# DATA PROCESSING IN HADOOP

Prof. Aparna Halbe

aparna_halbe@spit..ac.in

Department of Information Technology

Sardar Patel Institute of Technology, Mumbai

October 28, 2020

# Online match analysis using Hadoop map reduce

Consider a big data set of online game statistics.

Data set link
https://www.kaggle.com/chuckephron/leagueoflegends#matchinfo.csv

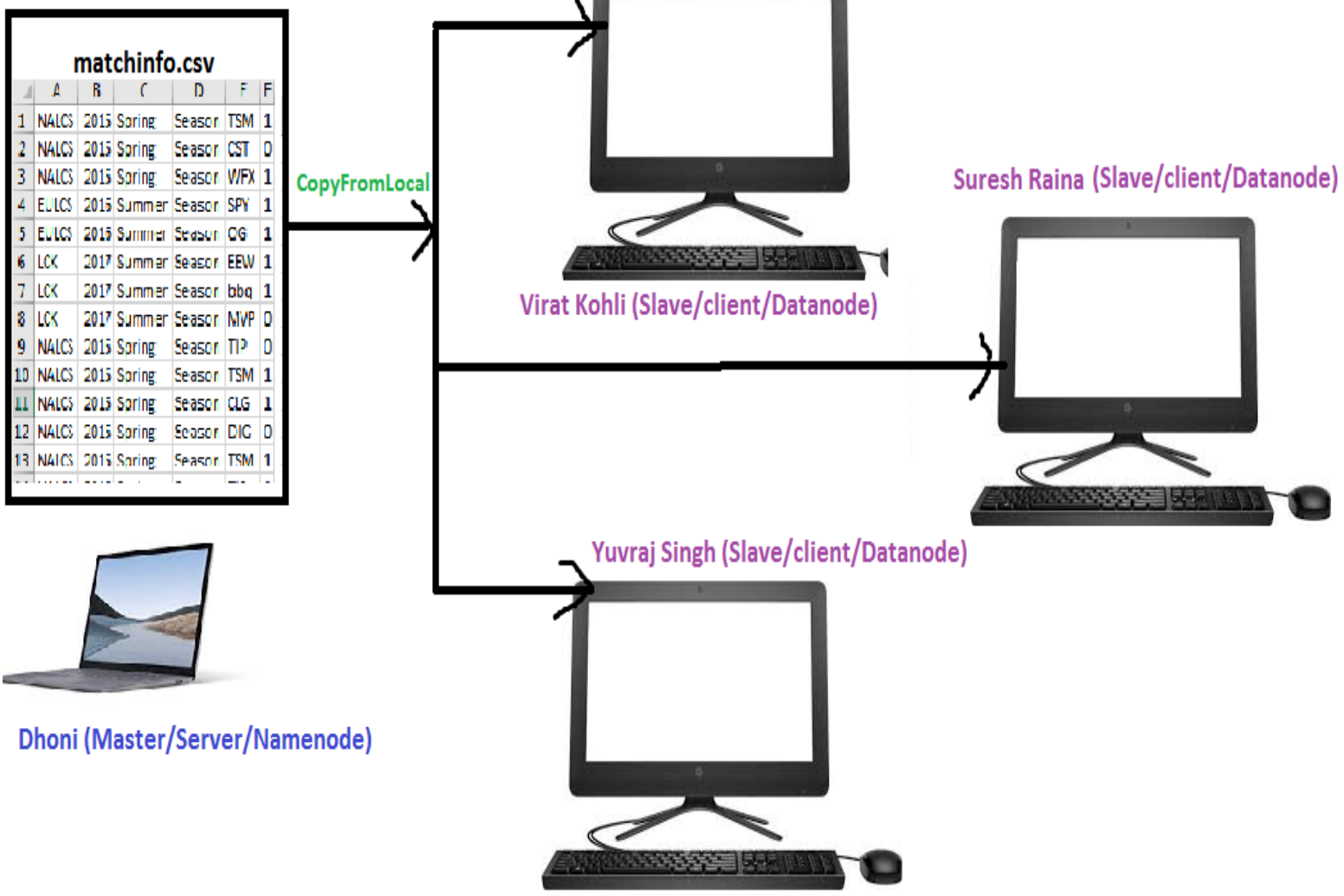
Write Hadoop map reduce code to perform following analytics
on this data set.

