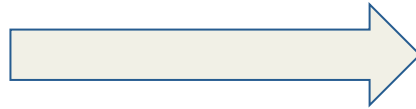


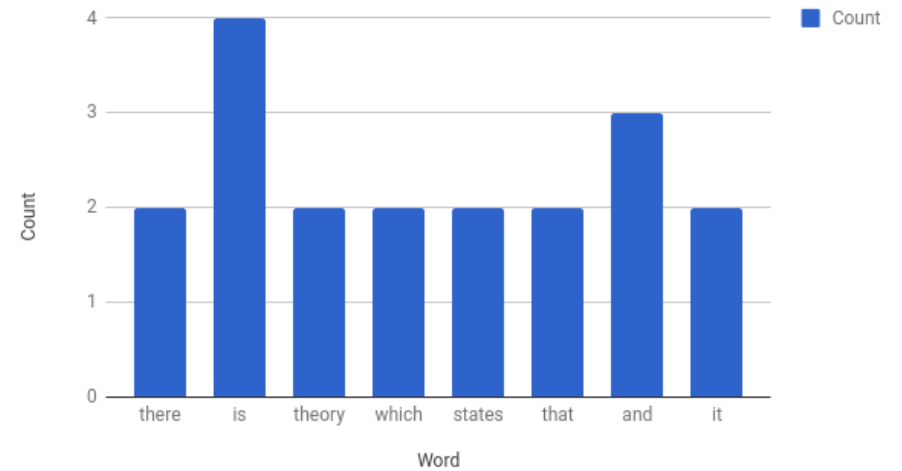
# WordCount

*There **is** a theory which states that if ever anyone discovers exactly what the Universe **is** for and why it **is** here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There **is** another theory which states that this has already happened.*

WordCount



Count vs. Word



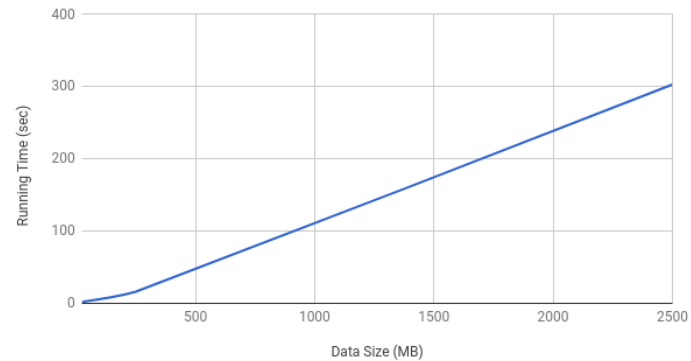
# A Naive Python Implementation

Let's run an experiment  
documents..

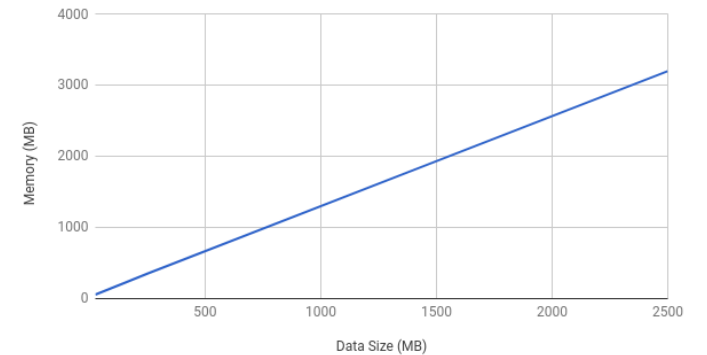


```
1 from collections import defaultdict
2 import sys
3
4 def wordcount(inputfile):
5     histo = defaultdict(int)
6     with open(inputfile, 'r') as f:
7         doclines = f.readlines()
8         for line in doclines:
9             words = line.split()
10            for w in words:
11                histo[w.lower()] += 1
12        f.close()
13
14    for w in histo:
15        print '{} {}'.format(w, histo[w])
16
17 if __name__ == '__main__':
18    wordcount(sys.argv[1])
```

Running Time (sec) vs. Data Size (MB)



Memory (MB) vs. Data Size (MB)



The bigger the dataset, the more resources we need!

# Deal With Scalability

If resources are not enough...

