Hadoop - Mapreduce Programming

عادل قاضی خانی
هیئت علمی گروه کامپیوتر دانشگاه امام رضا (ع)

تعریف

- بستر **پردازش و تحلیل داده ها** در هدوپ

موازی فکر کردن

- برای کار با Mapreduce **اولین مسئله** اینستکه باید **بصورت موازی به مسئله فکر کنیم**

- **مثال** شمارش تعداد پرواز برای هر شرکت هواپیمایی در مجموعه داده های پروازها

مثال

- نوشتن برنامه بصورت سری (serial)

```
create a two-dimensional array
  create a row for every airline carrier
    populate the first column with the carrier code
    populate the second column with the integer zero

for each line of flight data
  read the airline carrier code
  find the row in the array that matches the carrier code
    increment the counter in the second column by one

print the totals for each row in the two-dimensional array
```

مثال

- نوشتن برنامه بصورت موازی

```
Map Phase:
  for each line of flight data
    read the current record and extract the airline carrier code
    output the airline carrier code and the number one as a key/value pair

Shuffle and Sort Phase:
  read the list of key/value pairs from the map phase
  group all the values for each key together
    each key has a corresponding array of values
  sort the data by key
  output each key and its array of values

Reduce Phase:
  read the list of carriers and arrays of values from the shuffle and sort phase
  for each carrier code
    add the total number of ones in the carrier code's array of values together

print the totals for each row in the two-dimensional array
```

Mapreduce

- Mapreduce **یک مدل برنامه نویسی** برای **داده هایی که در خوشه** توزیع شده اند

- **تقسیم یک داده حجیم** به بخشهایی و **اجرای الگوریتم واحد بر روی آنها**

مراحل Mapreduce

- تعیین **مجموعه داده هایی** که باید **پردازش** شوند (تعیین مکان رکوردها در بلوکها)
- **اجرای الگوریتم** بر روی **تمام رکوردها**.
- **نمونه برنامه ای** که برای هر بلوک اجرا می شود **عمل نگاشت** است (Map)
- اجرای یک **مرحله میانی کاهش** که اختیاری است (shuffle)

مراحل Mapreduce ادامه

- با **توجه به خروجی** مورد نظر، **نتایجی که باید باهم ادغام** شوند مشخص شوند

- نتایج با هم **ادغام** شوند (**عمل کاهش**). نمونه برنامه ای که بر روی نتایج نگاشت اجرا می شود عمل کاهش است

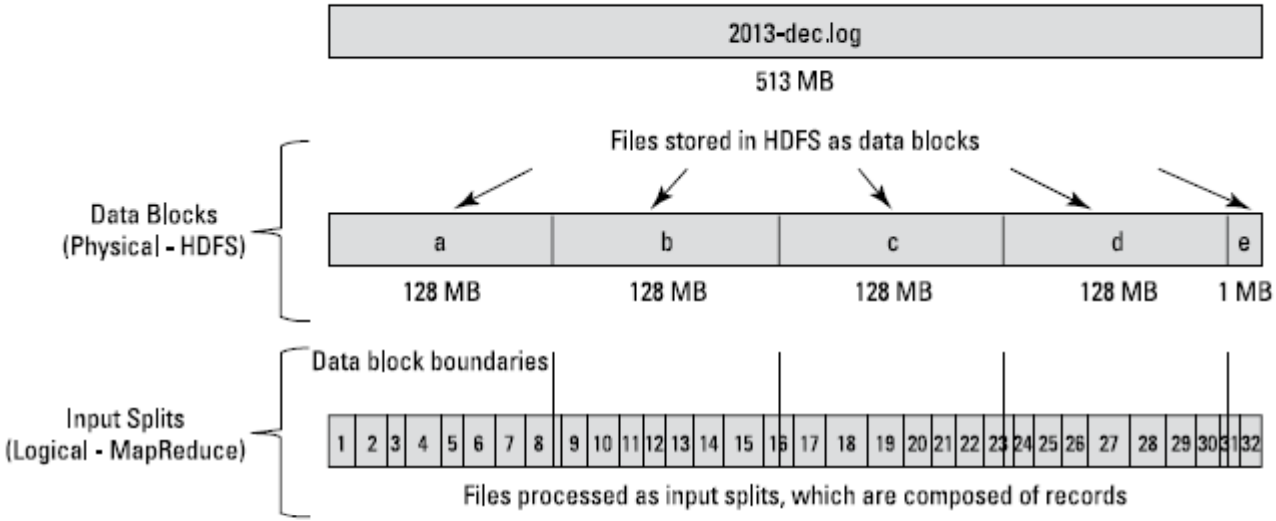
رکوردها در بلوکهای داده

- در HDFS داده ها به بلوکهایی تقسیم می شوند

- **Mapreduce** در **هر بلوک داده تمام رکوردها** را پردازش می کند. ممکن است بخشی از یک رکورد در بلوکهای دیگر باشد که در اینحالت اطلاعات بلوک بعدی باید وجود داشته باشد

