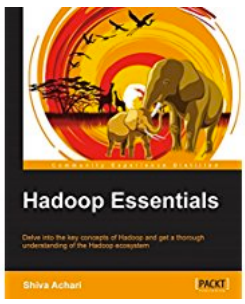
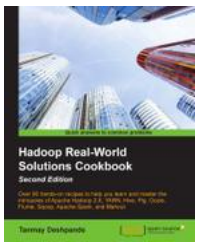
MapReduce

roberiogomes@gmail.com

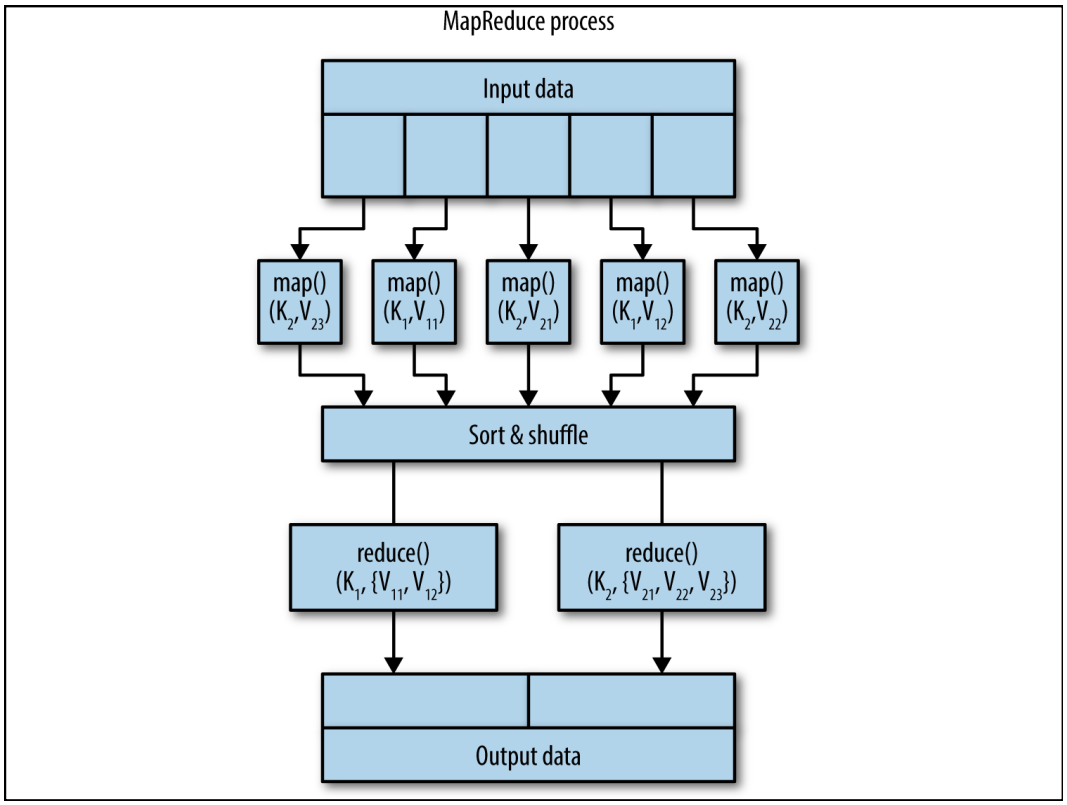
Good References





What Is MapReduce?

- MapReduce is a programming paradigm that allows for **massive scalability across hundreds or thousands of servers in a cluster environment.**
- The term MapReduce originated from functional programming and was introduced by Google in a paper called “MapReduce: Simplified Data Processing on Large Clusters.”



What Is MapReduce?