## ❖ Scale-Up: Get a **bigger** machine

- “Huge” machines not accessible to everyone (-)
- Not easy to migrate a live application (-)
- Code remains as is (+)

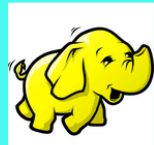
## ❖ Scale-Out: Use **more** machines

- Commodity-hardware (+)
- Elasticity actions (+)
- Distributed Algorithm is required (-)



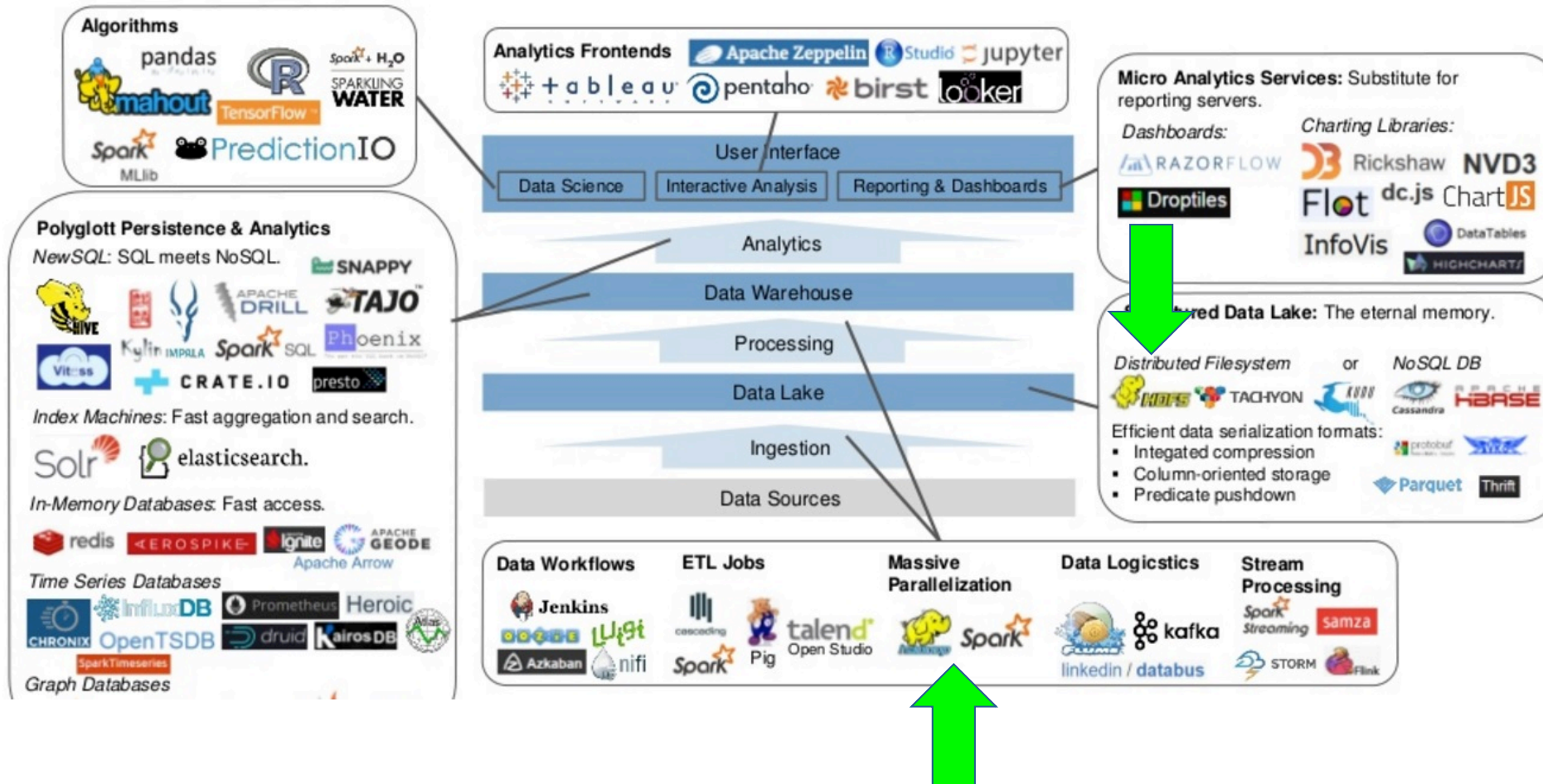
# Massive Datasets



  
**KEEP  
CALM  
AND  
USE  
HADOOP**

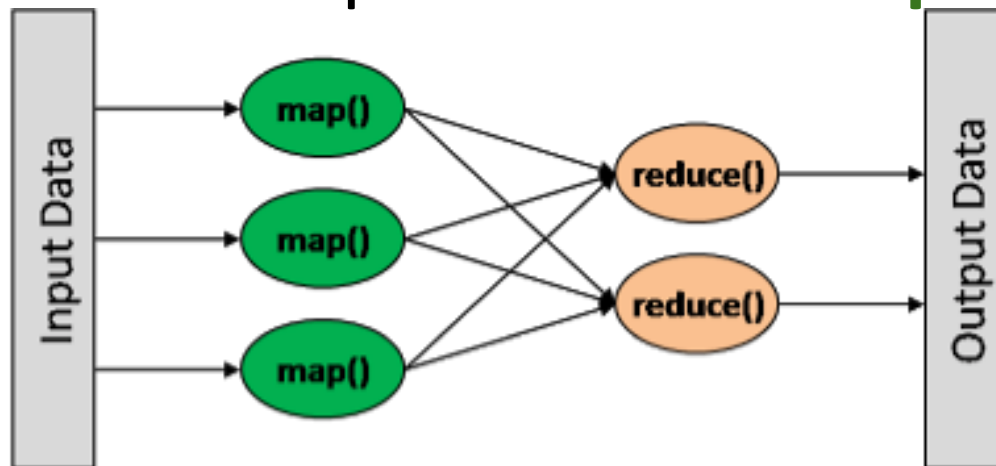


# Big Data Frameworks

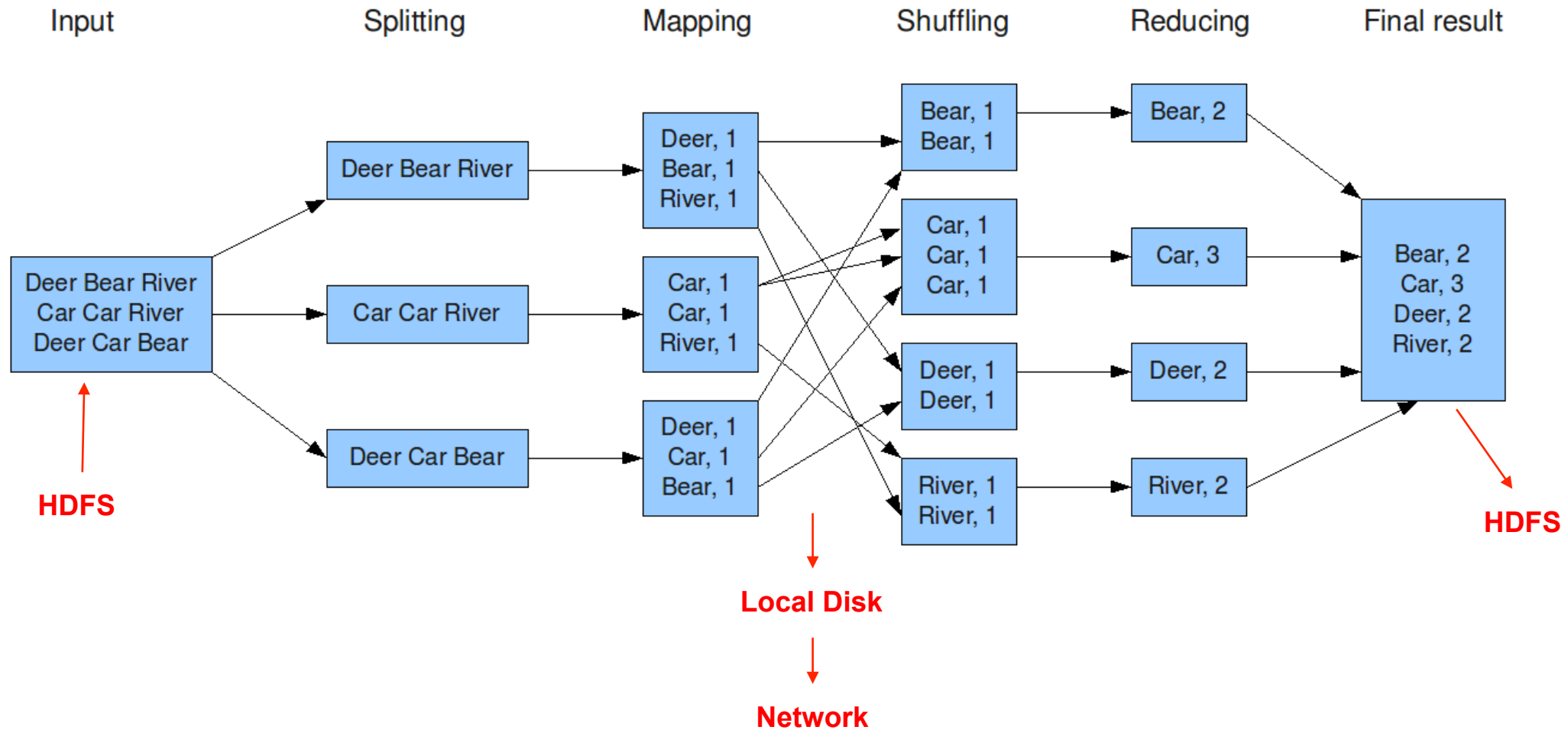


# The MapReduce Paradigm

- Programming framework
- Data parallel applications
- Computer Cluster
- Google proprietary implementation
  - [MapReduce: simplified data processing on large clusters](#)
