MapReduce

Scaling a simple program manually

WordCount Example

- "Do as I say, not as I do"

Word	Count
as	2
do	2
i	2
not	1
say	1
as	2

```
define wordCount as Multiset;
for each document in documentSet {
    T = tokenize(document);
    for each token in T {
        wordCount[token]++;
    }
}
display(wordCount);
```

Distributed Version

- When sets of documents become large
  - spam filter to know the words frequently used in the millions of spam emails you've received.

```
define wordCount as Multiset;
for each document in documentSubset {
    T = tokenize(document);
    for each token in T {
        wordCount[token]++;
    }
}
sendToSecondPhase(wordCount);
```

```
define totalWordCount as Multiset;
for each wordCount received from firstPhase {
    multisetAdd (totalWordCount, wordCount);
}
```

Issues

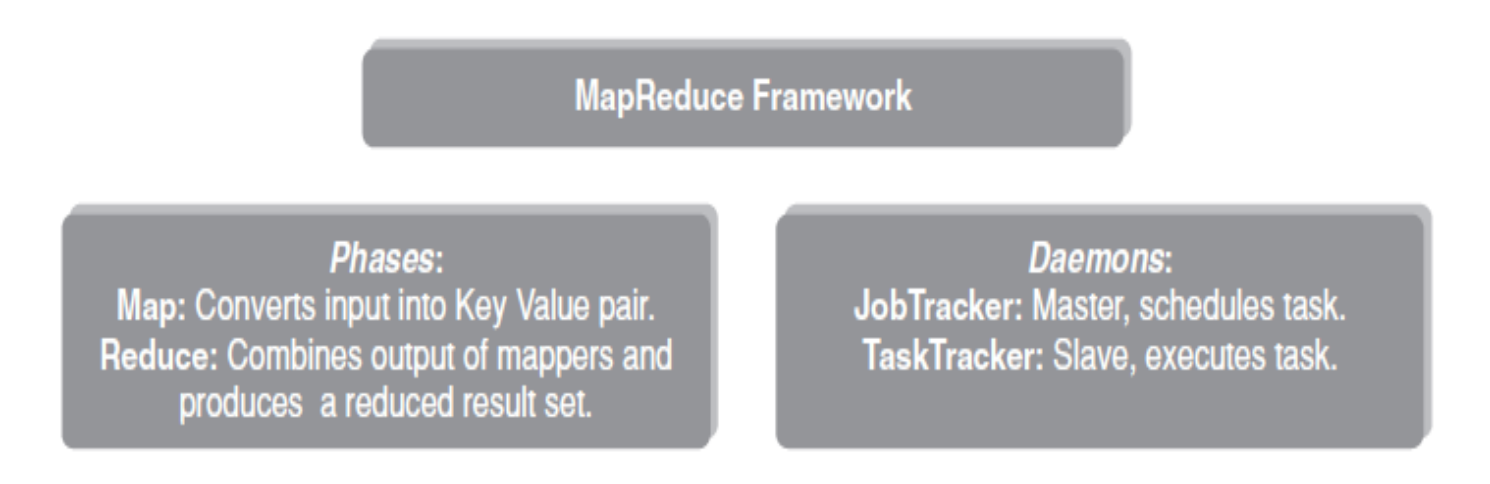
- Performance requirement of reading in the documents
- wordCount (and totalWordCount) are stored in memory. What if wordCount may not fit in memory.
- Phase two will become the bottleneck because it has only one machine
  - partition wordCount after phase one such that each machine in phase two only has to handle one partition.

