MapReduce

Hadoop

Large data files

- Word count, n-grams.
  - Language corpora.
- Inverted index.
- Reverse web graph.
- Ratings, averages, …

„Ilość przechodzi w jakość.”

The law of the passage of quantitative changes into qualitative changes

F. Engels, Dialektik der Natur

F.W. Hegel „Science of logic”

Parallelisation

- Processing speed
- I/O speed
- Storage capacity

Isues

- Job management
- Load ballancing
  - Input splitting
- Reliability
- Scalability

MapReduce

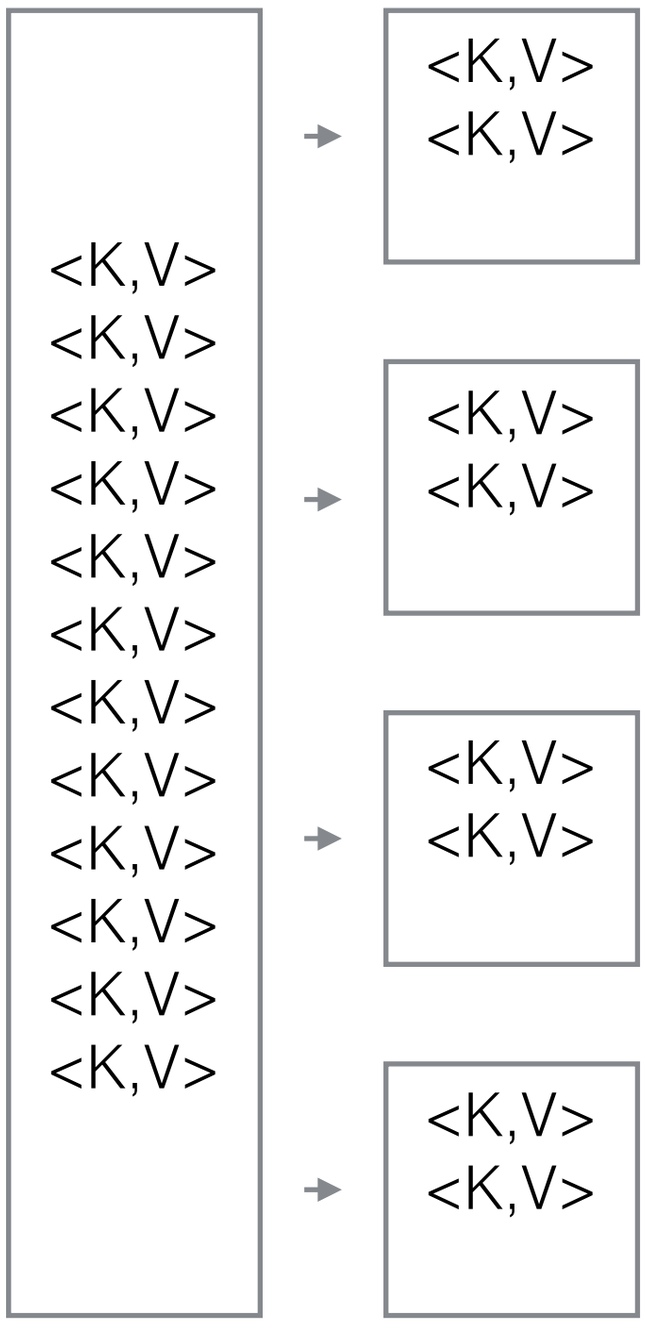
MapReduce: Simplified Data Processing on Large Clusters

J. Dean S. Ghemawat Google, Inc.

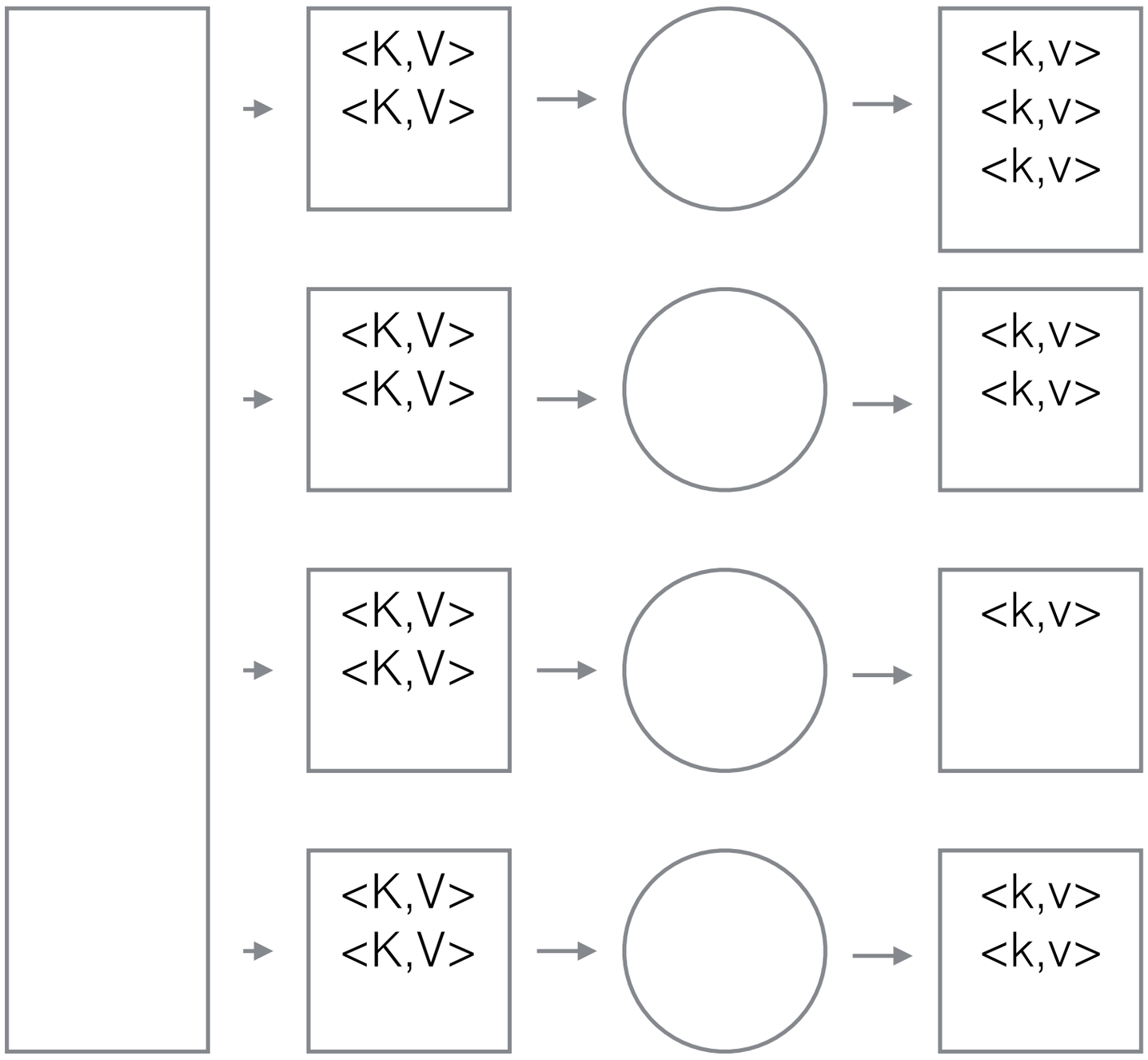
OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<K,V>

