Advanced MapReduce

Cary Goltermann

Galvanize

2017

Overview

Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners
- Multi-step Jobs

Hadoop Ecosystem Tools

- Hbase, Hive, Pig

MapReduce

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

Overview

Advanced MapReduce

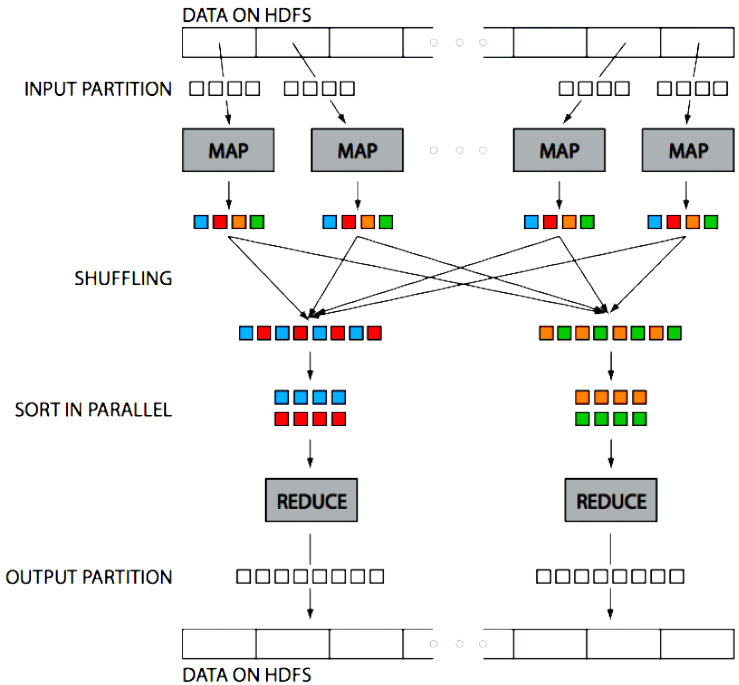
- Shuffle and Sort
- Counters
- Combiners
- Multi-step Jobs

Hadoop Ecosystem Tools

- Hbase, Hive, Pig

Shuffle and Sort

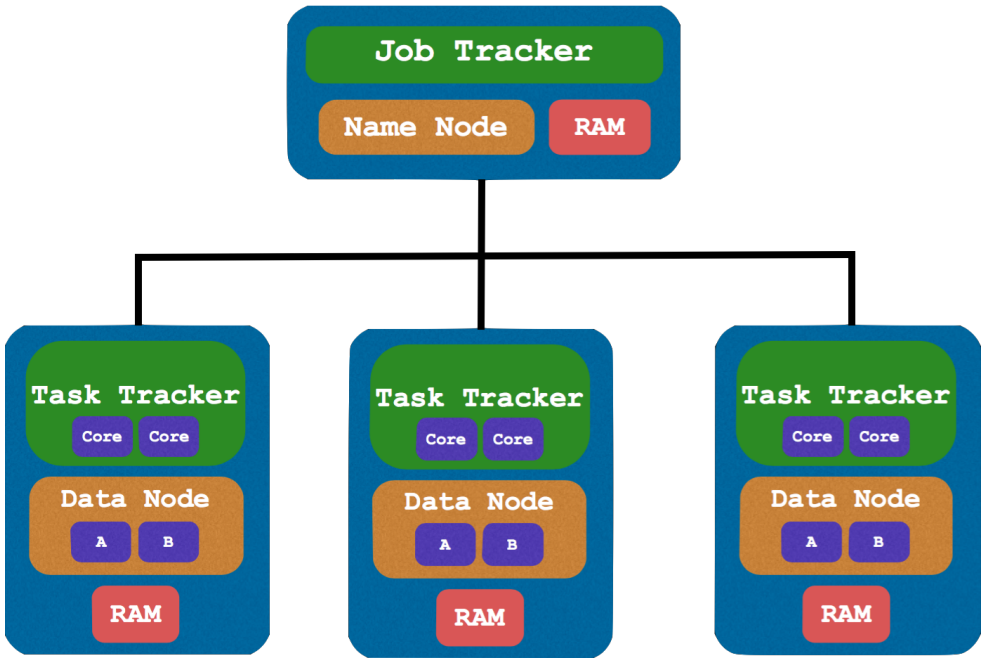
Data is re-distributed during a MapReduce job to continually take advantage of all the worker nodes.



