Advanced MapReduce

Game Plan

- Shuffle and Sort in a MapReduce Job
- Hadoop Tools
  - HBase
  - Pig
  - Hive

Objectives

- Understand the mechanics of counters/ combiners and their role in a MapReduce job
- Know about a couple of tools that are built on top of Hadoop MapReduce

Game Plan

- Shuffle and Sort in a MapReduce Job
- A quick survey of Hadoop tools
  - HBase
  - Pig
  - Hive

Objectives

- Understand the mechanics of counters/combiners and their role in a MapReduce job
- Know about a couple of tools that are built on top of Hadoop MapReduce

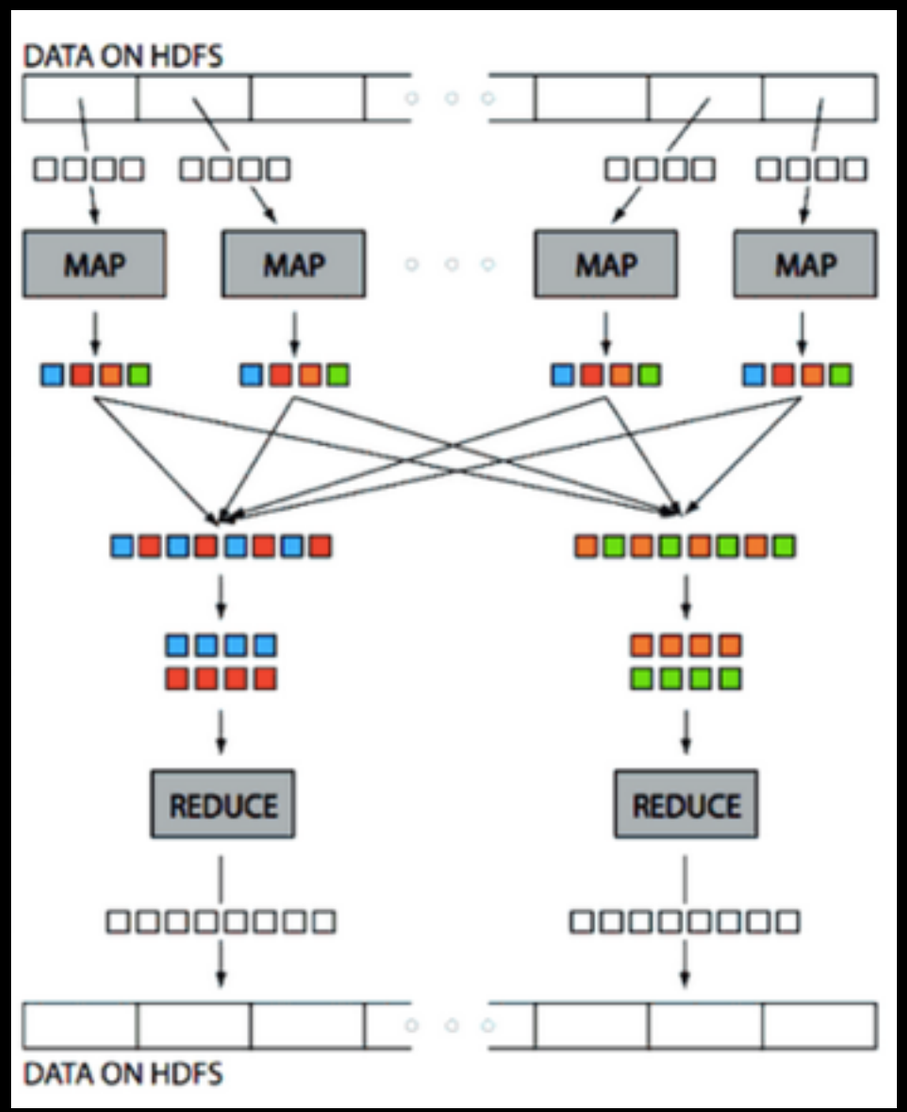
Hadoop MapReduce

Hadoop MapReduce handles a lot of the details of distributed computing for us and does most of the heavy lifting around:

- Parallelization & Distribution (input splits)
- Partitioning (shuffle & sort)
- Fault-tolerance
- Resource Management
- Status and monitoring