- Open-source implementation: **Apache Hadoop**



# WordCount in Hadoop



# Writing a Hadoop Program

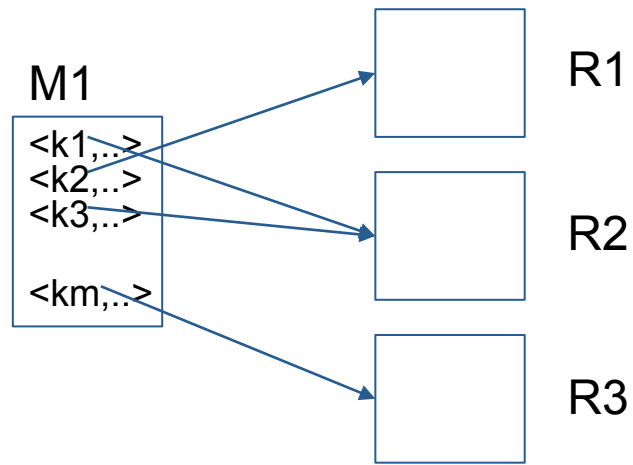
```
28     Job job = Job.getInstance(new Configuration());
29     job.setJarByClass(WordCountDriver.class);
30     job.setOutputKeyClass(Text.class);
31     job.setOutputValueClass(IntWritable.class);
32     job.setMapperClass(WordCountMapper.class);
33     job.setReducerClass(WordCountReducer.class);
34     job.setInputFormatClass(TextInputFormat.class);
35     job.setOutputFormatClass(TextOutputFormat.class);
36     FileInputFormat.setInputPaths(job, new Path(args[0]));
37     FileOutputFormat.setOutputPath(job, new Path(args[1]));
38     boolean status = job.waitForCompletion(true);
39     if (status) {