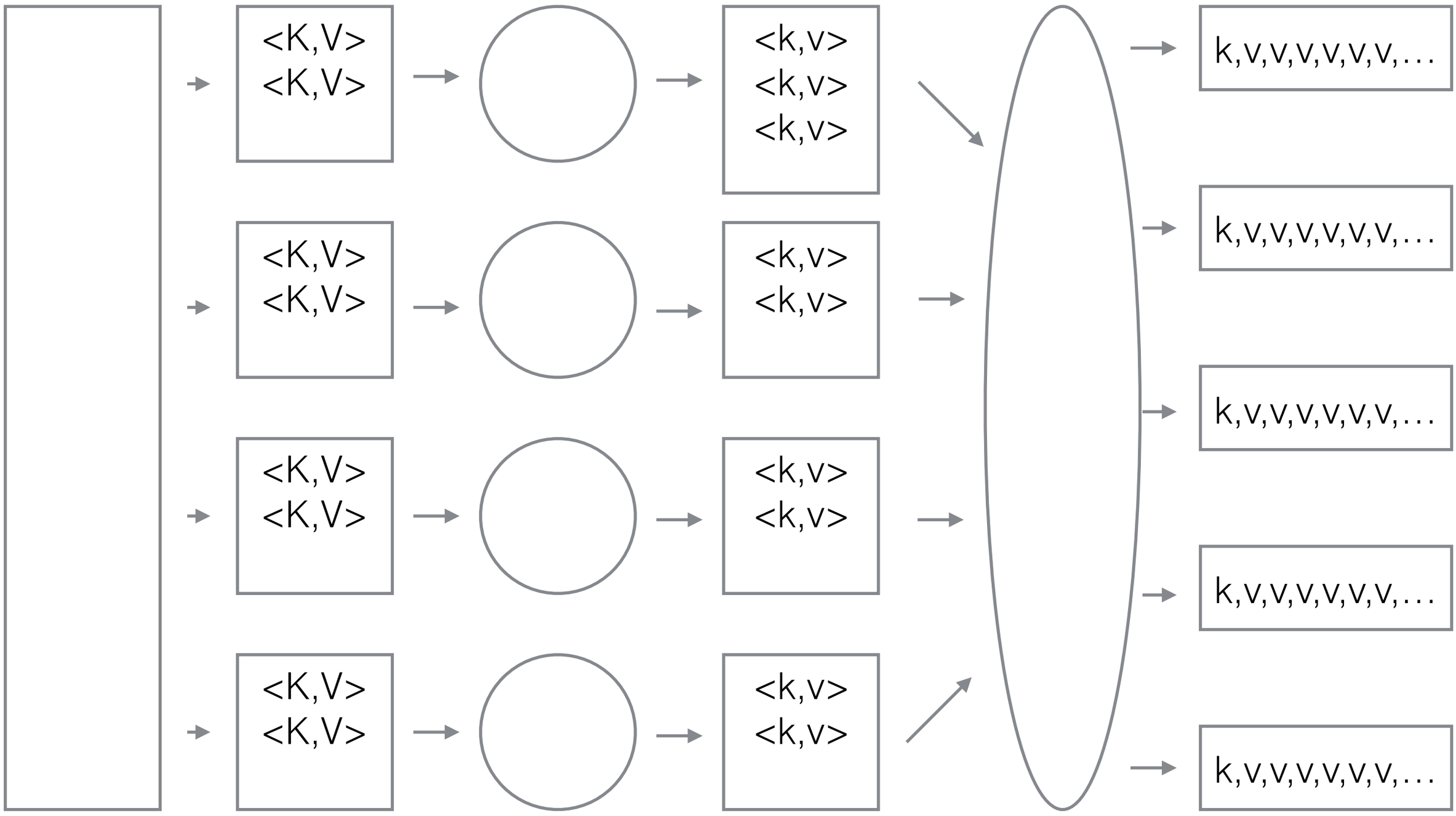
Split



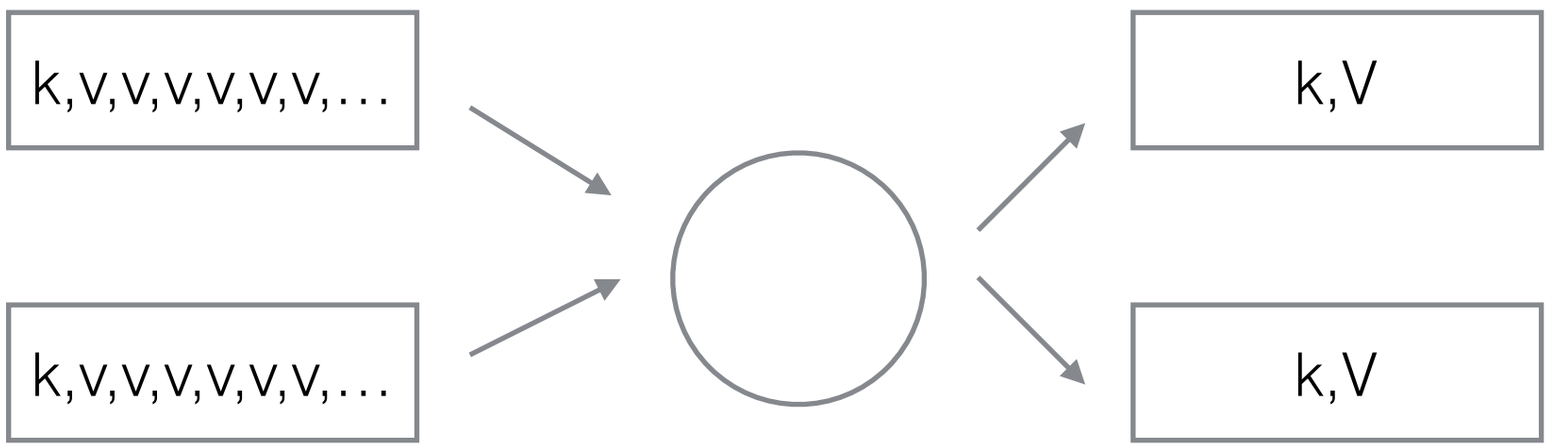
Map

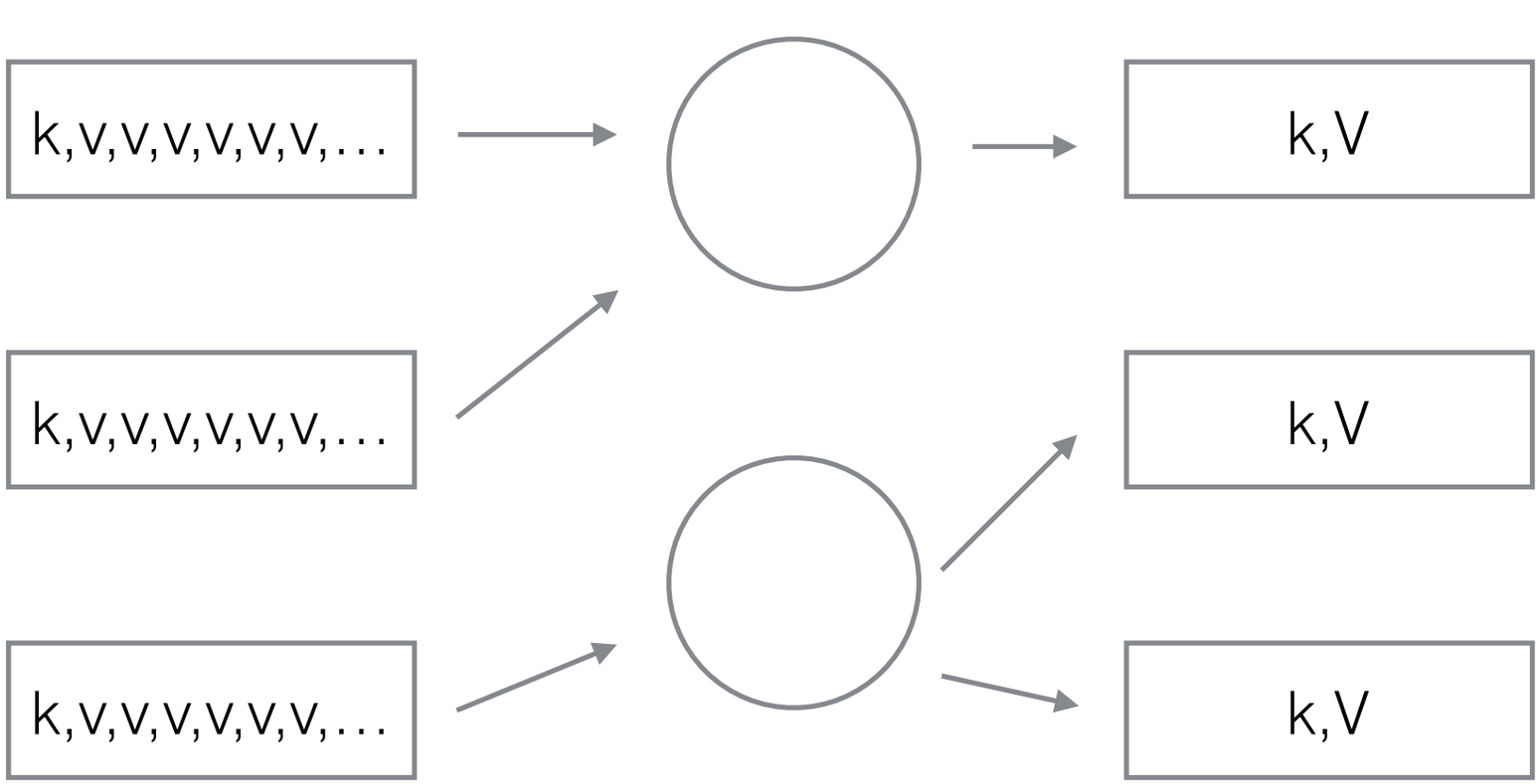


Shuffle


<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<K,V>
<k,v>
<k,v>
<k,v>
<k,v>
<k,v>
<k,v>
k,v,v,v,v,v,v,…
k,v,v,v,v,v,v,…
k,v,v,v,v,v,v,…
k,v,v,v,v,v,v,…

Reduce





Word count

(1, Ala ma kota)
(2, kot ma Alę)

(Ala,1)
(ma,1)
(kota,1)
(kot,1)
(ma,1)