- Simply put, MapReduce is about scalability.
- Using the MapReduce paradigm, you focus on writing two functions:
  - map() → Filters and aggregates data
  - Reduce() → Reduces, groups, and summarizes by keys generated by map()

map() function

- The master node takes the input, partitions it into smaller data chunks, and distributes them to worker (slave) nodes.
- The worker nodes apply the same transformation function to each data chunk, then pass the results back to the master node.

$$\text{map}(): (\text{Key}_1, \text{Value}_1) \rightarrow [(\text{Key}_2, \text{Value}_2)]$$

reduce() function

- The master node shuffles and clusters the received results based on unique key-value pairs;
- Then, through another redistribution to the workers/slaves, these values are combined via another type of transformation function.

```
reduce(): (Key2, [Value2]) → [(Key3, Value3)]
```

Map & Reduce in action

Table P-1. Mappers' output

Key	Value
K_1	V_{11}
K_2	V_{21}
K_1	V_{12}
K_2	V_{22}
K_2	V_{23}

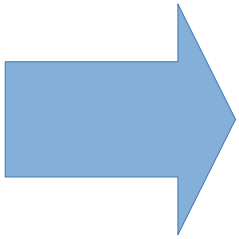
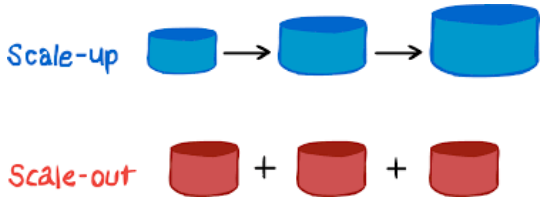


Table P-2. Reducers' input

Key	Value
K_1	$\{V_{11}, V_{12}\}$
K_2	$\{V_{21}, V_{22}, V_{23}\}$

Map & Reduce in action

- When writing your map() and reduce() functions, make sure that your solution is scalable.
- Scalability is the heart of MapReduce.
- When we talk about scalability, we mean scaling out .



Simple Explanation of MapReduce

- Let's say that we want to count the number of books in a library that has 1,000 shelves and report the final result to the librarian.
- Solution #1 (using map() and reduce()):
  - map(): Hire 1,000 workers; each worker counts one shelf.
  - reduce(): All workers get together and add up their individual counts.

Simple Explanation of MapReduce

- Solution #2 (using map(), combine(), and reduce()):
  - map(): Hire 1,110 workers (1,000 workers, 100 managers, 10 supervisors—each supervisor manages 10 managers, and each manager manages 10 workers); each worker counts one shelf, and reports its count to its manager.
  - combine(): Every 10 managers add up their individual counts and report the total to a supervisor.
  - reduce(): All supervisors get together and add up their individual counts (by reporting the results to the librarian).

Hello World Map Reduce

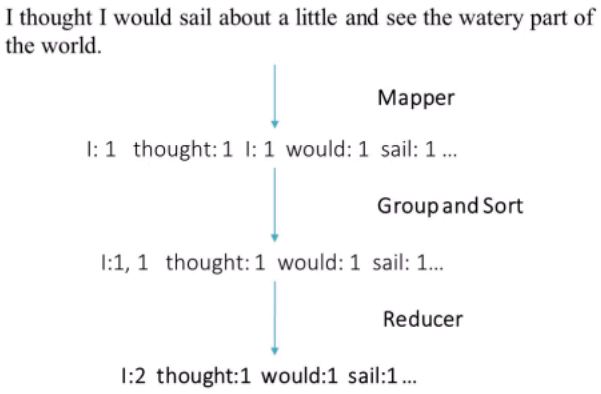
Input Data

Text from a book:

"I thought I would sail about a little and see the watery part of the world."

Hello World Map Reduce

Making it a MapReduce Problem



Hello World Map Reduce - Code

```
from mrjob.job import MRJob

class MRWordFrequency(MRJob):
    def mapper(self, _, line):
        words = line.split()
        for word in words:
            yield word.lower(), 1

    def reducer(self, word, values):
        yield word, sum(values)

if __name__ == '__main__':
    MRWordFrequency.run()
```

Hello World – Let's practice

Learning Activity

Total order amounts by customer

Input Data

CUSTOMER, ITEM, ORDER AMOUNT
44,8602,37.19
35,5368,65.89
44,3391,40.64
47,6694,14.98
29,680,13.08
91,8900,24.59
70,3959,68.68

What will your Mapper and Reducer Do?