رکوردها در بلوکهای داده

- **Mapreduce** در **هر بلوک داده تمام رکوردها** را پردازش می کند. ممکن است بخشی از یک رکورد در بلوکهای دیگر باشد که در اینحالت اطلاعات بلوک بعدی باید وجود داشته باشد



رکوردها در Mapreduce

- **Mapreduce** رکوردها را بصورت **کلید مقدار** دریافت می کند
- بعضی مواقع ممکن است از یک **کلید مقدار صوری** استفاده کنیم

مثال Mapreduce

- **مثال** شمارش تعداد پرواز برای هر شرکت هواپیمایی در مجموعه داده های پروازها

- فایل ورودی شامل پروازها

Input file

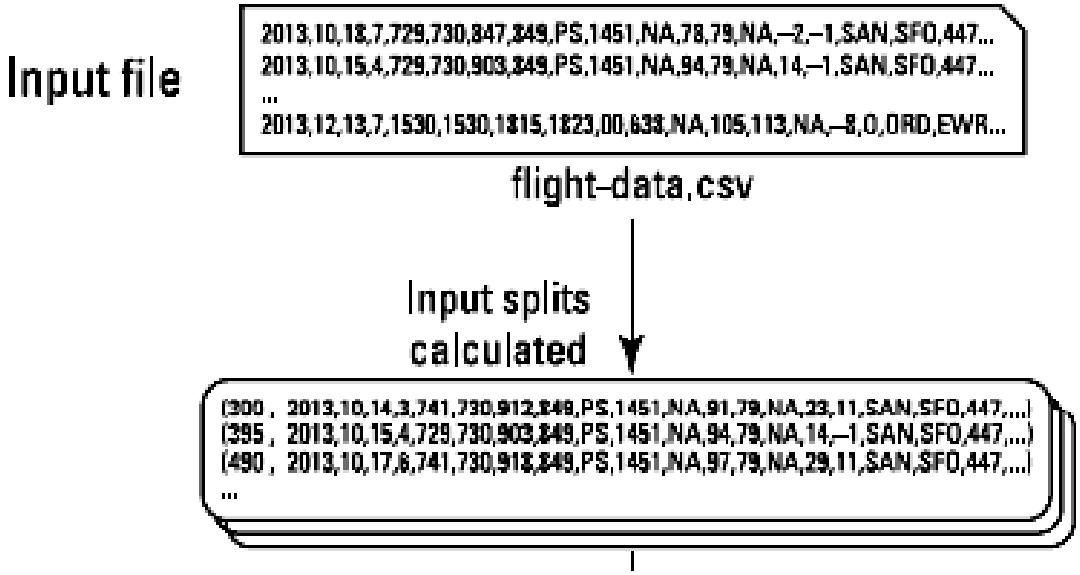
```
2013,10,18,7,729,730,847,849,PS,1451,NA,78,79,NA,-2,-1,SAN,SFO,447...
2013,10,15,4,729,730,903,849,PS,1451,NA,94,79,NA,14,-1,SAN,SFO,447...
...
2013,12,13,7,1530,1530,1815,1823,00,633,NA,105,113,NA,-8,0,ORD,EWR...
```