(Alę,1)

(Ala,1)
(ma,1,1)
(kota,1)
(kot,1)
(Alę,1)

(Ala,1)
(ma,2)
(kota,1)
(kot,1)
(Alę,1)

Inverted index

(Document ID, document) ⟶ (word, document ID)

Movie ratings

```
1,2,3.5,2005-04-02 23:53:47
1,29,3.5,2005-04-02 23:31:16
1,32,3.5,2005-04-02 23:33:39
1,47,3.5,2005-04-02 23:32:07
1,50,3.5,2005-04-02 23:29:40
1,112,3.5,2004-09-10 03:09:00
1,151,4,2004-09-10 03:08:54
1,223,4,2005-04-02 23:46:13
1,253,4,2005-04-02 23:35:40
1,260,4,2005-04-02 23:33:46
```

```
2 3.5
29 3.5
32 3.5
47 3.5
50 3.5
112 3.5
151 4
223 4
253 4
260 4
```

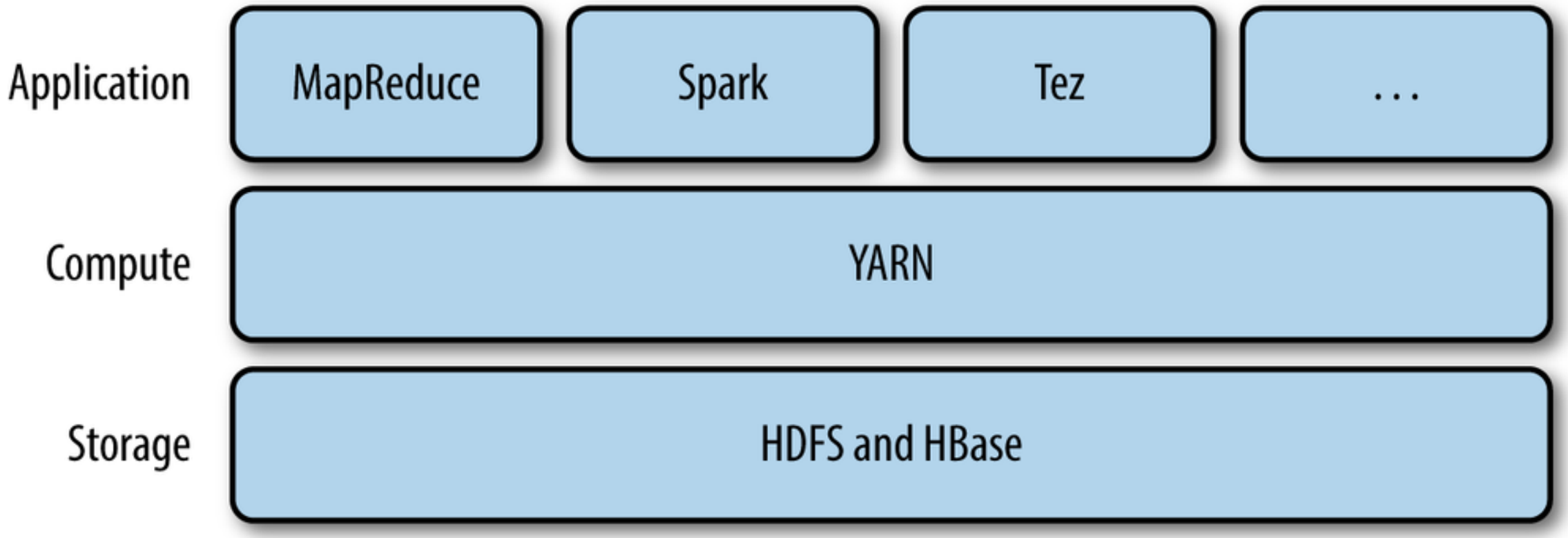
```
47 3.5 3.0 4.0 3.5

223 4 5 3.5 4.5 4
```

https://www.kaggle.com/grouplens/movielens-20m-dataset

Hadoop - Doug Cutting

Hadoop: The Definitive Guide, 4th Edition - O'Reilly Media
Tom White

Examples in book use the NCDC data.