CUSTOMER, ITEM, ORDER AMOUNT
44,8602,37.19
35,5368,65.89
44,3391,40.64

Mapper

?

Group and Sort

?

Reducer

35: 65.89 44: 77.83

Average Number of Friends by Age

Input data sample

User ID, Name, Age, Number of Friends

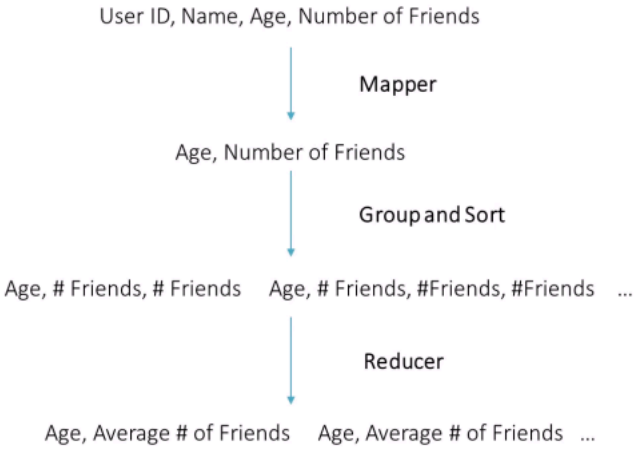
0,Will,33,385
1,Jean-Luc,26,2
2,Hugh,55,221
3,Deanna,40,465
4,Quark,68,21
5,Weyoun,59,318
6,Gowron,37,220
7,Will,54,307
8,Jadzia,38,380
9,Hugh,27,181
10,Odo,53,191

What fields do we care about?

User ID, Name, Age, Number of Friends

```
0,Will,33,385
1,Jean-Luc,26,2
2,Hugh,55,221
3,Deanna,40,465
4,Quark,68,21
5,Weyoun,59,318
6,Gowron,37,220
7,Will,54,307
8,Jadzia,38,380
9,Hugh,27,181
10,Odo,53,191
```

Making it a MapReduce Problem

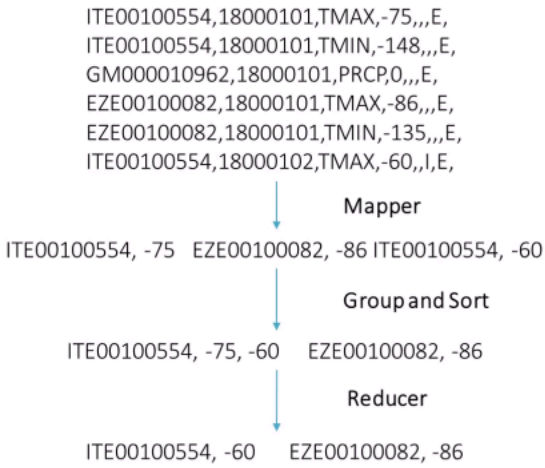


Finding Temperature Extremes

Input Data

ITE00100554,18000101,TMAX,-75,,,E,
ITE00100554,18000101,TMIN,-148,,,E,
GM000010962,18000101,PRCP,0,,,E,
EZE00100082,18000101,TMAX,-86,,,E,
EZE00100082,18000101,TMIN,-135,,,E,
ITE00100554,18000102,TMAX,-60,,I,E,

Making it a MapReduce Problem



When NOT to Use MapReduce

- If the computation of a value depends on previously computed values;
- If the data set is small enough to be computed on a single machine.
- If synchronization is required to access shared data.
- If all of your input data fits in memory.
- If one operation depends on other operations.
- If basic computations are processor-intensive.

When to Use MapReduce

- Is MapReduce good for everything?
- The simple answer is NO.
- When we have big data, if we can partition it and each partition can be processed independently, then we can start to think about MapReduce algorithms.

- Automatic parallelization and distribution

Why Use MapReduce?

- Fault tolerance (if a server dies, the job will be completed by other servers)
- Program/job scheduling, status checking, and monitoring