flight-data.csv

مثال Mapreduce

- **مثال** شمارش تعداد پرواز برای هر شرکت هواپیمایی در مجموعه داده های پروازها

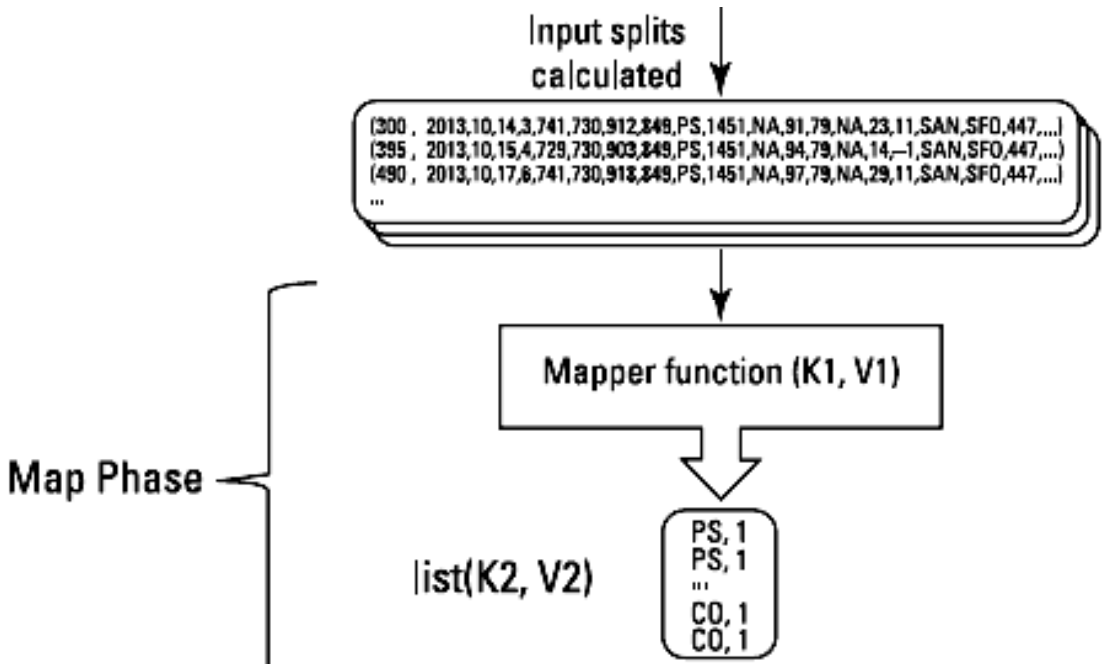
- تقسیم بندی فایل ورودی



مثال Mapreduce

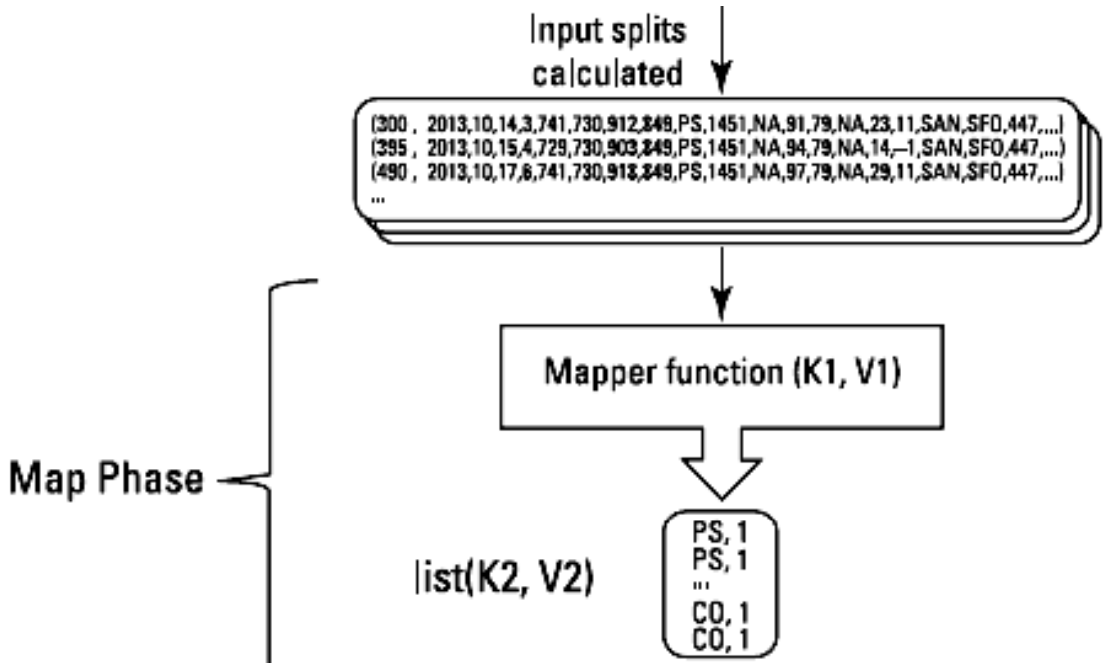
- **مثال** شمارش تعداد پرواز برای هر شرکت هواپیمایی در مجموعه داده های پروازها