Why MapReduce Framework

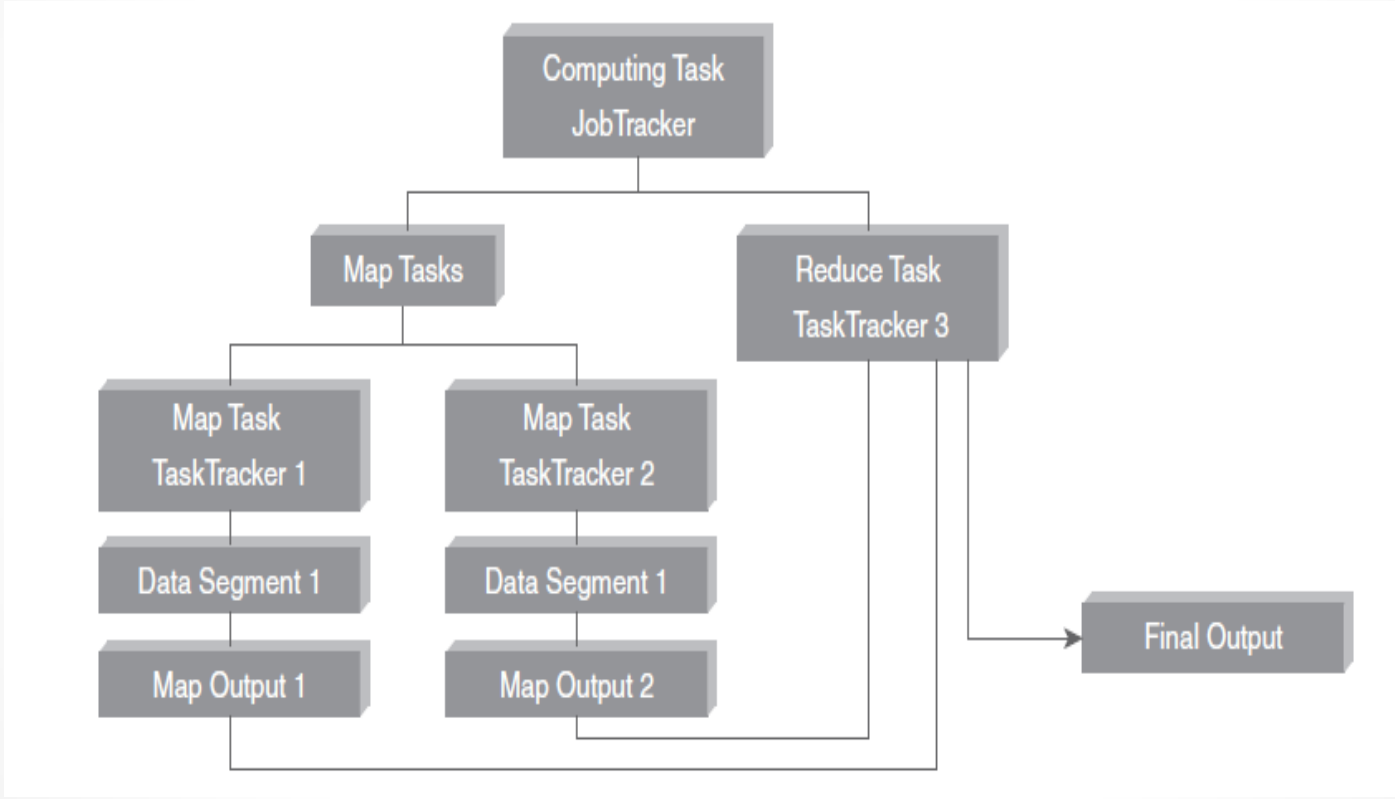
- To make wordcount work across a cluster of distributed machines
  - Store files over many processing machines (of phase one).
  - Write a disk-based hash table permitting processing without being limited by RAM capacity.
  - Partition the intermediate data (that is, wordCount) from phase one.
  - Shuffle the partitions to the appropriate machines in phase two.
  - Fault tolerance. (What if a machine fails in the middle of its task?)