40         System.exit(0);
41     }
42     else {
43         System.exit(1);
44     }
45
```



# Mapper & Reducer Code

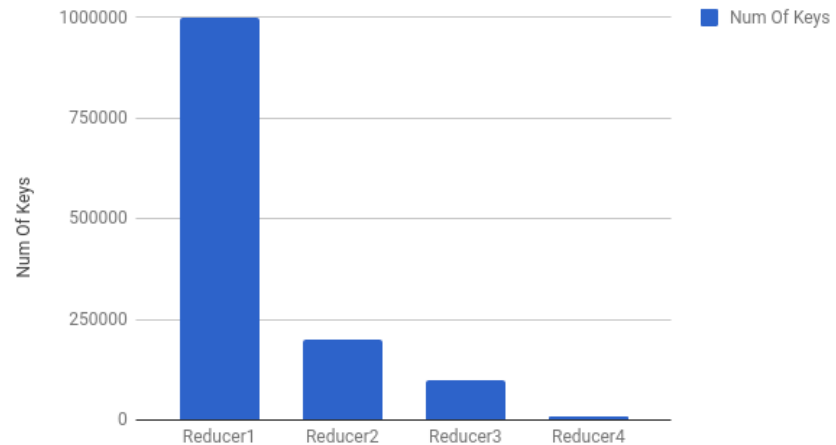
```
L0 public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
L1
L2 public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
L3     String[] words = value.toString().split(" ");
L4     for(String w: words){
L5         context.write(new Text(w), new IntWritable(1));
L6     }
L7 }
L8
L9 }
L0 public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
L1
L2 public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
L3     int sum = 0;
L4     for (IntWritable val : values) {
L5         sum += val.get();
L6     }
L7     context.write(key, new IntWritable(sum));
L8 }
L9
L0 }
```