- فاز نگاشت



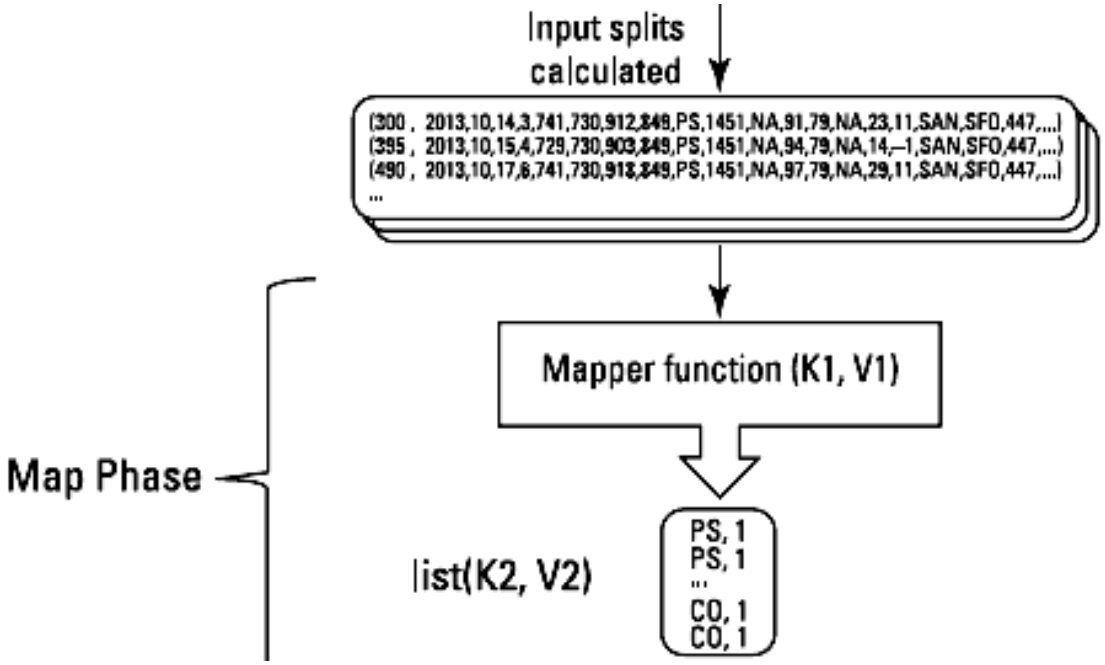
مثال Mapreduce

- **فاز نگاشت :** ورودی این قسمت (K1,V1) است.
- K1 ، offset هر رکورد از ابتدای فایل بعنوان **کلید**
- V1 ، خود رکورد بعنوان **مقدار**



مثال Mapreduce

- **فاز نگاشت :**
- خروجی این قسمت (K2,V2) است.
- متفاوت از ورودی



مثال Mapreduce

- **فاز نگاشت :**

- در این مثال یک کاهش دهنده وجود دارد بنابراین هر نگاشت کننده نیز یک فایل خروجی دارد

- ممکن است هر نگاشت کننده بیش از یک فایل خروجی داشته باشد

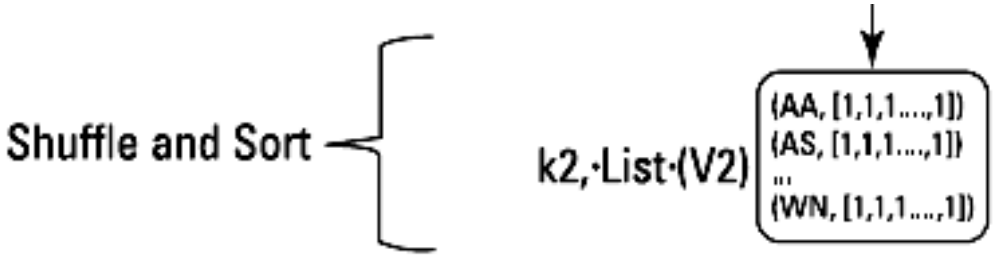
- خروجی نگاشت کننده را فشرده کنید
- mapreduce.map.output.compress=true

مثال Mapreduce

- **فاز مخلوط کردن و مرتب سازی :**

- نتایج مرحله نگاشت، بر اساس کلید مرتب، تقسیم بندی می شوند و به دیسک منتقل می شوند (روی خوشه توزیع نمی شود)

