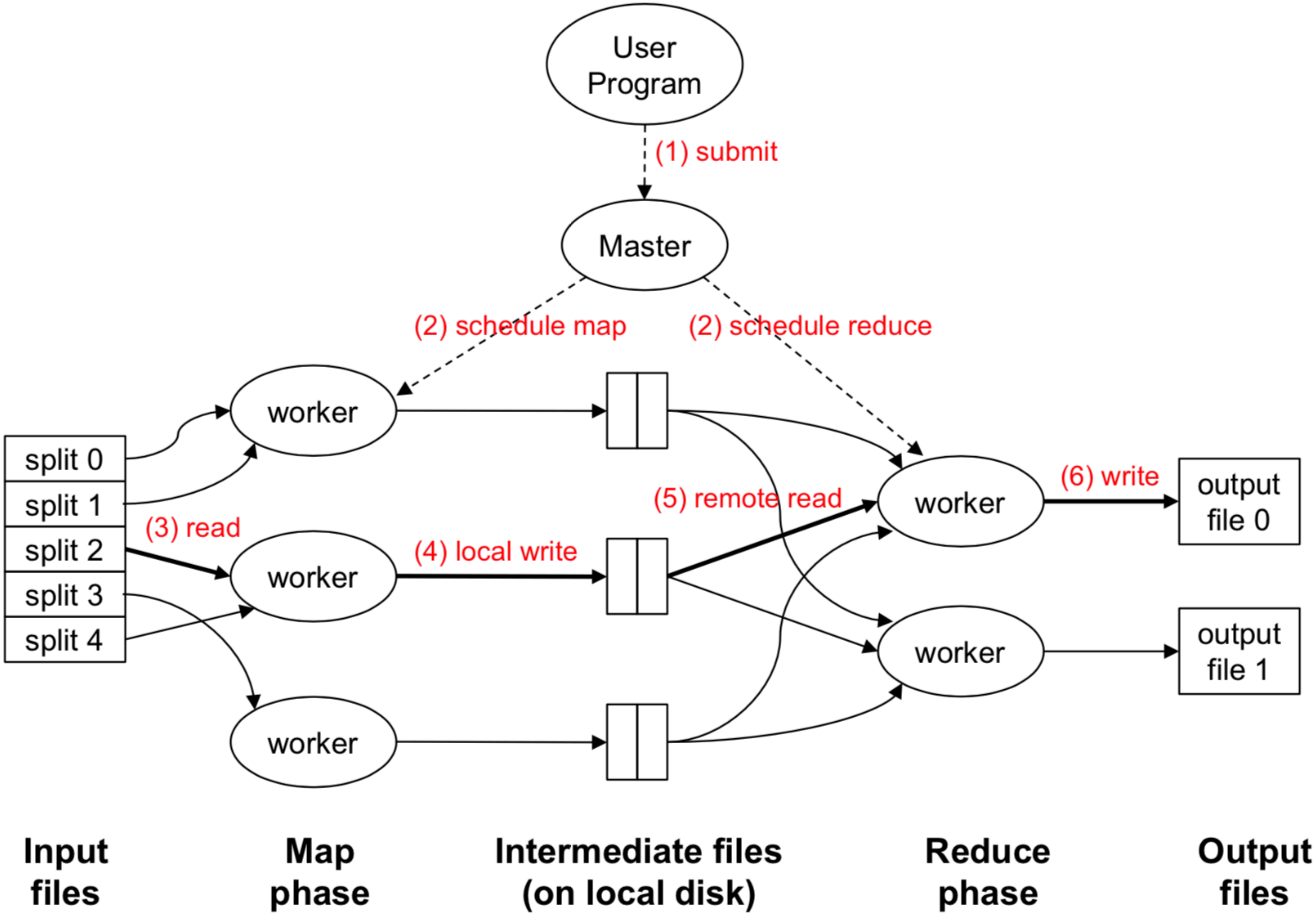
Key-Value pairs

- Input & output are key-values
- Examples:
  - Text Files
    - key: line offset
    - value: line content
  - in Web collection, which consists of Web pages
    - key: URL
    - Value: Content
  - Graph which consist of nodes and edges
    - key: node id
    - value: list of target nodes

