

# Advanced MapReduce

Cary Goltermann

Galvanize

2016

## Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners

## Hadoop Ecosystem Tools

- Hbase, Hive, Pig

MapReduce deals with most of the details of distributed computing (the ones that we would otherwise like to not think about) and automatically handles most of the concerns with:

- Parallelization and distribution of data and computation
- Fault-tolerance
- Resource management
- Status monitoring

## Advanced MapReduce

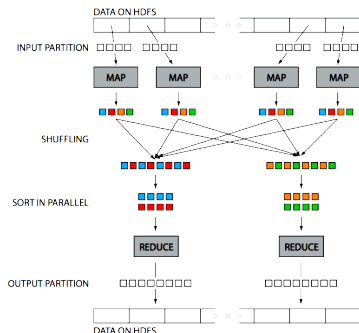
- Shuffle and Sort
- Counters
- Combiners

## Hadoop Ecosystem Tools

- Hbase, Hive, Pig

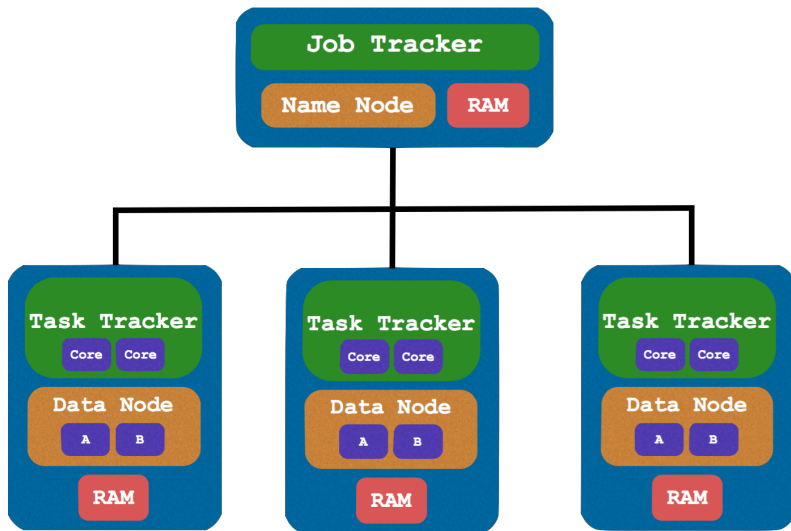
# Shuffle and Sort

How data is re-distributed during a MapReduce job to continually take advantage of all the worker nodes.

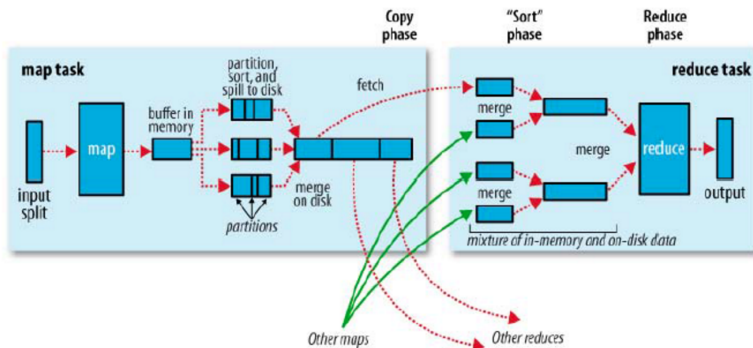


- **Shuffle Step:** Process of deciding how and where data from mappers will go to the reducers.
- **Sort Step:** Sorts data it gets from the mappers by key.

# Tracking



# Partitioning



The partitioner on each mapper takes care of any local sorting it can do to the data it's outputting and the reducers take care of merging all the sorted partitions that get sent to them during the shuffle.

In a distributed computing framework like Hadoop what is one of the most scarce resources?



In a distributed computing framework like Hadoop what is one of the most scarce resources?

## Bandwidth

The time cost of transmitting data around the cluster, between nodes, as in the shuffle and sort step, is quite high.

In a distributed computing framework like Hadoop what is one of the most scarce resources?