# Partitioner

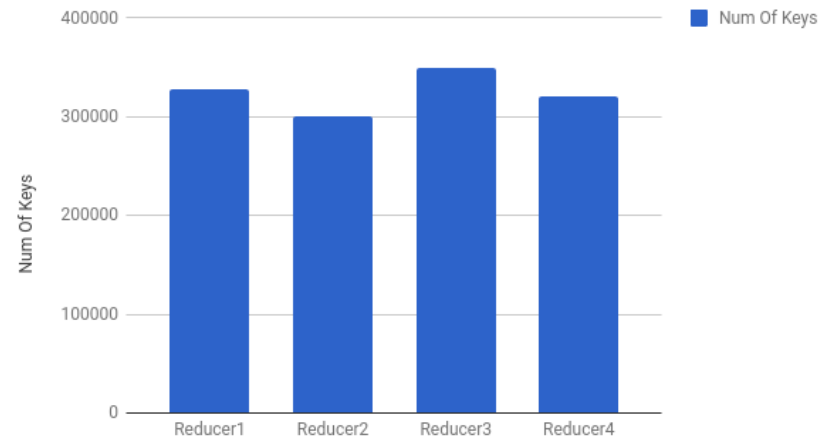


```
public class WordCountPartitioner extends Partitioner<Text, IntWritable> {  
  
    @Override  
    public int getPartition(Text key, IntWritable value, int numPartitions) {  
        return key.hashCode() % numPartitions;  
    }  
}
```