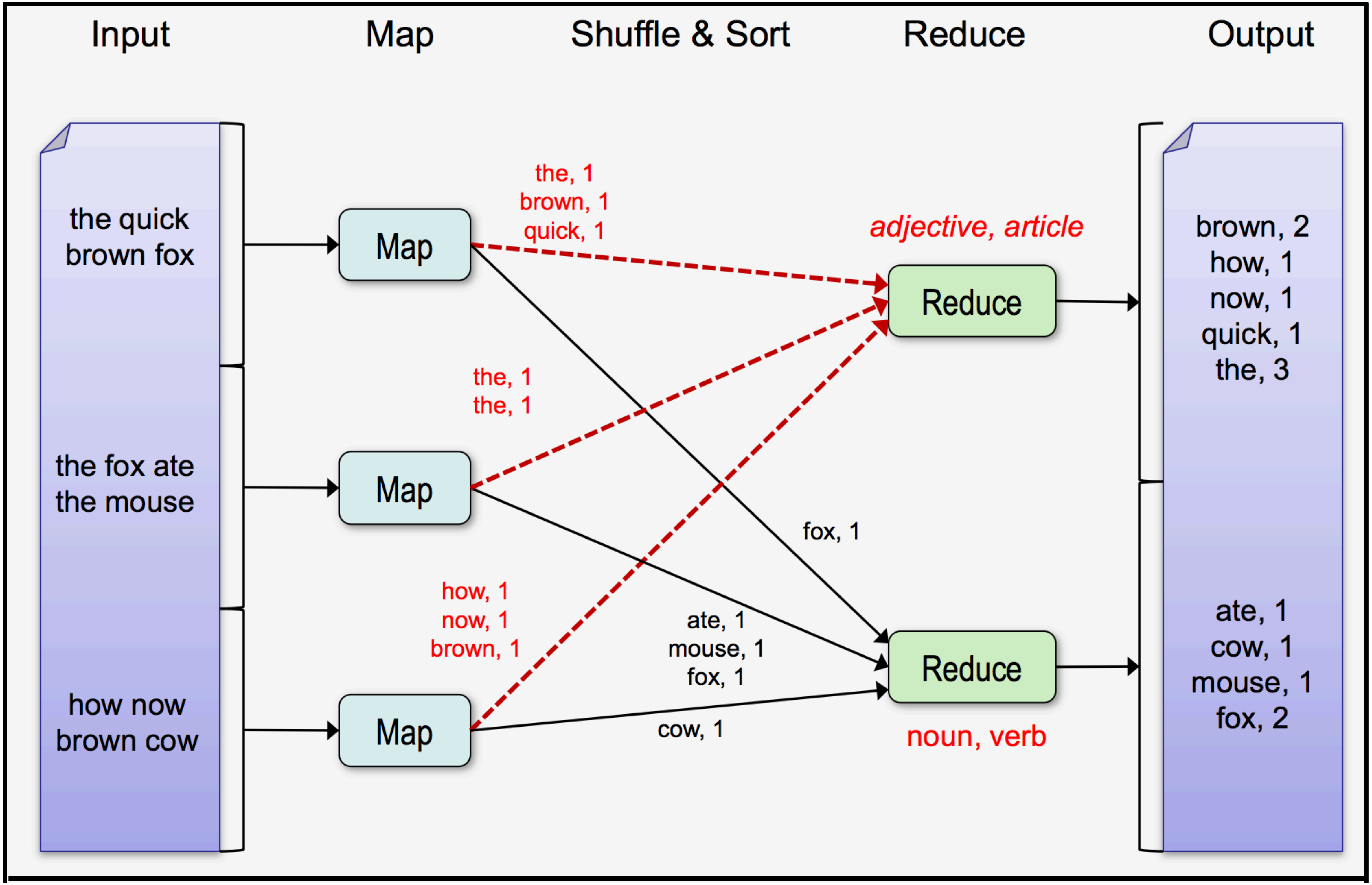
MapReduce Framework

- Handles:
  - scheduling
    - assigning workers to map & reduce tasks
  - data distribution
    - moves process to data
  - synchronization
    - group intermediate data
  - error & faults
    - detects workers failure, restart
  - Everything happens on top of Distributed File System