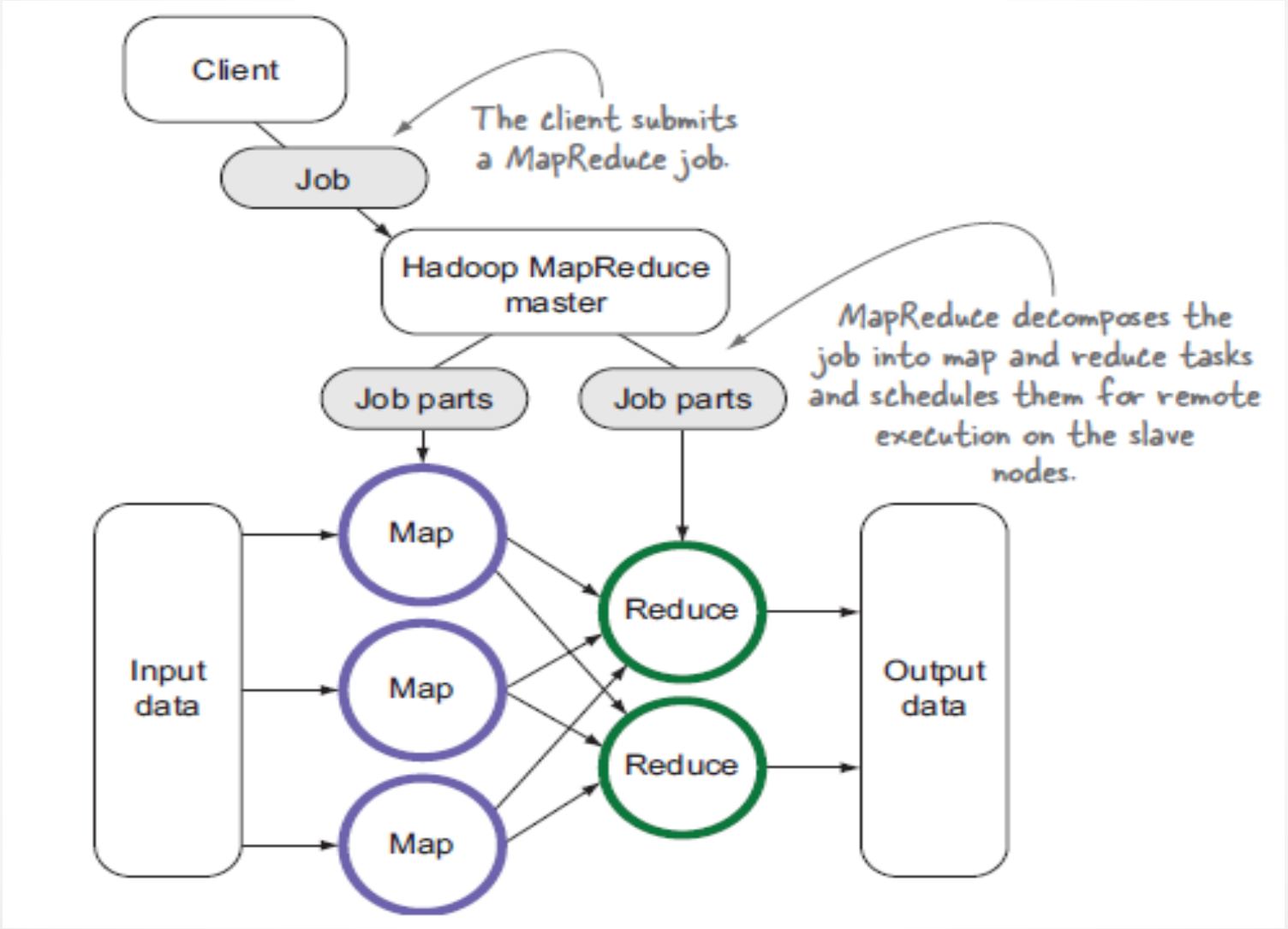
What is MapReduce Programming?

- MapReduce Programming is a software framework. MapReduce Programming helps you to process massive amounts of data in parallel