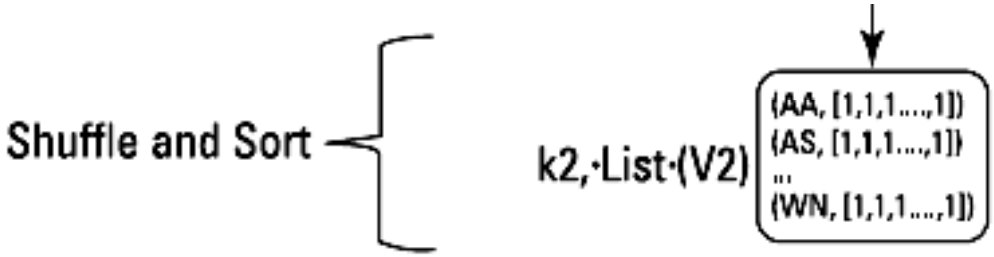
- داده ها به نودهای کاهش منتقل می شوند



مثال Mapreduce

- **فاز مخلوط کردن و مرتب سازی :**

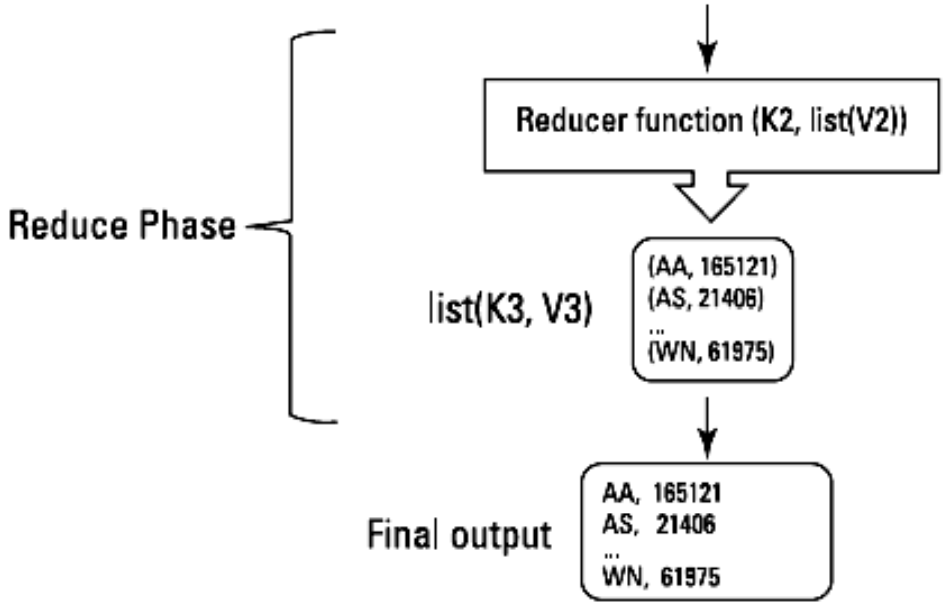
- اگر اجرای عملیات در یک گره slave کند باشد، اعمال دوباره در گره دیگر اجرا می شوند