1. Find total no of matches played in a league.

2. Find total number of matches played in a year.

3. Find the maxgame length of tournament played between team1 vs team2.

4. If a tournament is played between team1 vs team2.
   Find how many times team1 has won and
   how many times team2 has won.


matchinfo.csv
CopyFromLocal
Virat Kohli (Slave/client/Datanode)
Suresh Raina (Slave/client/Datanode)
Yuvraj Singh (Slave/client/Datanode)
Dhoni (Master/Server/Namenode)

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | KOHLI | | NALCS | 2015 | Spring | Season | TSM | 1 | 0 | C9 | 40 |
| 2 | | | NALCS | 2015 | Spring | Season | CST | 0 | 1 | DIG | 38 |
| 3 | | | NALCS | 2015 | Spring | Season | WFX | 1 | 0 | GV | 40 |
| 4 | | | EULCS | 2016 | Summer | Season | SPY | 1 | 0 | OG | 41 |
| 5 | | | EULCS | 2016 | Summer | Season | OG | 1 | 0 | SPY | 43 |
| 6 | | | LCK | 2017 | Summer | Season | EEW | 1 | 0 | bbq | 42 |
| 7 | | | LCK | 2017 | Summer | Season | bbq | 1 | 0 | EEW | 29 |
| 8 | | | LCK | 2017 | Summer | Season | MVP | 0 | 1 | SKT | 39 |
| 9 | YUVRAJ | | NALCS | 2015 | Spring | Season | TIP | 0 | 1 | CLG | 31 |
| 10 | | | NALCS | 2015 | Spring | Season | TSM | 1 | 0 | WFX | 33 |
| 11 | | | NALCS | 2015 | Spring | Season | CLG | 1 | 0 | C9 | 38 |
| 12 | | | NALCS | 2015 | Spring | Season | DIG | 0 | 1 | WFX | 40 |
| 13 | | | NALCS | 2015 | Spring | Season | TSM | 1 | 0 | TL | 39 |
| 14 | | | NALCS | 2015 | Spring | Season | TIP | 1 | 0 | T8 | 40 |
| 15 | | | NALCS | 2015 | Spring | Season | CST | 0 | 1 | GV | 39 |
| 16 | | | NALCS | 2015 | Spring | Season | GV | 0 | 1 | TIP | 49 |
| 17 | | | LLN | 2017 | Spring | Season | JTH | 0 | 1 | LYN | 33 |
| 18 | | | LLN | 2017 | Spring | Season | LYN | 1 | 0 | JTH | 26 |
| 19 | | | LLN | 2017 | Spring | Season | D9 | 0 | 1 | INF | 31 |
| 20 | | | LLN | 2017 | Spring | Season | INF | 1 | 0 | D9 | 34 |
| 21 | | | LLN | 2017 | Spring | Season | ZTG | 0 | 1 | GG | 39 |
| 22 | RAINA | | NALCS | 2016 | Spring | Season | CLG | 0 | 1 | IMT | 33 |
| 23 | | | NALCS | 2016 | Spring | Season | TSM | 1 | 0 | FOX | 39 |
| 24 | | | LCK | 2016 | Summer | Playoffs | ROX | 1 | 0 | kt | 34 |
| 25 | | | LCK | 2016 | Summer | Playoffs | kt | 1 | 0 | ROX | 38 |
| 26 | | | LLN | 2017 | Summer | Season | INF | 1 | 0 | GG | 33 |