- 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

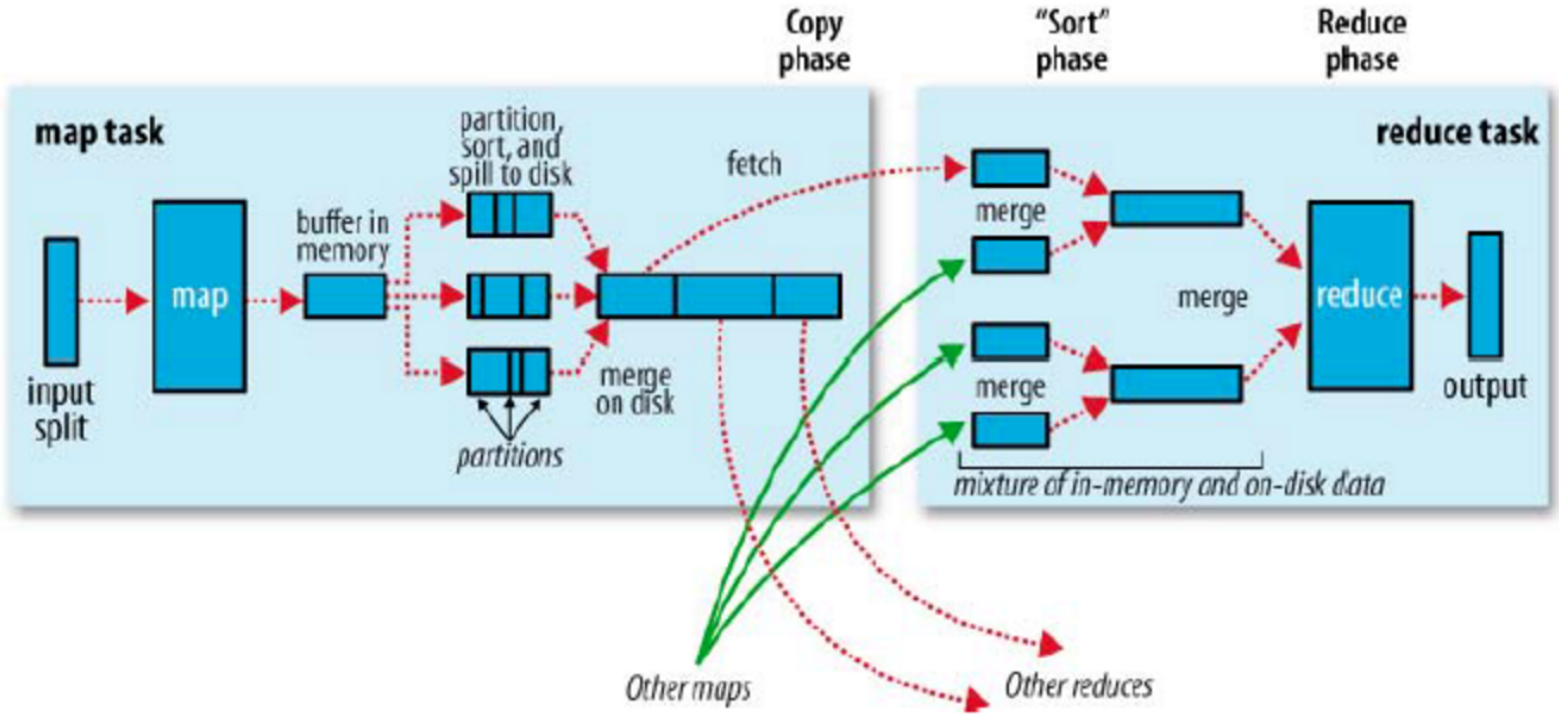
Keeping track of a MapReduce job involves two parts, the data and the steps. The location of all the data is registered with the name node. The status of a job is kept track of by job tracker, frequently this is on the name node.



Each of the nodes has as task tracker that communicates with the job track about its status on its assigned job.

The location of the data is registered with each node, known as a data node, with the name node.

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.

Scarce Resources

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

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.

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.