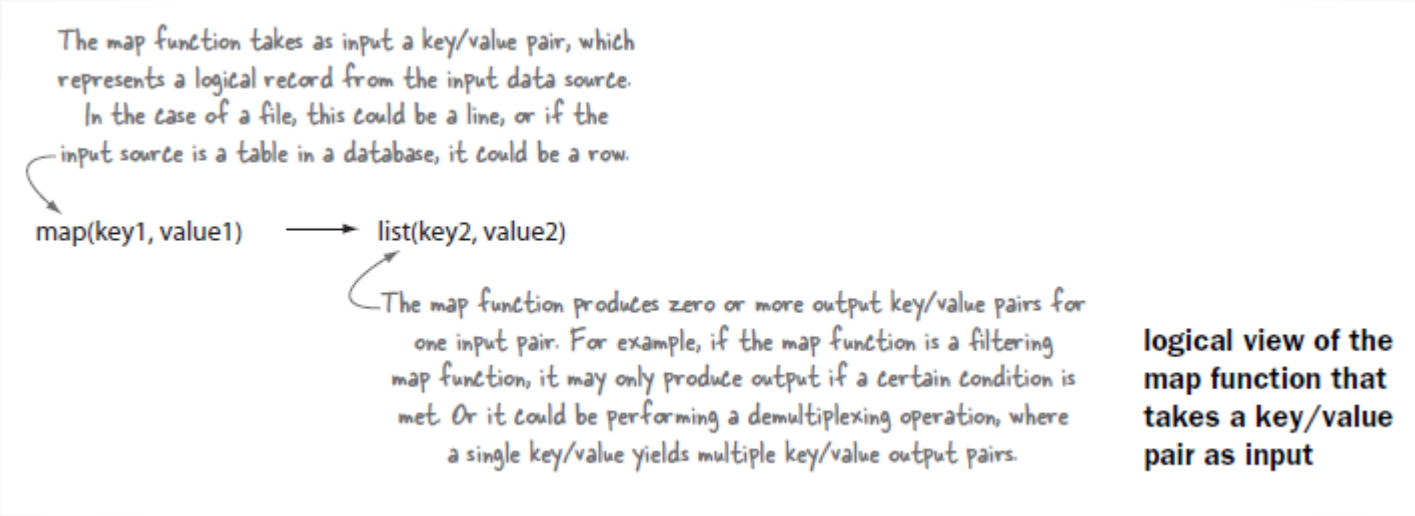


How MapReduce Programming Works



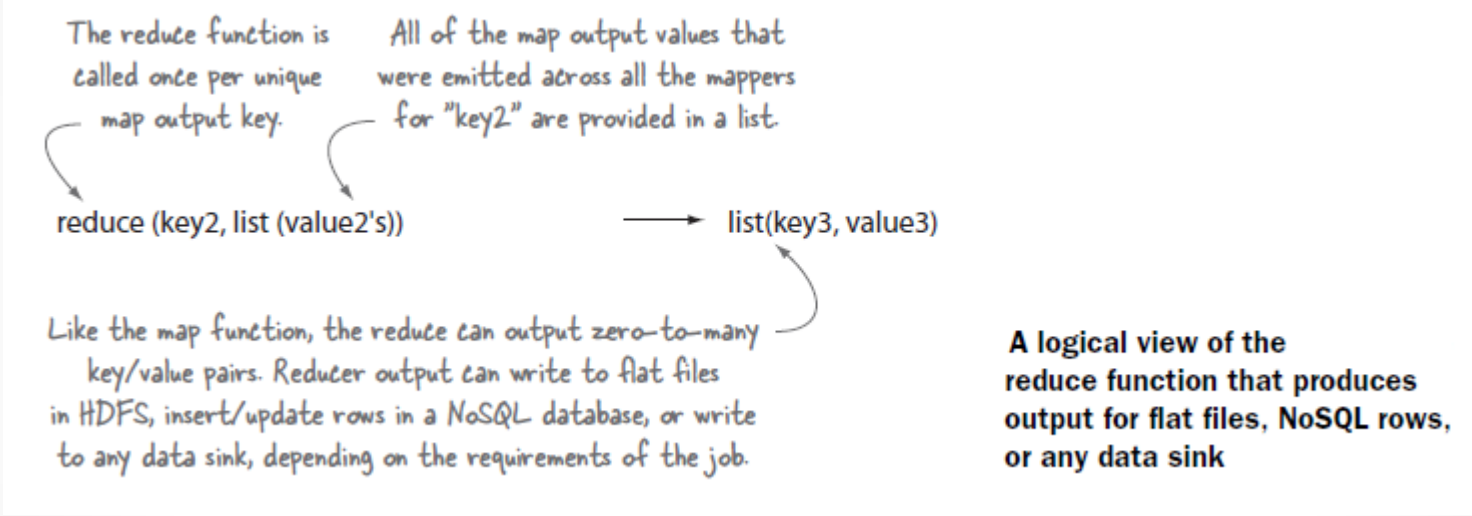

Client
The client submits a MapReduce job.
Job
Hadoop MapReduce master
MapReduce decomposes the job into map and reduce tasks and schedules them for remote execution on the slave nodes.
Job parts
Job parts
Input data
Map
Map
Map
Reduce
Reduce
Output data