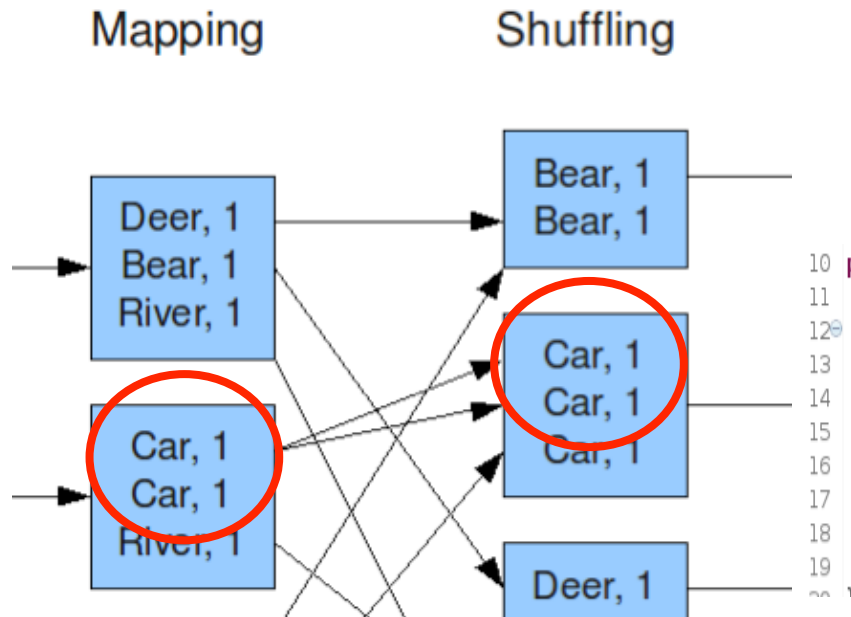
Unbalanced Partitioning



Balanced Partitioning



# Reducing Network Traffic



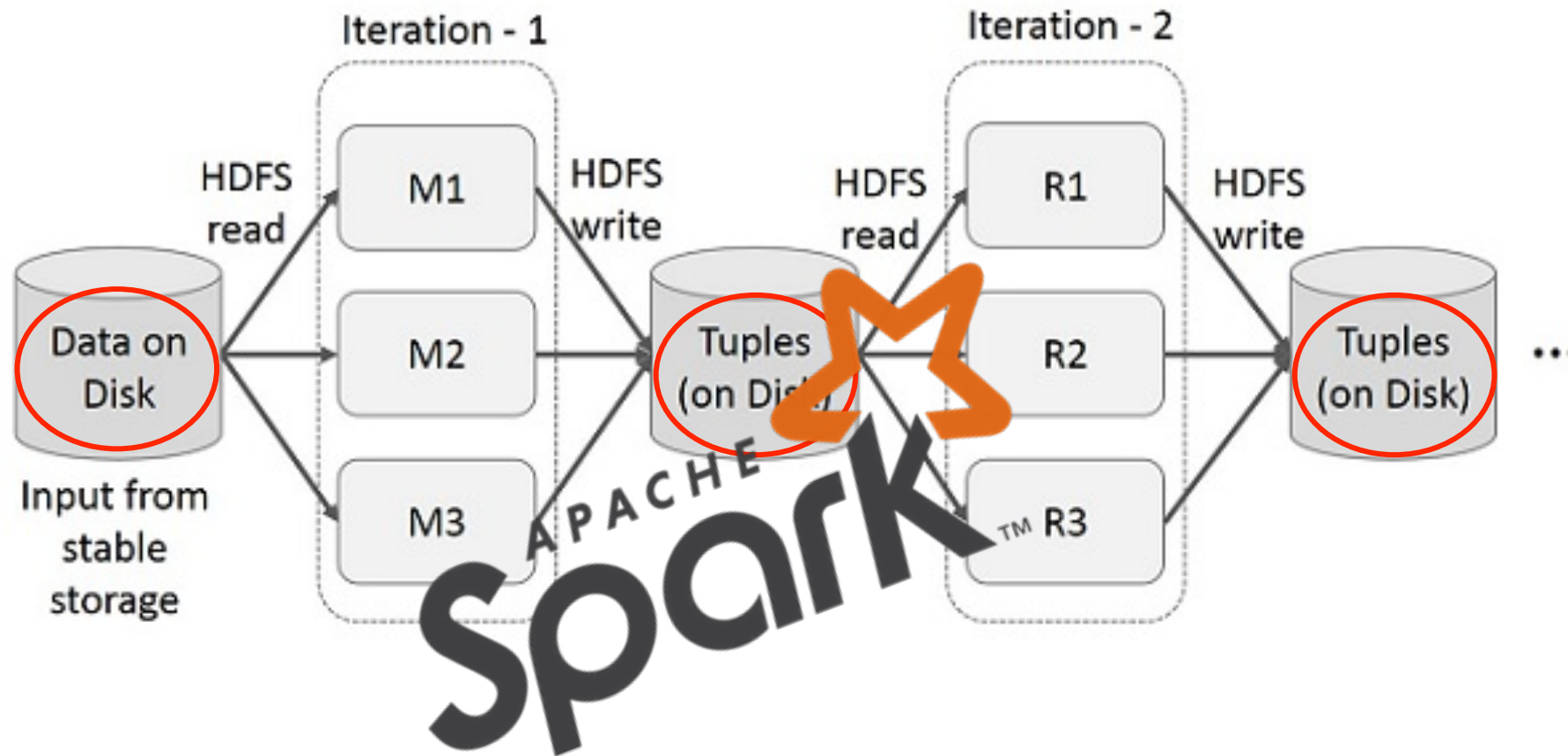
## Combiner

- Emits <Car, 2>
- Map-side reducer

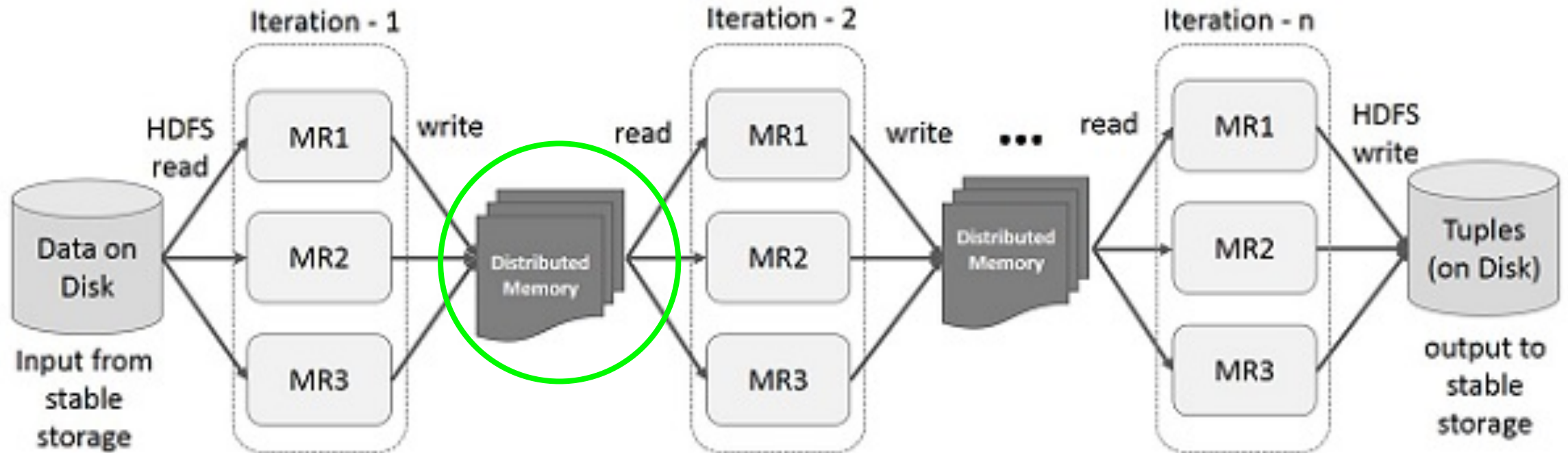
```
10 public class WordCountCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {
11
12     public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
13         int sum = 0;
14         for (IntWritable val : values) {
15             sum += val.get();
16         }
17         context.write(key, new IntWritable(sum));
18     }
19 }
20
```

```
Job job = Job.getInstance(new Configuration());
job.setJarByClass(WordCountDriver.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setMapperClass(WordCountMapper.class);