Application
MapReduce
Spark
Tez
...
Compute
YARN
Storage
HDFS and HBase

Mapper

```
class RatingMapper extends Mapper<LongWritable, Text, IntWritable,
FloatWritable> {

    @Override
public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
        String line=value.toString();
        String[] elements=line.split(",");
        int movie = Integer.parseInt(elements[1]);
        float rating = Float.parseFloat(elements[2]);
        context.write(new IntWritable(movie),
                      new FloatWritable(rating));
    }
}
```

Brak obsługi błędów

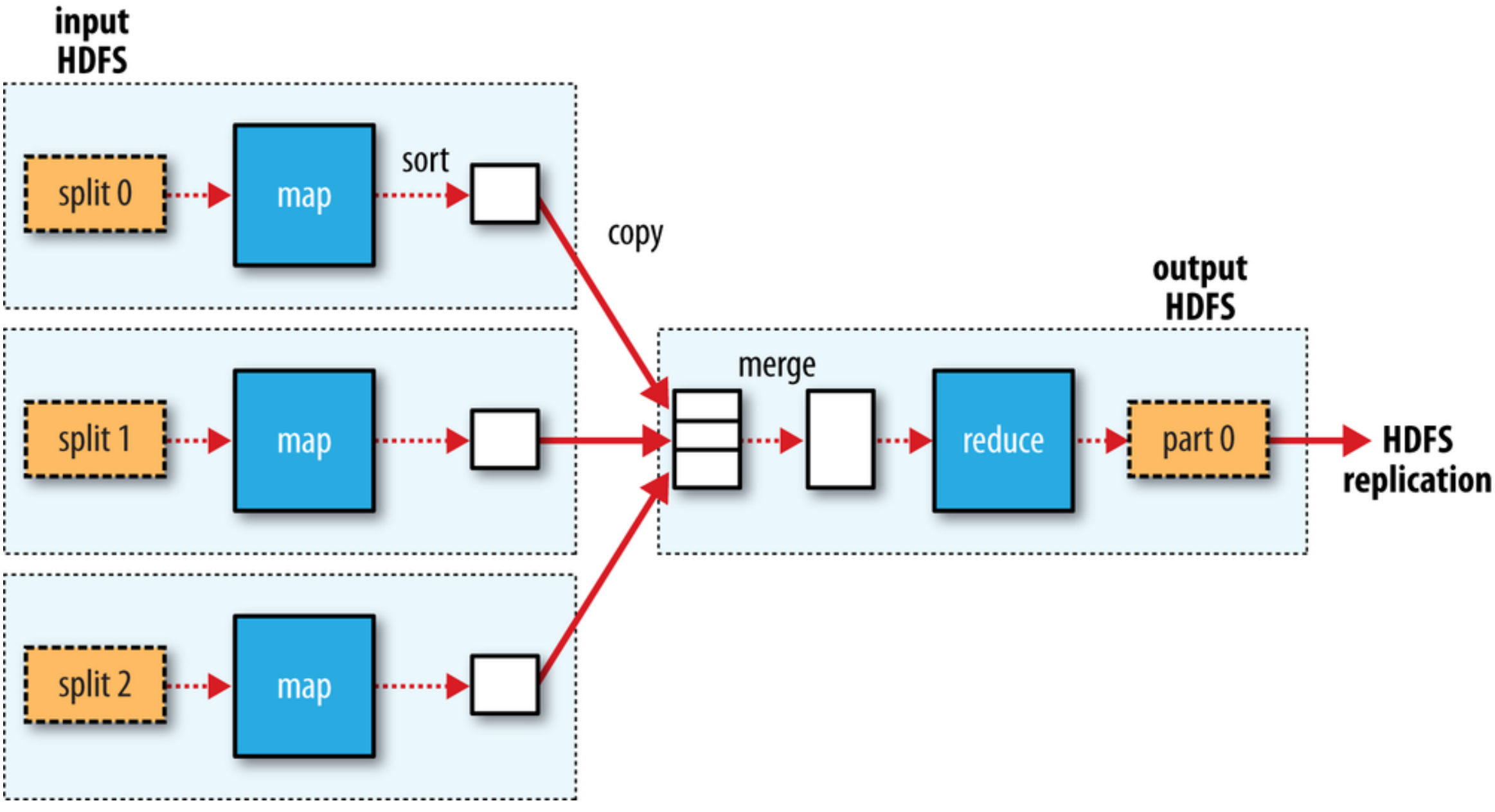