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

Overview

Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners
- Multi-step Jobs

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)
```

Overview

Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners
- Multi-step Jobs

Hadoop Ecosystem Tools

- Hbase, Hive, Pig

Combiners

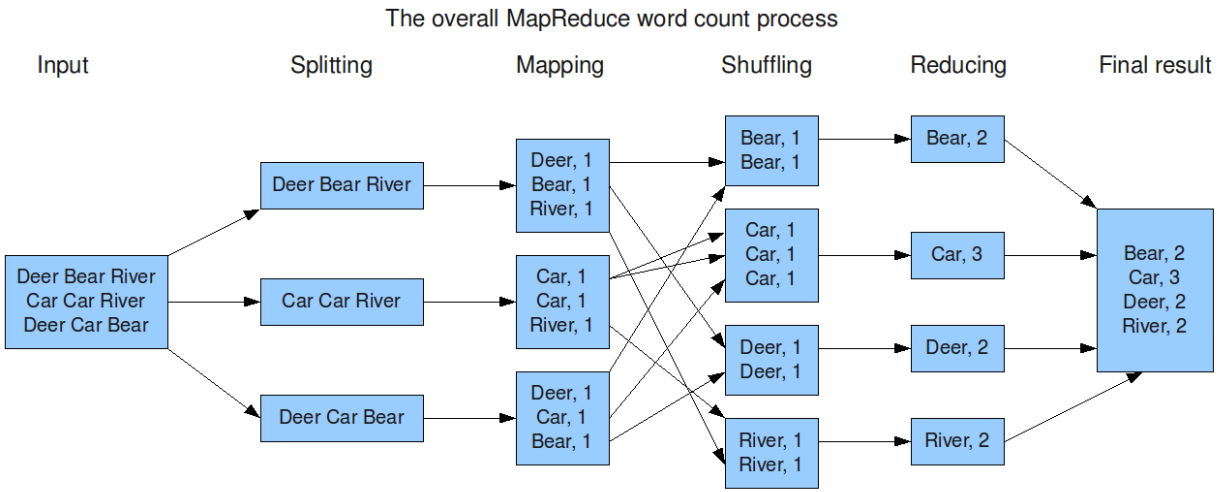
- Help decrease the amount of data that needs to be passed from mappers to reducers.
- Frequently look like reducers that aggregate like data on the mapper so that there is "less" of it to copy to the reducers.
- Note: The same information needs to get passed to the reducers no matter what, combiners 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.

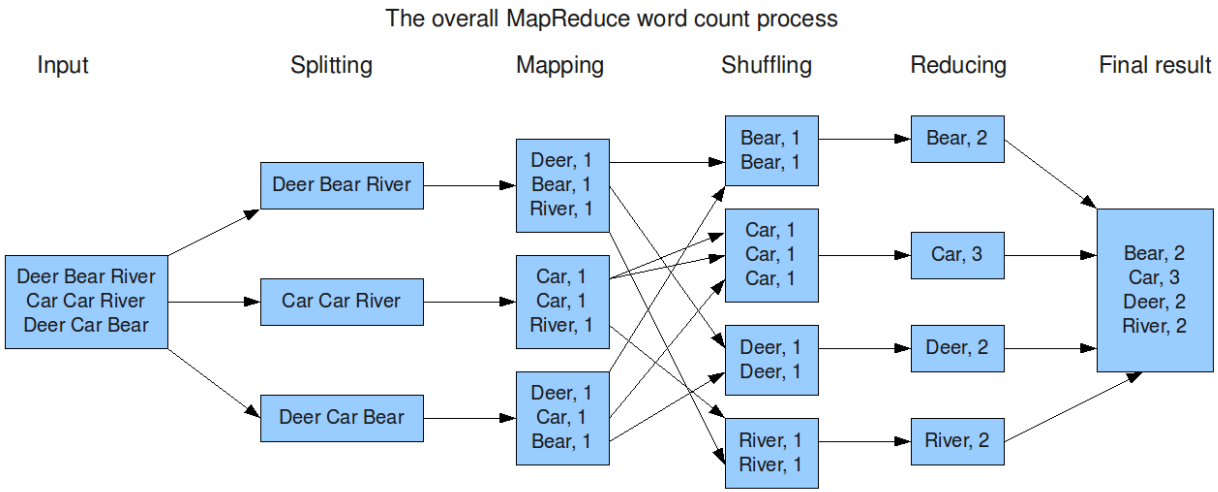
Combiner Example Illustrated

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.

Overview

Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners
- Multi-step Jobs

Hadoop Ecosystem Tools

- Hbase, Hive, Pig

More than One Step

Sometimes we need more than map-reduce pass to answer the question we're interested in.

These cases are called multi-step jobs. Creating a multi-step job with MRJob is really easy!

More than One Step

Sometimes we need more than map-reduce pass to answer the question we're interested in.

These cases are called multi-step jobs. Creating a multi-step job with MRJob is really easy!

Consider trying to answer the question:

In Need of Multiple Steps

What is the most common word per letter in a document?

Multiple Steps in MRJob

```
1   class MRMostFrequentWord(MRJob):
2       def mapper1(self, _, line):
3           clean_line = ''.join(c for c in line if c not in PUNCT)
4           for word in clean_line.lower().split():
5               yield word, 1
6       def reducer1(self, word, counts):
7           yield word, sum(counts)
8       def mapper2(self, word, count):
9           yield word[0], (word, count)
10      def reducer2(self, letter, word_lengths):
11          yield letter, max(word_lengths, key=lambda x: x[1])
12      def steps(self):
13          return [MRStep(mapper=self.mapper1,
14                         reducer=self.reducer1),
15                  MRStep(mapper=self.mapper2,
16                         reducer=self.reducer2)]
```

Overview

Advanced MapReduce

- Shuffle and Sort
- Counters
- Combiners
- Multi-step Jobs

Hadoop Ecosystem Tools

- Hbase, Hive, Pig

Hadoop Ecosystem

Pig

All of the Tools

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.