**2CS702 - Big Data Analytics**

# Practical 7

**Aim**: - **Implement PCY/Multi-Hash/SON algorithm for identification of frequent item set by handling larger datasets in main memory.**

**Author: Darshil Maru 20BCE514**

**Guide: Dr. Purnima Gandhi**

**AIM : Implement PCY/Multi-Hash/SON algorithm for identification of frequent item set by handling larger datasets in main memory.**

## Steps Involved-

We installed designed MapReduce Algorithms to perform the analytic of implementing the apriori algorithm on the given data set to find the frequent itemset

## Background

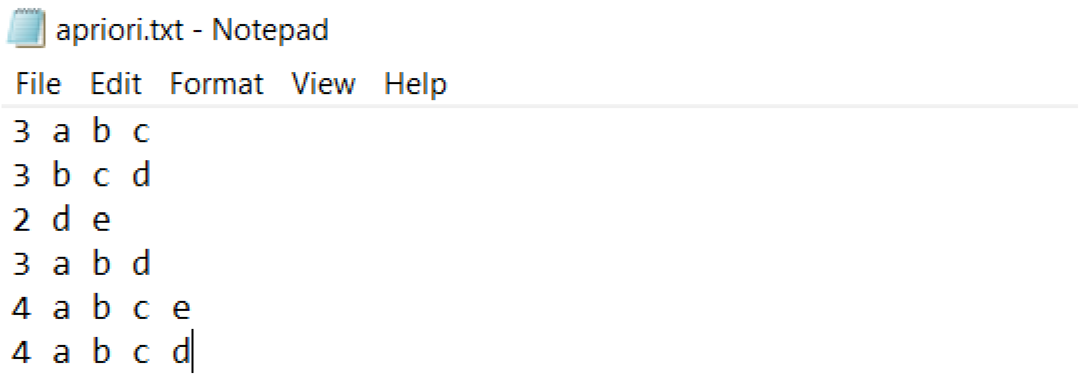
### Hadoop

Apache Hadoop is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model

### MapReduce

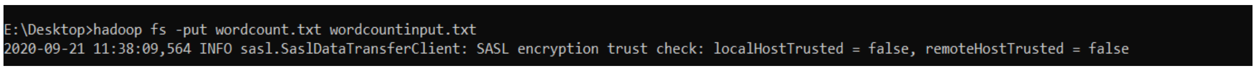
MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a map procedure, which performs filtering and sorting, and a reduce method, which performs a summary operation.

- **Input File**



```
3 a b c
3 b c d
2 d e
3 a b d
4 a b c e
4 a b c d
```

Input file consisted of multiple lines. Each line was a transaction where first word of each line was the number of items purchased. the file was copied to the HDFS system, by using the command hadoop fs -put apriori.txt apriori



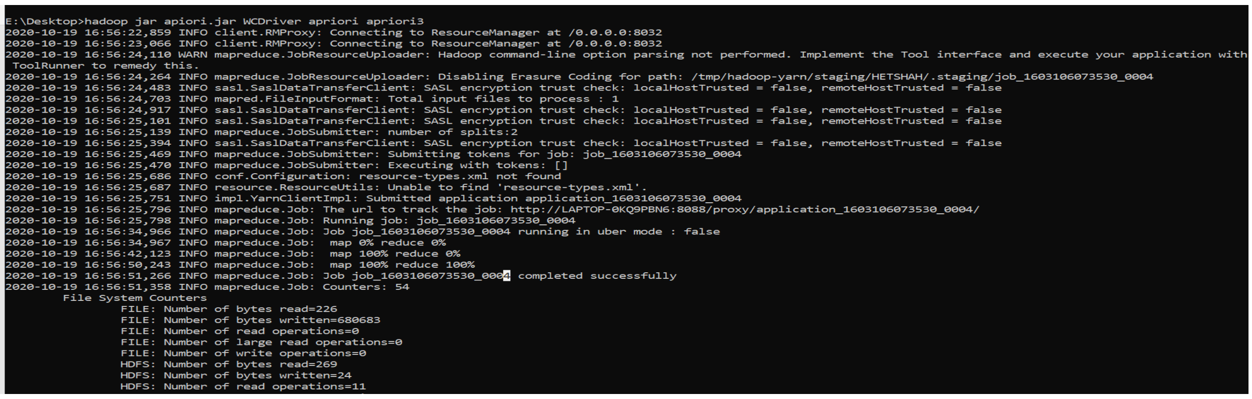
```
E:\Desktop>hadoop fs -put wordcount.txt wordcountinput.txt
2020-09-21 11:38:09,564 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
```

## How to run

Each MapReduce task had 3 class files associated with it WCDriver, WCMapper and WCReducer. WCDriver was the main file and it would call the mapper and reducer
All the files were kept in the same package and after importing all the Hadoop extensions, the files were exported as a jar file which was then used to run the mapreduce program