| | | | | | | | | | | | Mapper Output |
|---|---|---|---|---|---|---|---|---|---|---|---|
| KOHLI | NALCS | 2015 | Spring | Season | TSM | 1 | 0 | C9 | 40 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | CST | 0 | 1 | DIG | 38 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | WFX | 1 | 0 | GV | 40 | (NALCS,1) |
| | EULCS | 2016 | Summer | Season | SPY | 1 | 0 | OG | 41 | (EULCS,1) |
| | EULCS | 2016 | Summer | Season | OG | 1 | 0 | SPY | 43 | (EULCS,1) |
| | LCK | 2017 | Summer | Season | EEW | 1 | 0 | bbq | 42 | (LCK,1) |
| | LCK | 2017 | Summer | Season | bbq | 1 | 0 | EEW | 29 | (LCK,1) |
| | LCK | 2017 | Summer | Season | MVP | 0 | 1 | SKT | 39 | (LCK,1) |
| YUVRAJ | NALCS | 2015 | Spring | Season | TIP | 0 | 1 | CLG | 31 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | TSM | 1 | 0 | WFX | 33 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | CLG | 1 | 0 | C9 | 38 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | DIG | 0 | 1 | WFX | 40 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | TSM | 1 | 0 | TL | 39 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | TIP | 1 | 0 | T8 | 40 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | CST | 0 | 1 | GV | 39 | (NALCS,1) |
| | NALCS | 2015 | Spring | Season | GV | 0 | 1 | TIP | 49 | (NALCS,1) |
| | LLN | 2017 | Spring | Season | JTH | 0 | 1 | LYN | 33 | (LLN,1) |
| | LLN | 2017 | Spring | Season | LYN | 1 | 0 | JTH | 26 | (LLN,1) |
| | LLN | 2017 | Spring | Season | D9 | 0 | 1 | INF | 31 | (LLN,1) |
| | LLN | 2017 | Spring | Season | INF | 1 | 0 | D9 | 34 | (LLN,1) |
| | LLN | 2017 | Spring | Season | ZTG | 0 | 1 | GG | 39 | (LLN,1) |
| RAINA | NALCS | 2016 | Spring | Season | CLG | 0 | 1 | IMT | 33 | (NALCS,1) |
| | NALCS | 2016 | Spring | Season | TSM | 1 | 0 | FOX | 39 | (NALCS,1) |
| | LCK | 2016 | Summer | Playoffs | ROX | 1 | 0 | kt | 34 | (LCK,1) |
| | LCK | 2016 | Summer | Playoffs | kt | 1 | 0 | ROX | 38 | (LCK,1) |
| | LLN | 2017 | Summer | Season | INF | 1 | 0 | GG | 33 | (LLN,1) |

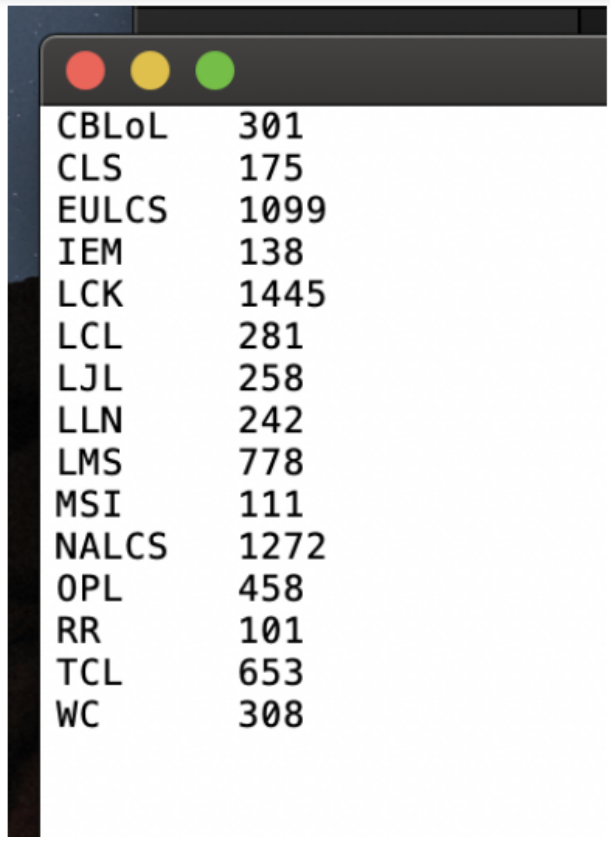
## *MatchAnalysisMapper1.java*

```
package Analytics;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MatchAnalysisMapper1 extends Mapper<LongWritable, Text, Text, IntWritable> {

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException {


		Text word = new Text();

		String line = value.toString();

		word.set(line.split(",")[0]);

		IntWritable one = new IntWritable(1);

		con.write(word, one);
	}
}
```