Physical View

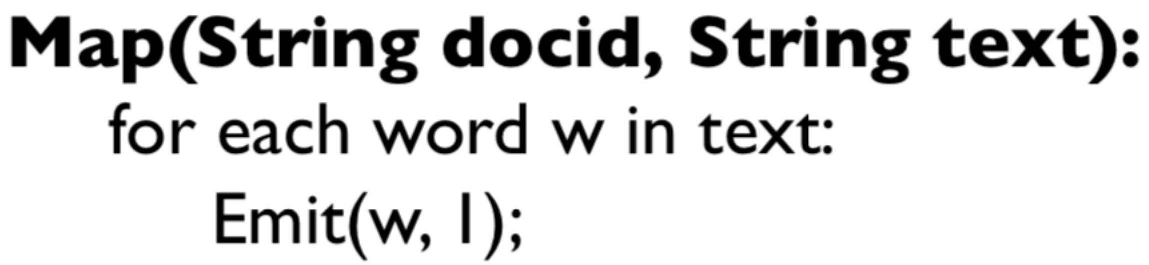


Word Count Example



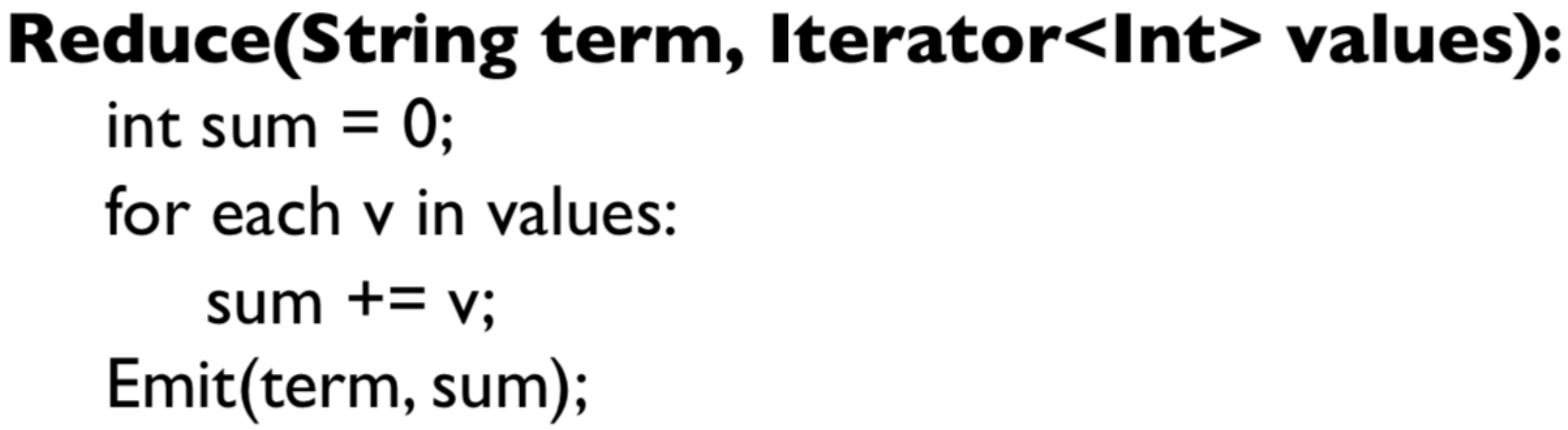
World Count Implementation

- Map



```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);
```

- Reduce



```
Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
    Emit(term, sum);
```