Mapper



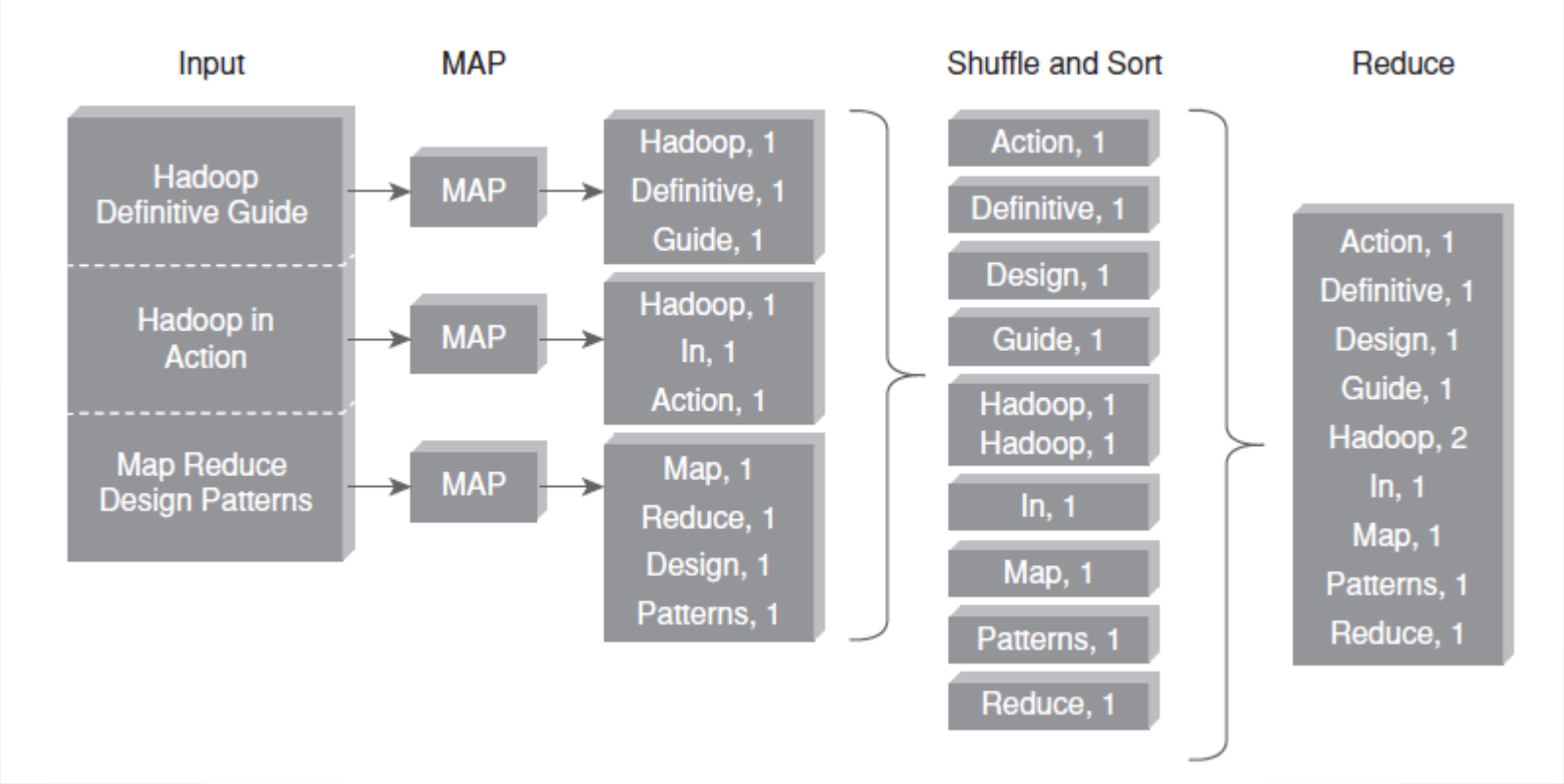
logical view of the map function that takes a key/value pair as input