## Bandwidth

The time cost of transmitting data around the cluster, between nodes, as in the shuffle and sort step, is quite high.

What if we could decrease the amount of data we have to move in our shuffle and sort, or eliminate it all together?

## Advanced MapReduce

- Shuffle and Sort
- **Counters**
- Combiners

## Hadoop Ecosystem Tools

- Hbase, Hive, Pig

# Counters

- Used to keep track of events that occur over a given map or reduce step.
- Mostly used for debugging purposes. E.g. To verify that all of the folders in your file system are getting accessed correctly.

```
1  class MRCountingJob(MRJob):  
2  
3      def mapper(self, _, value):  
4          self.increment_counter('group', 'counter_name', 1)  
5          yield _, value
```

# Counters Code

Counters can also be used to avoid using a reducer in some cases. If all you're trying to do is count by some key then a counter can help you avoid the time consuming process of the shuffle and sort when you use a reducer.

```
1  import os
2
3  class MRWordCount(MRJob):
4
5      def mapper(self, _, line):
6          filepath = os.environ['map_input_file']
7          filename = filepath.split('/')[-1]
8
9          for word in line.split():
10             self.increment_counter(filename, 'word_counts', 1)
```

## Advanced MapReduce

- Shuffle and Sort
- Counters
- **Combiners**

## Hadoop Ecosystem Tools

- Hbase, Hive, Pig

- Help decrease the amount of data that needs to be passed from mappers to reducers.
- Frequently look like reducers that aggregate like data on the mapper so that there is "less" of it to copy to the reducers.
- **Note:** The same information needs to get passed to the reducers no matter what, combiners generally find a way to achieve this while using fewer bits to represent that information.

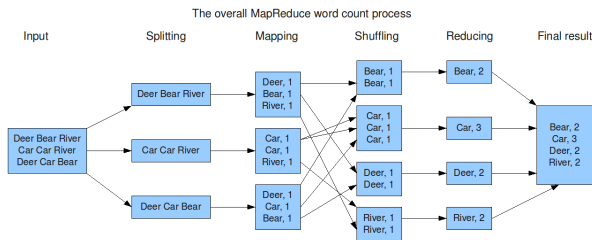
# Combiner Example Illustrated

Let's take a look at an example that illustrates unnecessarily moving data.



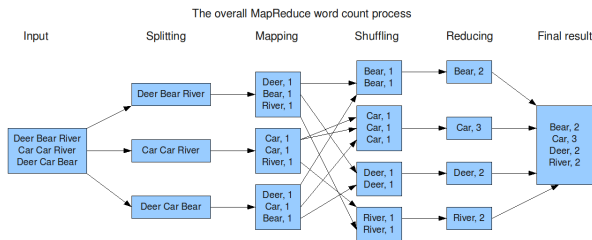
# Combiner Example Illustrated

Let's take a look at an example that illustrates unnecessarily moving data.



# Combiner Example Illustrated

Let's take a look at an example that illustrates unnecessarily moving data.



Does anything strike you as inefficient about this MapReduce implementation?

# Combiner Code

```
1  from string import punctuation
2
3  class MRWordCount(MRJob):
4
5      def mapper(self, _, line):
6          for word in line.split():
7              yield (word.strip(punctuation).lower(), 1)
8
9      def combiner(self, word, counts):
10         yield (word, sum(counts))
11
12     def reducer(self, word, counts):
13         yield (word, sum(counts))
```

**Note:** While the combiner and the reducer implement the same function in this example, this is not generally the case when using a combiner.

## Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners

## Hadoop Ecosystem Tools

- Hbase, Hive, Pig

APACHE  
**HBASE**



Pig

- Hbase** A mutable big data storage system on top of HDFS. **Note:** HDFS is mutable, batch oriented.
- Hive** A high level language for writing MapReduce jobs in an SQL style.
- Pig** Another high level language for writing MapReduce jobs in. Kind of like Hive, but with less structure and less familiarity from SQL.