job.setReducerClass(WordCountReducer.class);
job.setCombinerClass(WordCountCombiner.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
boolean status = job.waitForCompletion(true);
```

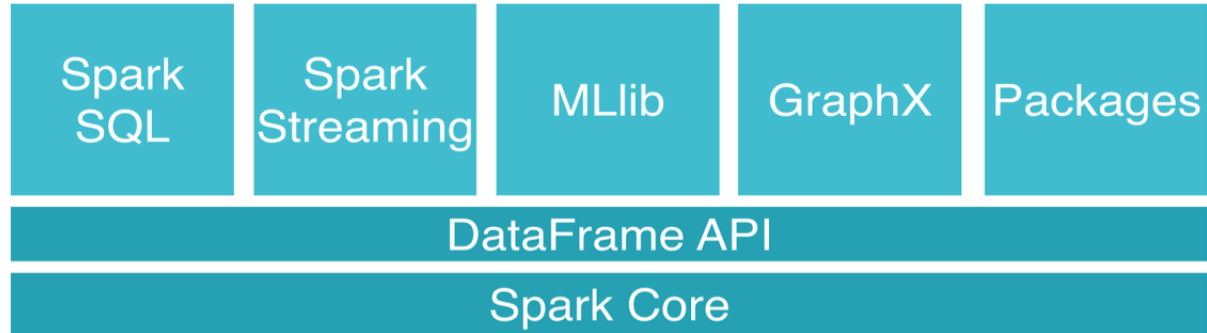
# Iterative Operations on Hadoop



# Iterative Operations on Spark



# APACHE Spark™



# Resilient Distributed Datasets (RDD)

- Immutable distributed collection of objects
- Divided into logical partitions
- Can be persisted **in memory**
- How to create an RDD:
  - Parallelize existing collection
  - Reference dataset in an external storage system (e.g., HDFS)
- **Transformations**
  - **Lazy evaluation**
- **Actions**
  - Trigger actual computation