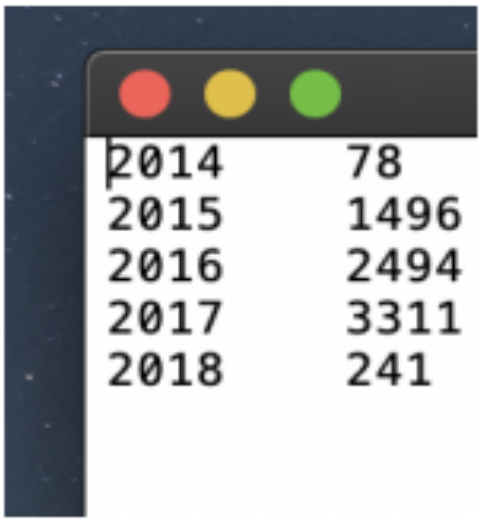
| | | | | | | | | | Mapper Output | | Shuffler |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NALCS | 2015 | Spring | Season | TSM | 1 | 0 | C9 | 40 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | CST | 0 | 1 | DIG | 38 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | WFX | 1 | 0 | GV | 40 | (NALCS,1) | | (NALCS,1) |
| EULCS | 2016 | Summer | Season | SPY | 1 | 0 | OG | 41 | (EULCS,1) | | (NALCS,1) |
| EULCS | 2016 | Summer | Season | OG | 1 | 0 | SPY | 43 | (EULCS,1) | | (NALCS,1) |
| LCK | 2017 | Summer | Season | EEW | 1 | 0 | bbq | 42 | (LCK,1) | | (NALCS,1) |
| LCK | 2017 | Summer | Season | bbq | 1 | 0 | EEW | 29 | (LCK,1) | | (NALCS,1) |
| LCK | 2017 | Summer | Season | MVP | 0 | 1 | SKT | 39 | (LCK,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | TIP | 0 | 1 | CLG | 31 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | TSM | 1 | 0 | WFX | 33 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | CLG | 1 | 0 | C9 | 38 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | DIG | 0 | 1 | WFX | 40 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | TSM | 1 | 0 | TL | 39 | (NALCS,1) | | (NALCS,1) |
| NALCS | 2015 | Spring | Season | TIP | 1 | 0 | T8 | 40 | (NALCS,1) | | (EULCS,1) |
| NALCS | 2015 | Spring | Season | CST | 0 | 1 | GV | 39 | (NALCS,1) | | (EULCS,1) |
| NALCS | 2015 | Spring | Season | GV | 0 | 1 | TIP | 49 | (NALCS,1) | | (LCK,1) |
| LLN | 2017 | Spring | Season | JTH | 0 | 1 | LYN | 33 | (LLN,1) | | (LCK,1) |
| LLN | 2017 | Spring | Season | LYN | 1 | 0 | JTH | 26 | (LLN,1) | | (LCK,1) |
| LLN | 2017 | Spring | Season | D9 | 0 | 1 | INF | 31 | (LLN,1) | | (LCK,1) |
| LLN | 2017 | Spring | Season | INF | 1 | 0 | D9 | 34 | (LLN,1) | | (LCK,1) |
| LLN | 2017 | Spring | Season | ZTG | 0 | 1 | GG | 39 | (LLN,1) | | (LLN,1) |
| NALCS | 2016 | Spring | Season | CLG | 0 | 1 | IMT | 33 | (NALCS,1) | | (LLN,1) |
| NALCS | 2016 | Spring | Season | TSM | 1 | 0 | FOX | 39 | (NALCS,1) | | (LLN,1) |
| LCK | 2016 | Summer | Playoffs | ROX | 1 | 0 | kt | 34 | (LCK,1) | | (LLN,1) |
| LCK | 2016 | Summer | Playoffs | kt | 1 | 0 | ROX | 38 | (LCK,1) | | (LLN,1) |
| LLN | 2017 | Summer | Season | INF | 1 | 0 | GG | 33 | (LLN,1) | | (LLN,1) |