## To run the program we write

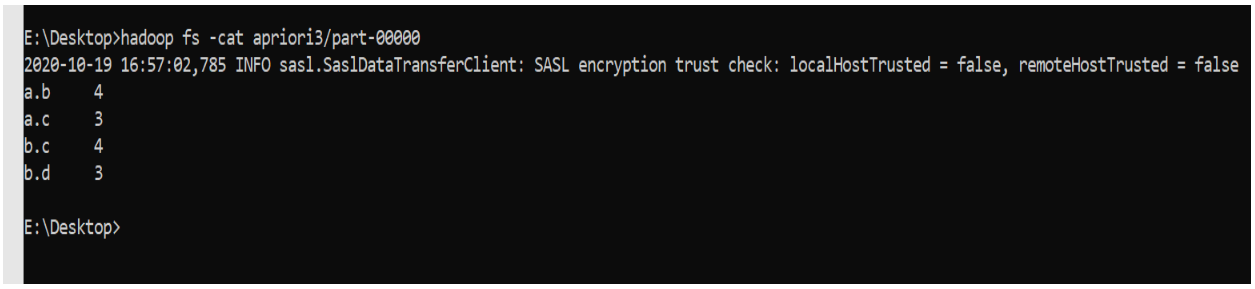
hadoop jar apiori.jar WCDriver apriori apriori3



```
E:\Desktop>hadoop jar apiori.jar WCDriver apriori apriori3
2020-10-19 16:56:22,859 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2020-10-19 16:56:23,066 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2020-10-19 16:56:24,110 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with
 ToolRunner to remedy this.
2020-10-19 16:56:24,264 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/HETSHAH/.staging/job_1603106073530_0004
2020-10-19 16:56:24,483 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-10-19 16:56:24,703 INFO mapred.FileInputFormat: Total input files to process : 1
2020-10-19 16:56:24,917 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-10-19 16:56:25,101 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-10-19 16:56:25,139 INFO mapreduce.JobSubmitter: number of splits:2
2020-10-19 16:56:25,394 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-10-19 16:56:25,469 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1603106073530_0004
2020-10-19 16:56:25,470 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-10-19 16:56:25,686 INFO conf.Configuration: resource-types.xml not found
2020-10-19 16:56:25,687 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2020-10-19 16:56:25,751 INFO impl.YarnClientImpl: Submitted application application_1603106073530_0004
2020-10-19 16:56:25,796 INFO mapreduce.Job: The url to track the job: http://LAPTOP-0KQ9PBN6:8088/proxy/application_1603106073530_0004/
2020-10-19 16:56:25,798 INFO mapreduce.Job: Running job: job_1603106073530_0004
2020-10-19 16:56:34,966 INFO mapreduce.Job: Job job_1603106073530_0004 running in uber mode : false
2020-10-19 16:56:34,967 INFO mapreduce.Job:  map 0% reduce 0%
2020-10-19 16:56:42,123 INFO mapreduce.Job:  map 100% reduce 0%
2020-10-19 16:56:50,243 INFO mapreduce.Job:  map 100% reduce 100%
2020-10-19 16:56:51,266 INFO mapreduce.Job: Job job_1603106073530_0004 completed successfully
2020-10-19 16:56:51,358 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=226
                FILE: Number of bytes written=680683
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=269
                HDFS: Number of bytes written=24
                HDFS: Number of read operations=11
```

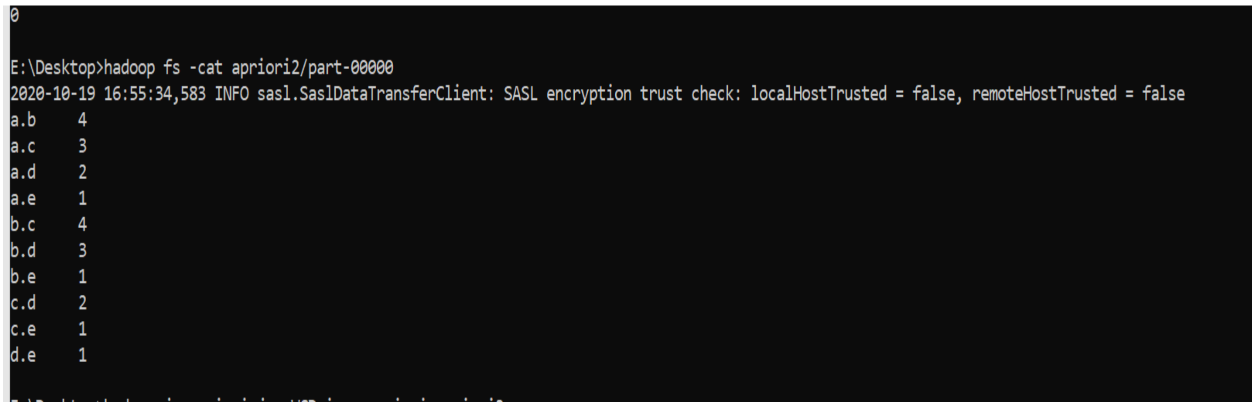
## To print the output we write

hadoop fs -cat apriori3/part-00000



```
E:\Desktop>hadoop fs -cat apriori3/part-00000
2020-10-19 16:57:02,785 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
a.b     4
a.c     3
b.c     4
b.d     3

E:\Desktop>
```