# WordCount in Spark

**RDD**

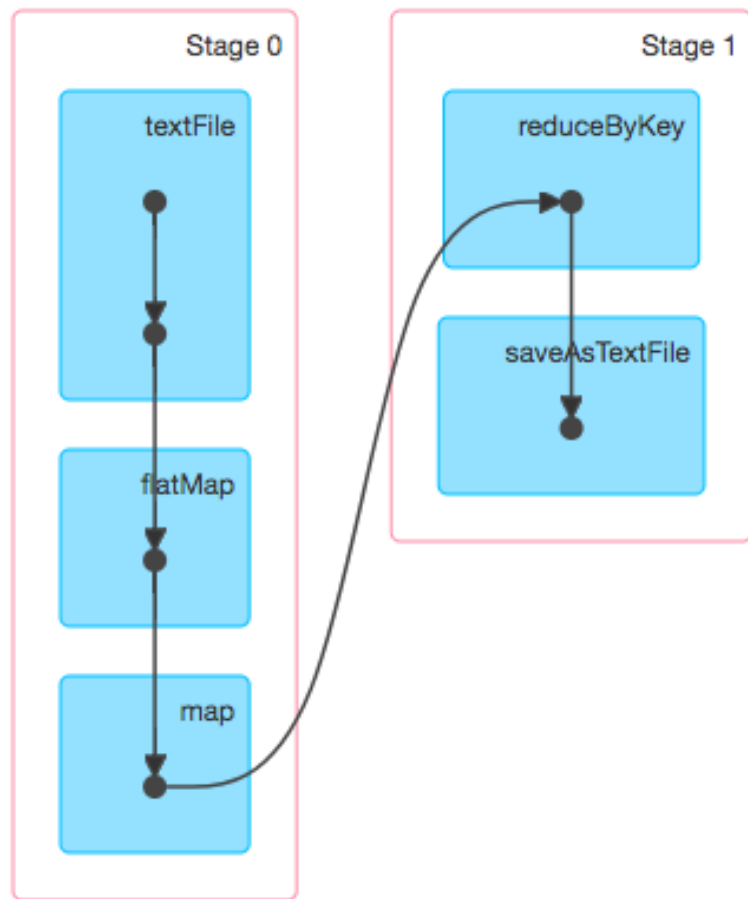
**Transformations**

```
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                    .map(word => (word, 1))
                    .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

**Action**



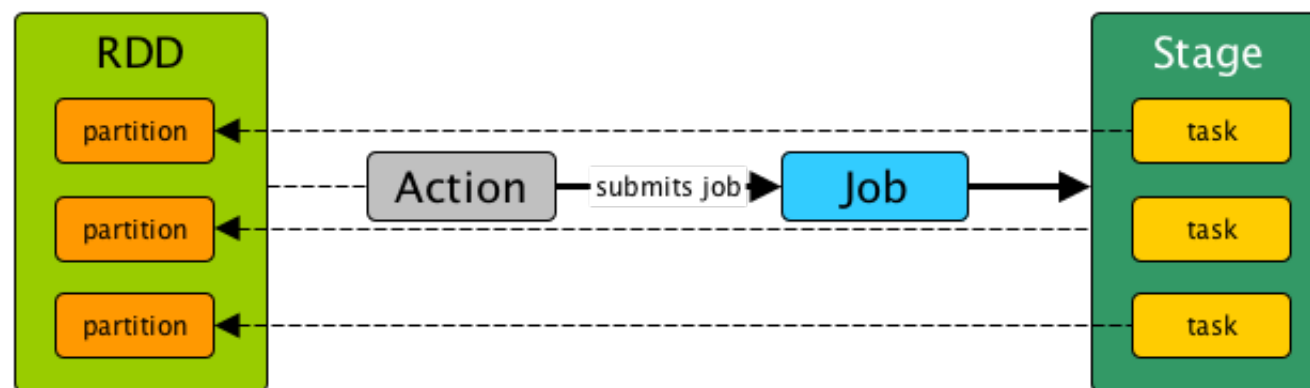
# Stages and tasks



DAG Scheduler

DAG partitioned to **stages**

A stage is a set of parallel **tasks** —  
one task per **partition**



# More reading material..

- <http://hadoop.apache.org/docs/stable/>
- <https://spark.apache.org/docs/latest/>
- White, Tom. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.
- Lin, Jimmy, and Chris Dyer. "Data-intensive text processing with MapReduce." *Synthesis Lectures on Human Language Technologies 3.1* (2010)