## *MatchAnalysisReducer1.java*

```
package Analytics;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MatchAnalysisReducer1 extends Reducer<Text, Text, Text, IntWritable> {
   public void reduce(Text key, Iterable<Text> values, Context con) throws IOException, Interrupte

            int sum = 0;

            Iterator it = values.iterator();

            while (it.hasNext()) {
                  sum += Integer.parseInt(it.next().toString());
            }

            con.write(key, new IntWritable(sum));
      }
}
```



## 2. Find total number of matches played in a year.



## *MatchAnalysisMapper3.java*

```
package Analytics;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MatchAnalysisMapper3 extends Mapper<LongWritable, Text, Text, IntWritable> {

	public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException {
		Text word = new Text();

		String line = value.toString();

		String[] split = line.split(",");

		word.set(split[4] + " vs " + split[7]);

		IntWritable one = new IntWritable(Integer.parseInt(split[5]));

		con.write(word, one);
	}
}
```

## *MatchAnalysisReducer3.java*

```
package Analytics;

import java.io.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class MatchAnalysisReducer3 extends Reducer<Text, IntWritable, Text, IntWritable> {

      int max = 0;
      Text maxWord = new Text();
      int temp = 0;

      public void reduce(Text key, Iterable<IntWritable> values, Context con) throws IOException
    {


            for (IntWritable value : values) {
                  temp = value.get();

                  if (temp > max) {
                        max = temp;
                        maxWord.set(key);
                  }
            }
            context.write(maxWord, new IntWritable(max));

     }
}
```

## If a tournament is played between team1 vs team2. Find how many times team1 has won and how many times team2 has won.

```
package Match.Match;

import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;


public class MatchMapper3 extends Mapper<LongWritable,Text,Text,Text> {

public void map(LongWritable key,Text value,Context con)
throws IOException,InterruptedException {

String line = value.toString();

String[] words = line.split(",");

String outputKey = words[4] + " " + words[7];

String outputValue = words[5] + " " + words[6];

con.write(new Text(outputKey),new Text(outputValue));

}
}
```

```
package Match.Match;


import java.io.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;


public class MatchReducer3 extends Reducer<Text,Text,Text,Text> {

public void reduce(Text word,Iterable<Text> values,Context con)
throws IOException,InterruptedException {

int sum_first = 0;
int sum_second = 0;

        for(Text value:values)
        {

                String v = value.toString();

                String[] val = v.split(" ");

                String c = "1";

                if(val[0].equals(c)) {

                sum_first ++;

                }
                else {

                sum_second++;

                }

        }

    String outputValue = sum_first + " " + sum_second;

    con.write(word,new Text(outputValue));

  }
}
```


AFs CJ 2 0
AFs CJE 2 1
AFs EEW 2 0
AFs EMF 1 0
AFs ESC 1 1
AFs JAG 7 7
AFs KDM 4 1
AFs KSV 0 1
AFs KT 2 0
AFs KZ 0 1
AFs LZ 7 3
AFs MVP 8 5
AFs ROX 4 8
AFs SKT 7 3
AFs SSB 1 1
AFs SSG 6 8
AFs bbq 5 1
AFs kt 6 5
AHQ BJK 1 0
AHQ C9 1 1
AHQ DP 1 0
AHQ EDG 2 3
AHQ FNC 1 0
AHQ FW 0 4
AHQ H2K 0 1
AHQ HKA 0 1
AHQ IG 1 0
AHQ ITZ 1 0
AHQ JT 3 1
AHQ M17 1 0