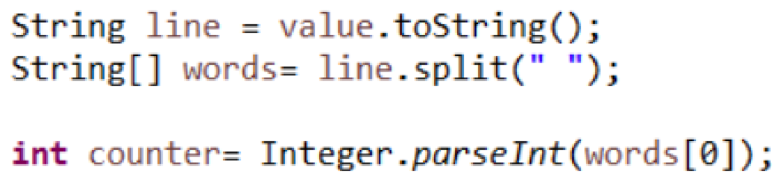
All the frequent data itemset without the support threshold



```
0

E:\Desktop>hadoop fs -cat apriori2/part-00000
2020-10-19 16:55:34,583 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
a.b     4
a.c     3
a.d     2
a.e     1
b.c     4
b.d     3
b.e     1
c.d     2
c.e     1
d.e     1
```

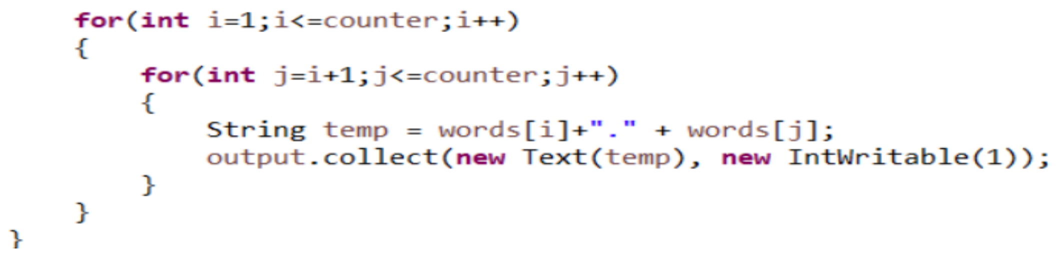
**Logic of Mapper and Reducer**

The text file is read and the split into the words using the split and the number of items brought are stored in the counter variable



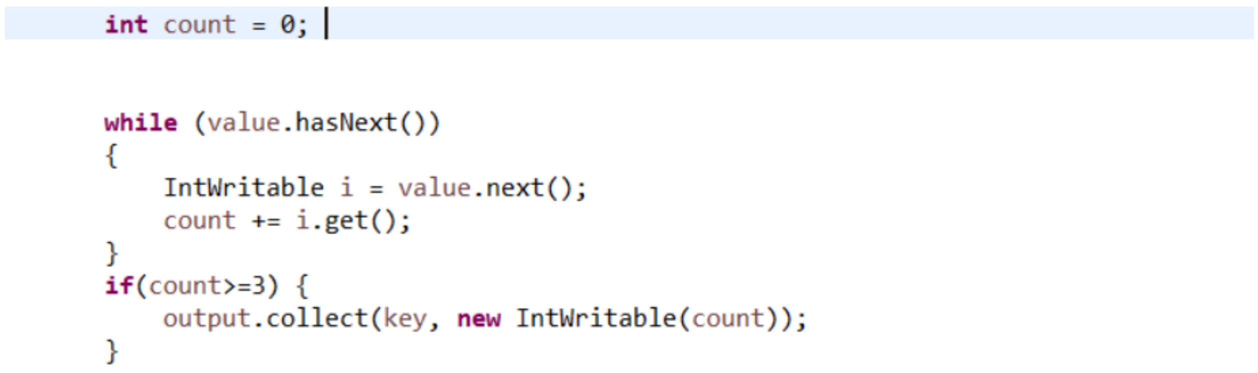
```
String line = value.toString();
String[] words= line.split(" ");

int counter= Integer.parseInt(words[0]);
```

Two for loops are used to find the association rules and the mapper emits the combination of the words and 1



```
    for(int i=1;i<=counter;i++)
    {
        for(int j=i+1;j<=counter;j++)
        {
            String temp = words[i]+"." + words[j];
            output.collect(new Text(temp), new IntWritable(1));
        }
    }
}
```

The reducer collects the word and then counts the number of instances that word has occurs and then emits (combination ,count) if the count is more than support threshold



```java
int count = 0;

while (value.hasNext())
{
    IntWritable i = value.next();
    count += i.get();
}
if(count>=3) {
    output.collect(key, new IntWritable(count));
}
```

```java
// Counting the frequency of each words
while (value.hasNext())
{
    IntWritable i = value.next();
    count += i.get();
}
```

**Conclusion**

In this practical we learned how to program using the MapReduce programming paradigm and used it to perform the apriori algorithm on the given list of transactions.