مثال Mapreduce

- **فاز کاهش :**

- در فاز کاهش نتایج میان با هم ترکیب می شوند

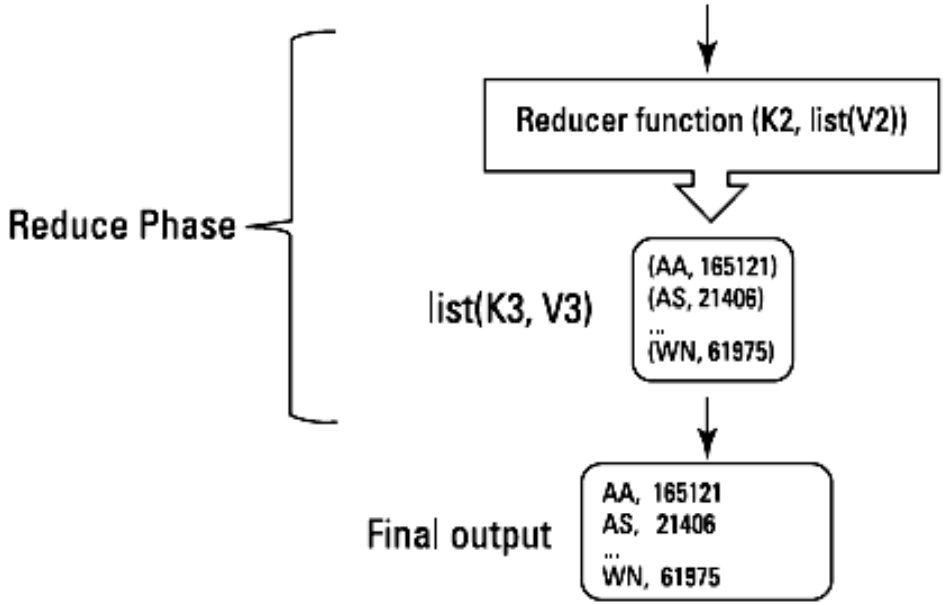


map: (K1, V1) → list(K2, V2)
reduce: (K2, list(V2)) → list(K3, V3)

مثال Mapreduce

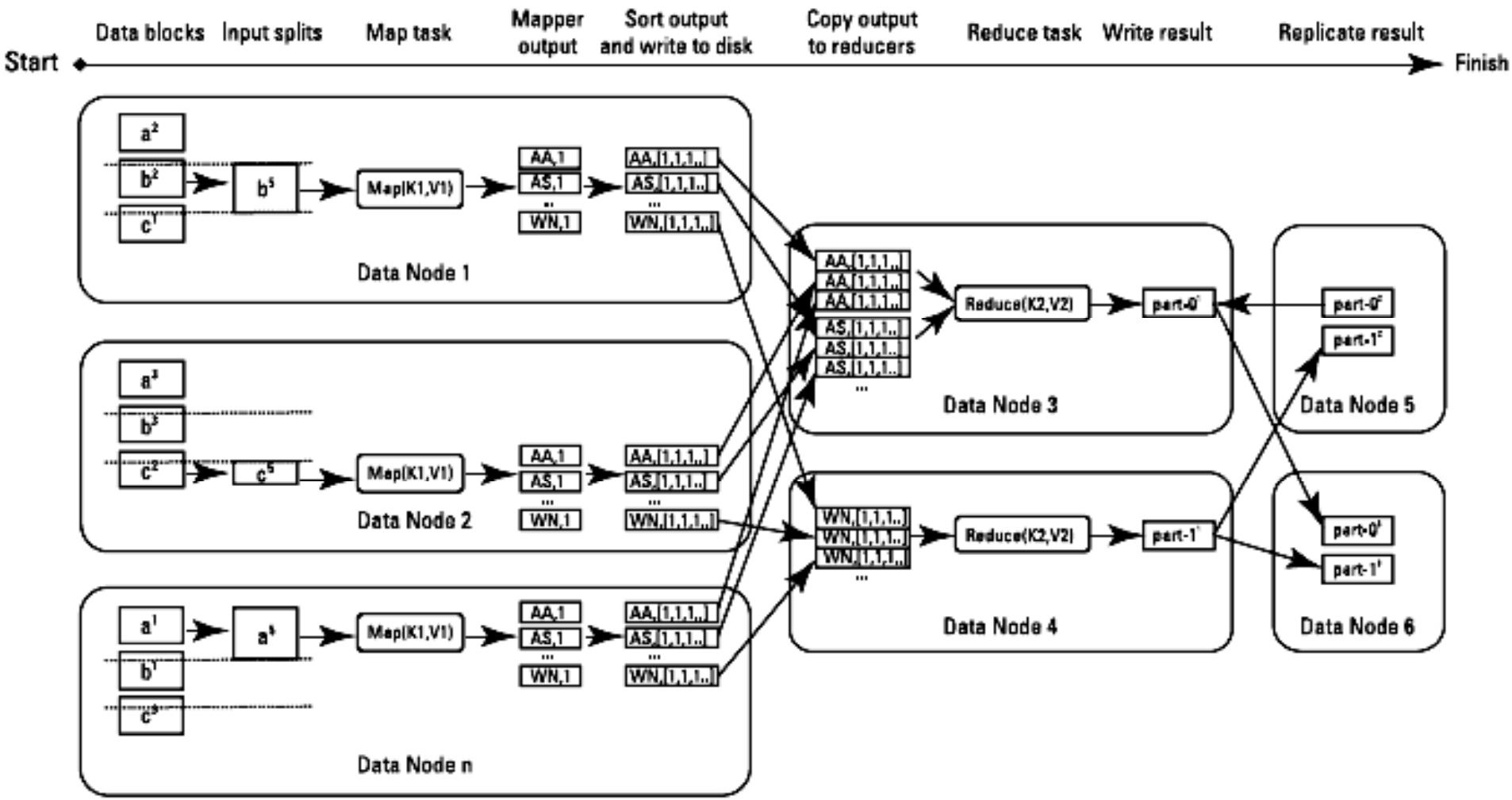
- **فاز کاهش :**

- هر کلید بصورت مجزا پردازش می شود



map: (K1, V1) → list(K2, V2)
reduce: (K2, list(V2)) → list(K3, V3)