Reducer

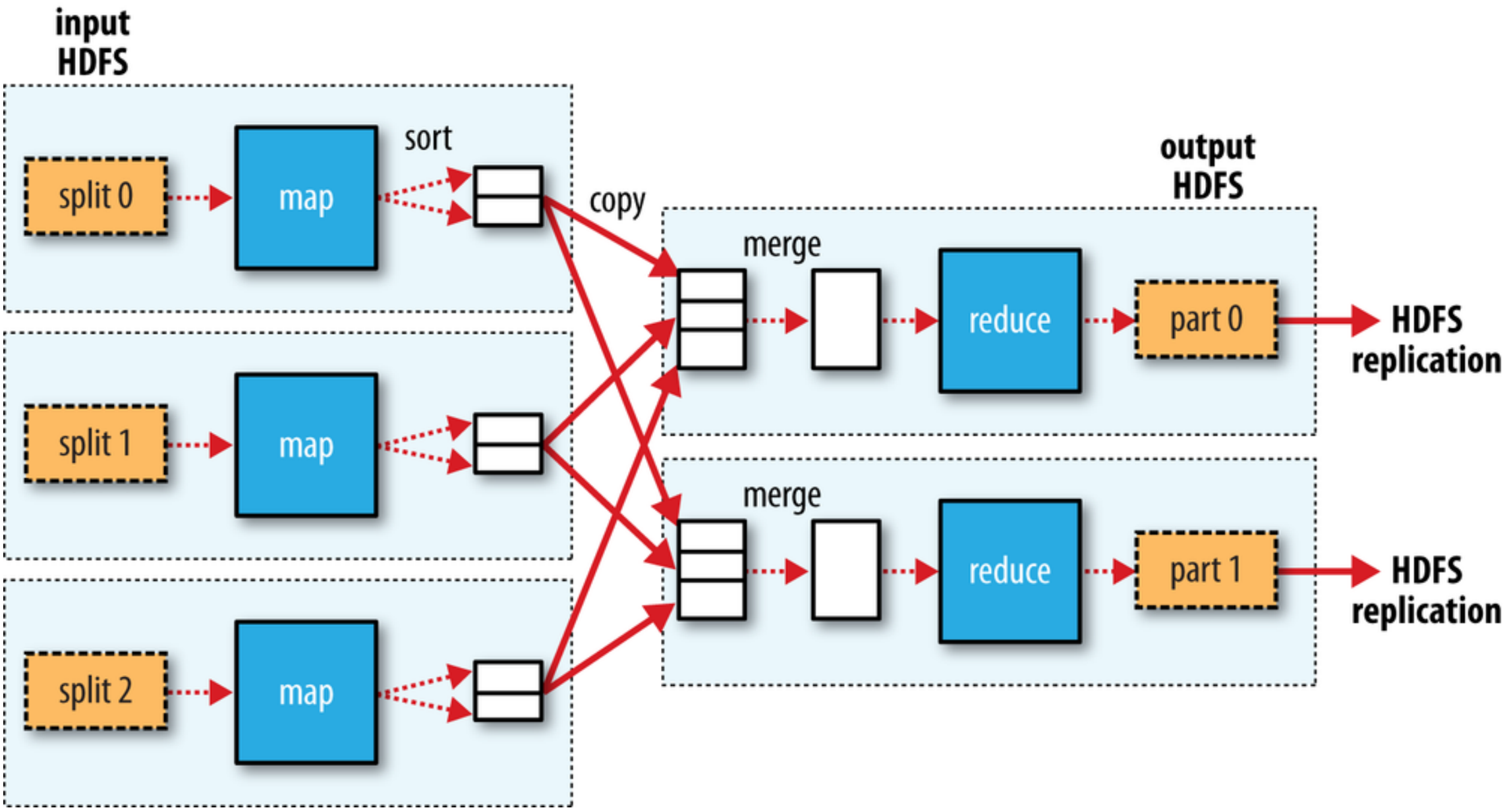
```
class RatingReducer extends Reducer<IntWritable, FloatWritable,
          IntWritable, FloatWritable> {

    @Override
    public void reduce(IntWritable key,
                       Iterable<FloatWritable> values,
                        Context context)
         throws IOException, InterruptedException {
        double sum = 0.0;
        int count = 0;
        for (FloatWritable value : values) {
            sum += value.get();
            count++;
        }
        context.write(key, new FloatWritable((float) (sum/count)));
    }
}
```