MapReduce Framework

Input data - stored on HDFS data nodes



Map step

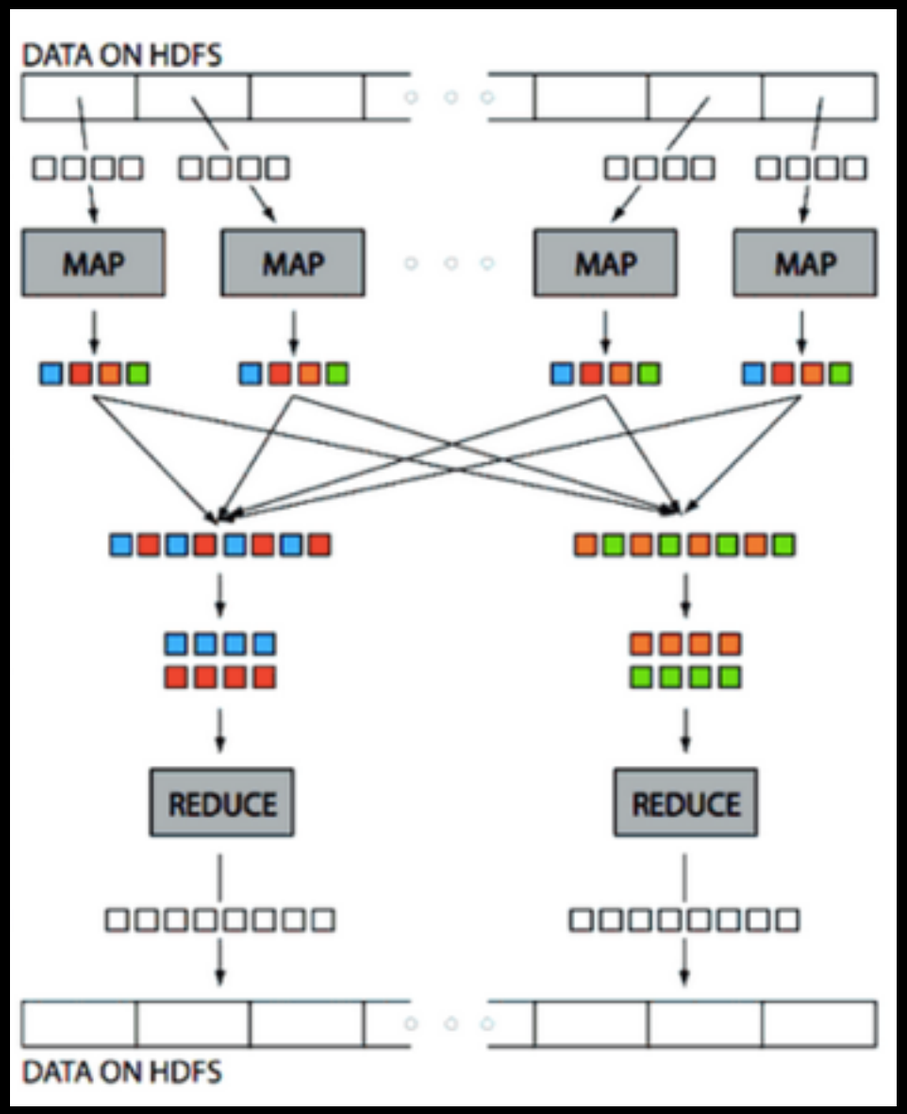
Shuffle step

Sort step

Reduce step

Output data - stored on HDFS data nodes

Shuffle & Sort



Shuffle step - the process of transferring data from the mappers to the reducers

Sort step - sorts the data inputted to the reducer by key (increases efficiency)

Cost

The cost of the shuffle & sort is relatively high - moving that much data around takes awhile, as does sorting it. What if we could avoid it all together, or reduce the volume of the data that goes through the shuffling & sorting steps?

Counters

- Can be used to keep the counts of a **fixed** number of categories that you know **ahead** time
- Mostly used in debugging, but could be used to avoid the shuffle & sort steps in the right scenario

Counter Code

```
from mrjob.job import MRJob
from string import punctuation

class MRWordFreqCount(MRJob):

    def mapper(self, _, line):
        filename = os.environ['map_input_file']
        filename_parts = filename.split('/')
        grouping_part = filename_parts[1].split('.')


        for word in line.split():
            self.increment_counter(grouping_part[0], 1)

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

Increments one of the **known** categories by 1

Combiners

- Can help to lower the amount of data that is passed from the map step to the reduce step
- Are effectively Reducers that the map step uses to reduce the data locally (e.g. on the mapper)

Combiner Code

```
'''The classic MapReduce job: count the frequency of words.'''

from mrjob.job import MRJob
from string import punctuation


class MRWordFreqCount(MRJob):

    def mapper(self, _, line):
        for word in line.split():
            yield (word.strip(punctuation).lower(), 1)

    def combiner(self, word, counts):
        yield (word, sum(counts))

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

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

- Data is read in in the same format that it is read into the reducer - a key with it's grouped list of values
- Data is output in the same format that it is output from the reducer - as **key-value pairs**

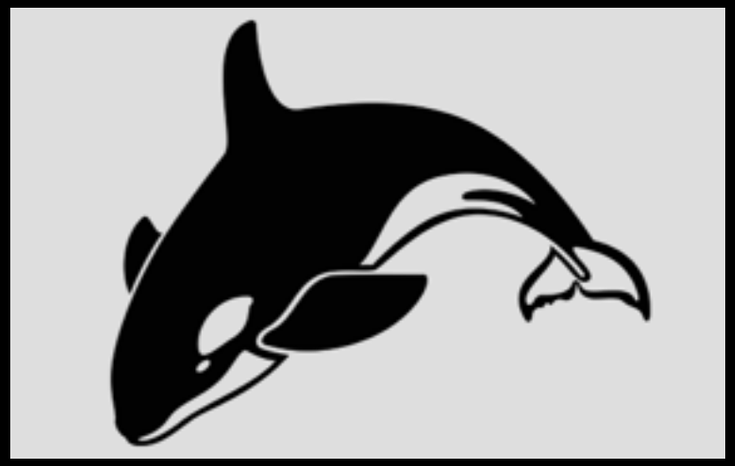
Game Plan

- Shuffle and Sort in a MapReduce Job
- A quick survey of Hadoop tools
  - HBase
  - Pig
  - Hive

Objectives

- Understand the mechanics of counters/combiners and their role in a MapReduce job
- Know about a couple of tools that are built on top of Hadoop MapReduce

Hadoop Tools



HBase

Pig

Hive

Hadoop Tools

- **HBase -** a mutable Big Data storage system on top of HDFS (HDFS is immutable, batch oriented)

- **Pig -** a high-level language for writing MapReduce jobs
    - a SQL-like scripting language
    - a little bit less structured than Hive

- **Hive** - another high-level language for writing MapReduce jobs
    - let's you write MapReduce jobs in SQL

Game Plan

- Shuffle and Sort in a MapReduce Job
- A quick survey of Hadoop tools
  - HBase
  - Pig
  - Hive

Objectives

- Understand the mechanics of counters/combiners and their role in a MapReduce job
- Know about a couple of tools that are built on top of Hadoop MapReduce