مراحل Mapreduce



ساختار برنامه

- برای نوشتن یک برنامه Mapreduce می توان چندین بخش طراحی کرد

- راه انداز (Driver) - ضروری
- کلاس Mapper - ضروری
- کلاس Reducer - اختیاری
- کلاس Combiner - اختیاری
- کلاس Partitioner - اختیاری
- کلاس RecordReader و RecordWriter - اختیاری

ساختار برنامه

- **راه انداز:** بخشی از برنامه که توسط client فراخوانی می شود. کلاس عمل Mapreduce را تنظیم می کند

ساختار برنامه

- **کلاس Mapper:** فرمت کلید – مقدار ورودی و خروجی بخش نگاشت را مشخص می کند. یک متد بنام map دارد که نحوه پردازش رکوردها در آن مشخص می شود. خروجی ها به نمونه ای از کلاس Context نوشته می شوند

ساختار برنامه

- **کلاس Reducer:** عمل کاهش را مشخص می کند
- **کلاس Combiner:** برای افزایش کارایی می توان بخشی از کاهش را در این کلاس نوشت
- **کلاس Partitioner:** برای تعریف عملیات خاص مثل مرتب سازی ثانویه استفاده می شود
- **کلاس Recordreader و writer:** برای خواندن و نوشتن داده هایی که فرمت خاصی دارند که در استاندارد Hadoop نیست

مثال شرکت هواپیمایی – بخش راه انداز

```
@@1
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class FlightsByCarrier {
       public static void main(String[] args) throws Exception {
              @@2
              Job job = new Job();
              job.setJarByClass(FlightsByCarrier.class);
              job.setJobName("FlightsByCarrier");

              @@3
              TextInputFormat.addInputPath(job, new Path(args[0]));
              job.setInputFormatClass(TextInputFormat.class);

              @@4
              job.setMapperClass(FlightsByCarrierMapper.class);
              job.setReducerClass(FlightsByCarrierReducer.class);

              @@5
              TextOutputFormat.setOutputPath(job, new Path(args[1]));
              job.setOutputFormatClass(TextOutputFormat.class);
              job.setOutputKeyClass(Text.class);
              job.setOutputValueClass(IntWritable.class);
              @@6
              job.waitForCompletion(true);
        }
}
```

مثال شرکت هواپیمایی – کلاس Mapper

```
@@1
import java.io.IOException;
import au.com.bytecode.opencsv.CSVParser;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

@@2
public class FlightsByCarrierMapper extends
                                Mapper<LongWritable, Text, Text, IntWritable> {
        @Override
        @@3
        protected void map(LongWritable key, Text value, Context context)
                        throws IOException, InterruptedException {
                @@4
                if (key.get() > 0) {
                        String[] lines = new
                CSVParser().parseLine(value.toString());
                        @@5
                        context.write(new Text(lines[8]), new IntWritable(1));
                }
        }
}
```

مثال شرکت هواپیمایی – کلاس Reducer

```
@@1
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

@@2
public class FlightsByCarrierReducer extends
              Reducer<Text, IntWritable, Text, IntWritable> {
       @Override
       @@3
       protected void reduce(Text token, Iterable<IntWritable> counts,
                     Context context) throws IOException, InterruptedException {
              int sum = 0;

              @@4
              for (IntWritable count : counts) {
                     sum+= count.get();
              }
              @@5
              context.write(token, new IntWritable(sum));
        }
}
```

مثال – اجرای برنامه

1-Compile:

- javac -classpath $CLASSPATH MapRed/FlightsByCarrier/*.java

2- Make JAR file

- jar cvf FlightsByCarrier.jar *.class

3- Run Driver

- hadoop jar FlightsByCarrier.jar FlightsByCarrier /user/root/airlinedata/2008.csv /user/root/output/flightsCount