Driver

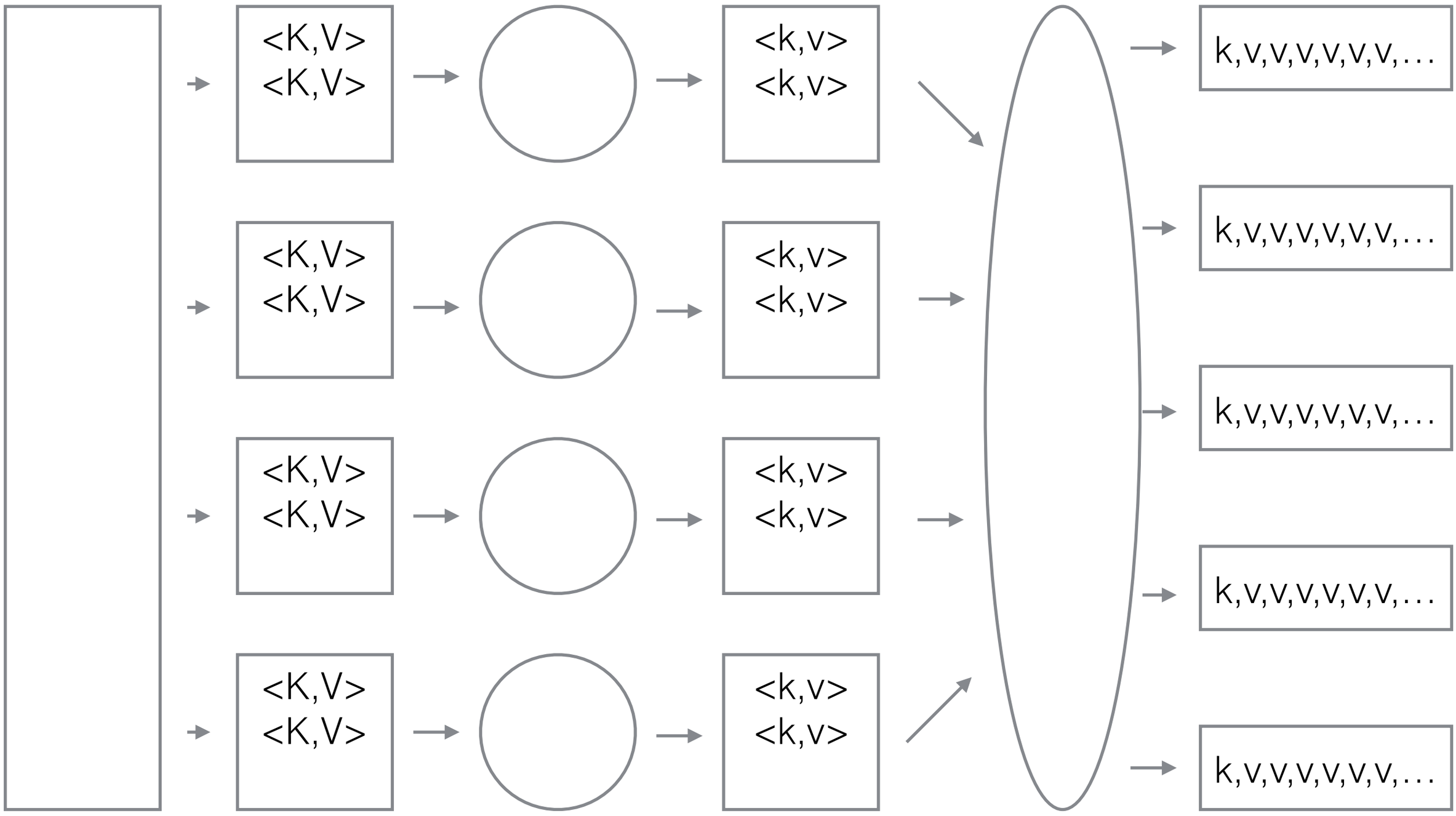
```
public class RatingDriver  extends Configured implements Tool {
    @Override
    public int run(String[] args) throws Exception {

        Job job = new Job(getConf(),"Ratings");
        job.setJarByClass(getClass());

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(RatingMapper.class);
        job.setReducerClass(RatingReducer.class);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(FloatWritable.class);

        return job.waitForCompletion(true) ? 0:1;

}
```

```
16/12/15 11:40:45 INFO input.FileInputFormat: Total input paths to process : 1
16/12/15 11:40:45 INFO mapreduce.JobSubmitter: number of splits:21
16/12/15 11:40:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local50838167_0001
16/12/15 11:40:45 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
16/12/15 11:40:45 INFO mapred.LocalJobRunner: OutputCommitter set in config null
16/12/15 11:40:45 INFO mapreduce.Job: Running job: job_local50838167_0001
16/12/15 11:40:45 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
16/12/15 11:40:45 INFO mapred.LocalJobRunner: OutputCommitter is
org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
16/12/15 11:40:46 INFO mapred.LocalJobRunner: Waiting for map tasks
16/12/15 11:40:46 INFO mapred.LocalJobRunner: Starting task:
attempt_local50838167_0001_m_000000_0
16/12/15 11:40:46 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
16/12/15 11:40:46 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported
only on Linux.
16/12/15 11:40:46 INFO mapred.Task:  Using ResourceCalculatorProcessTree : null
16/12/15 11:40:46 INFO mapred.MapTask:
    Processing split: file:/Users/pbialas/Projects/DataScience/hadoop-movies/ratings/rating.csv:
    0+33554432
16/12/15 11:40:46 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
16/12/15 11:40:46 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
16/12/15 11:40:46 INFO mapred.MapTask: soft limit at 83886080
16/12/15 11:40:46 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
16/12/15 11:40:46 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
16/12/15 11:40:46 INFO mapred.MapTask: Map output collector class =
org.apache.hadoop.mapred.MapTask$MapOutputBuffer
16/12/15 11:40:46 INFO mapred.LocalJobRunner:
16/12/15 11:40:46 INFO mapred.MapTask: Starting flush of map output
16/12/15 11:40:46 INFO mapred.MapTask: Spilling map output
16/12/15 11:40:46 INFO mapred.MapTask: bufstart = 0; bufend = 8098632; bufvoid = 104857600
16/12/15 11:40:46 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 22165084(88660336);
length = 4049313/6553600
16/12/15 11:40:46 INFO mapreduce.Job: Job job_local50838167_0001 running in uber mode : false
```