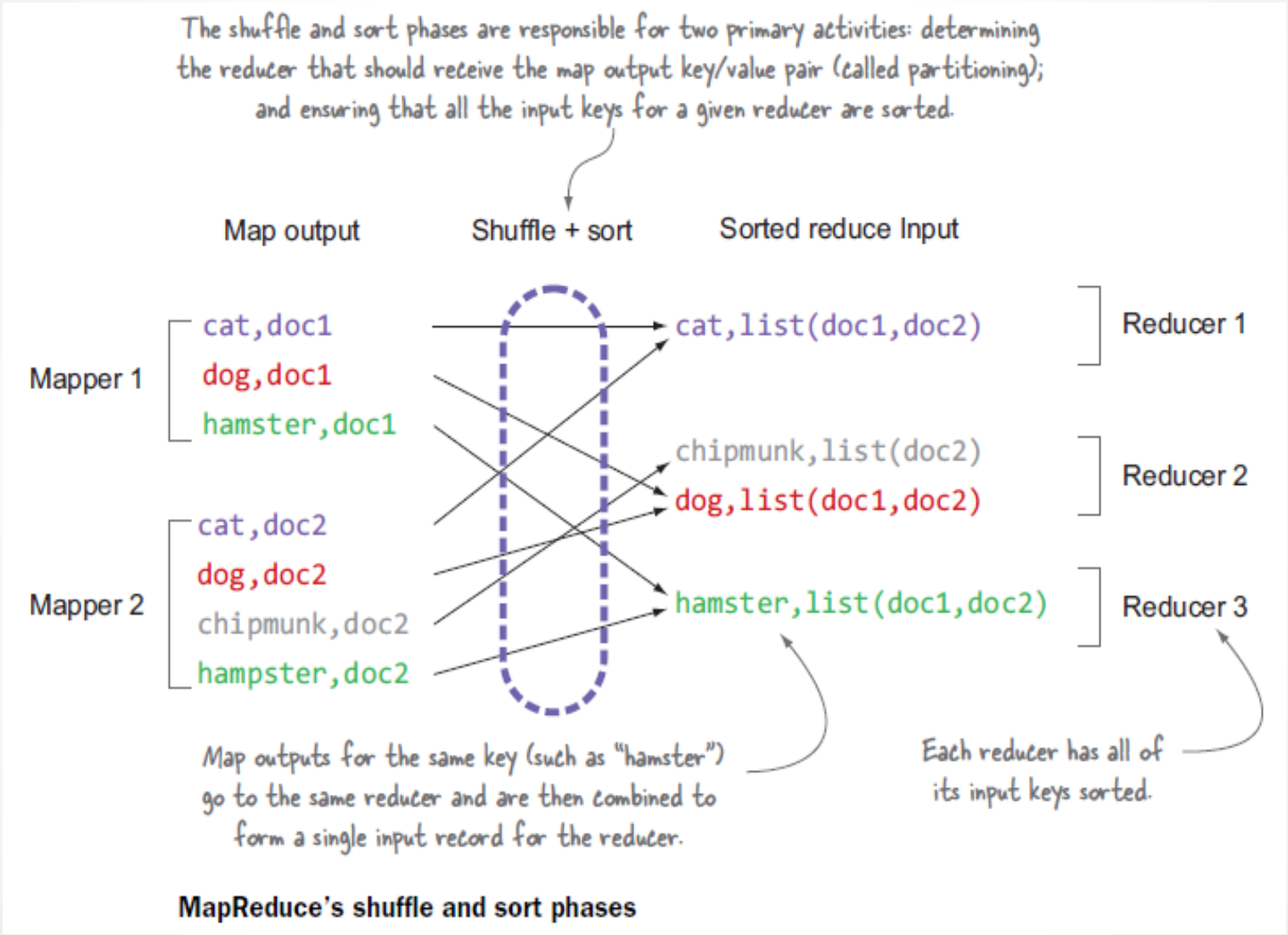
Reducer



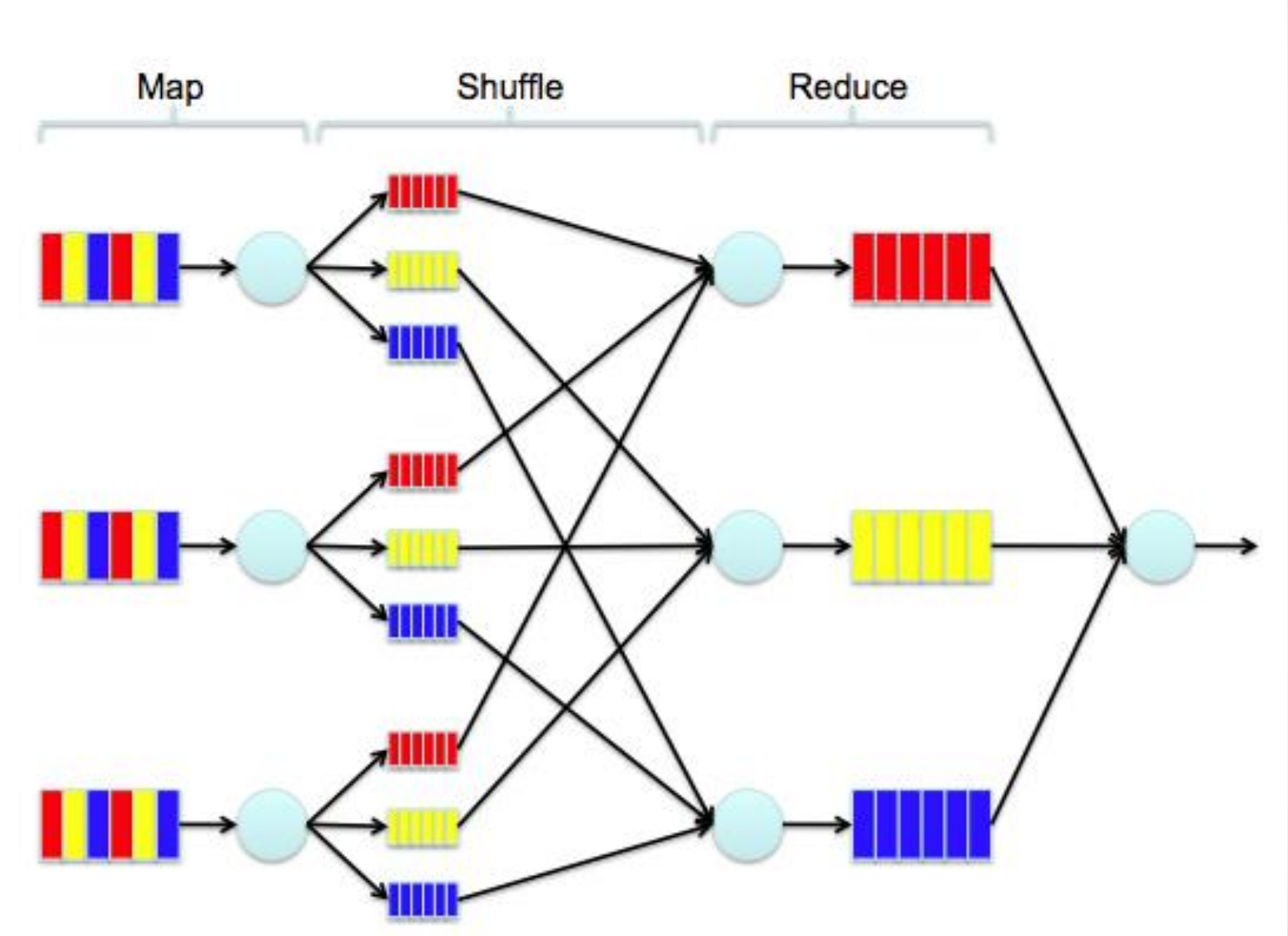
A logical view of the reduce function that produces output for flat files, NoSQL rows, or any data sink

Wordcount Example





MapReduce's shuffle and sort phases


Map
Shuffle
Reduce

Left outer join