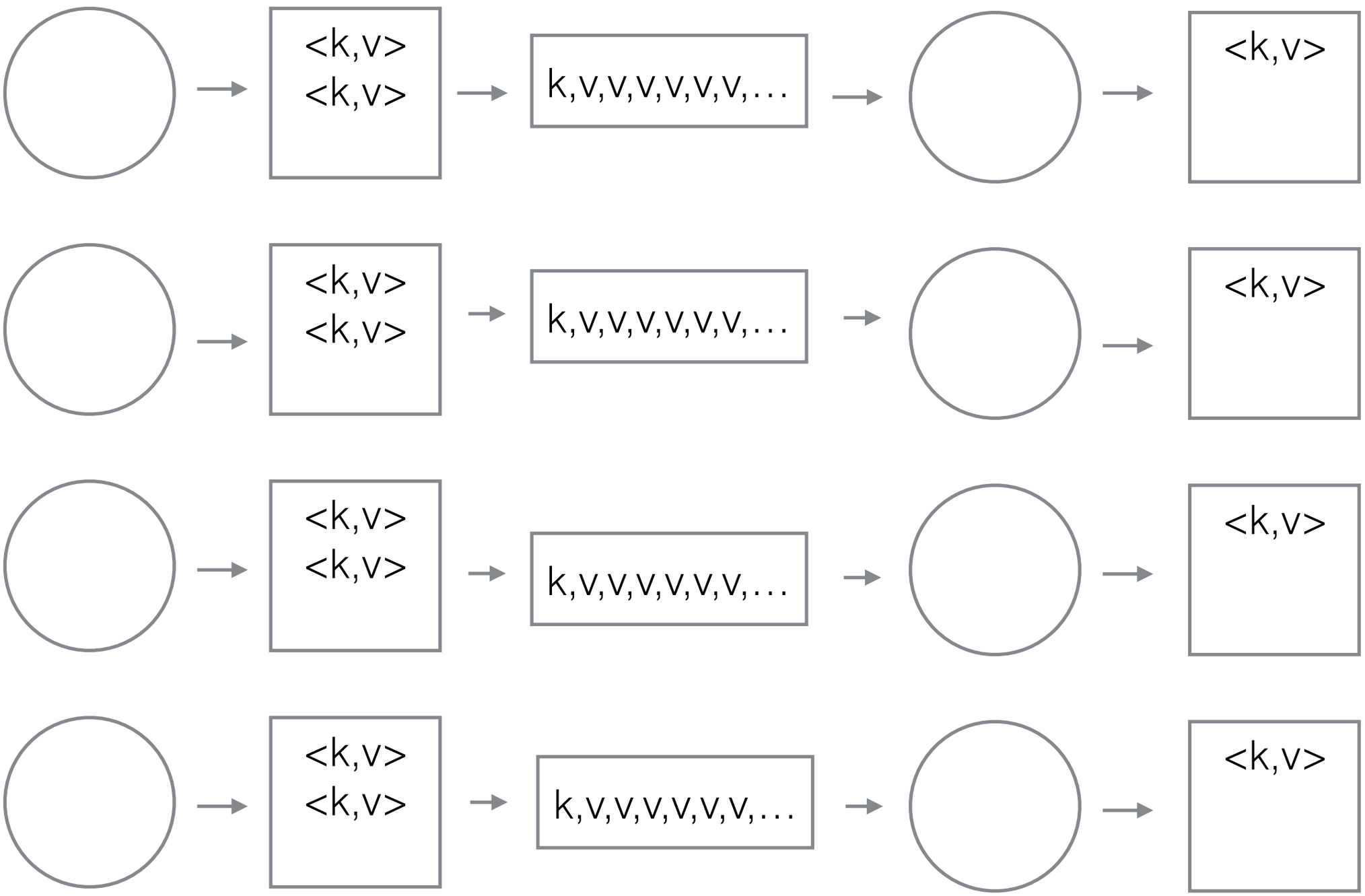

input
HDFS
split 0
map
sort
split 1
map
split 2
map
copy
output
HDFS
merge
reduce
part 0
HDFS
replication


input
HDFS
split 0
map
sort
split 1
map
split 2
map
copy
output
HDFS
merge
reduce
part 0
HDFS
replication
merge
reduce
part 1
HDFS
replication

Combiners


<K,V>
<K,V>
<k,v>
<k,v>
<K,V>
<K,V>
<k,v>
<k,v>
<K,V>
<K,V>
<k,v>
<k,v>
<K,V>
<K,V>
<k,v>
<k,v>
k,v,v,v,v,v,v,…
k,v,v,v,v,v,v,…
k,v,v,v,v,v,v,…
k,v,v,v,v,v,v,…

Combiners



Combiners

```
class RatingCountCombiner extends Reducer<IntWritable,
FloatIntWritable, IntWritable, FloatIntWritable> {

    @Override
    public void reduce(IntWritable key,
        Iterable<FloatIntWritable> values, Context context)
        throws IOException, InterruptedException {
          double sum = 0.0;
          int count = 0;
          for (FloatIntWritable value : values) {
             sum += value.getF();
             count+=value.getI();
          }
          context.write(key,
                  new FloatIntWritable((float) (sum), count));
    }
}
```

Combiners

```
class RatingCountReducer extends Reducer<IntWritable,
FloatIntWritable, IntWritable, FloatIntWritable> {

    @Override
    public void reduce(IntWritable key,
                       Iterable<FloatIntWritable> values,
                       Context context)
                       throws IOException, InterruptedException {
        double sum = 0.0;
        int count = 0;
        for (FloatIntWritable value : values) {
            sum += value.getF();
            count+=value.getI();
        }
        context.write(key, new FloatIntWritable((float) (sum/count),
                                                count));
    }
}
```

Other frameworks

- Hive (SQL)
- Pig
- Execution engines
  - MapReduce
  - Spark
  - Tez
- Real time
  - Storm

Spark

- Resilient Distributed Dataset (RDD)
- Transformations
- Actions

pyspark

```
from pyspark import SparkContext
import re, sys

sc = SparkContext("local","Movie Ratings");
sc.textFile(sys.argv[1])\
    .map(lambda s: s.split(","))\
    .map(lambda r: (int(r[1]), float(r[2])))\
    .mapValues(lambda v: (v,1))\
.reduceByKey(lambda a,b: (a[0]+b[0], a[1]+b[1]))\
.saveAsTextFile(sys.argv[2])
```

Project

- We should provide students with small Hadoop cluster.
- Give them a data set that is impossible/hard to process on a single computer.
  - e.g. NCDC climate data
- Combine with visualization (e.g. D3.js)