ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

-----🕮-----

Môn học: Dữ liệu lớn

Thực hành buổi 2

Lớp: **IS405.O11**

Sinh viên thực hiện:

20521068 – Nguyễn Bảo Anh

TP. Hồ Chí Minh, *Thứ Ba, 07 Tháng Mười Một 2023*

1. **Bài tập 1**: Tạo cây thư mục trong hệ thống tập tin HDFS theo yêu cầu của GVHD.

   hadoop fs -mkdir /user/banhnguyen/UIT

   hadoop fs -mkdir /user/banhnguyen/UIT/HTTT

   hadoop fs -mkdir /user/banhnguyen/UIT/CNTT

   hadoop fs -mkdir /user/banhnguyen/UIT/KHMT

   hadoop fs -mkdir /user/banhnguyen/UIT/KTPM

   hadoop fs -mkdir /user/banhnguyen/UIT/MMT

   hadoop fs -mkdir /user/banhnguyen/UIT/KTMT

   hadoop fs -mkdir /user/banhnguyen/UIT/HTTT/HTTTQL

   hadoop fs -mkdir /user/banhnguyen/UIT/HTTT/HTTTTM

   hadoop fs -mkdir /user/banhnguyen/UIT/HTTT/TMDT

2. **Bài tập 2:** Tạo project WordCount bằng ngôn ngữ Java, thêm các thư viện cần thiết của Apache Hadoop, có thể tham khảo mã nguồn trên trang chủ. Biên dịch và chạy thử chương trình.

   - Mã nguồn:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
```

```
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

3. Bài tập 3:

Download dữ liệu bán hàng tại địa chỉ:

http://fimi.uantwerpen.be/data/retail.dat.gz

Tập dữ liệu đầu vào là dữ liệu các giao dịch có dạng:

```
25 52 164 240 274 328 368 448 538 561 630 687 730 775 825 834
39 120 124 205 401 581 704 814 825 834
35 249 674 712 733 759 854 950
39 422 449 704 825 857 895 937 954 964
15 229 262 283 294 352 381 708 738 766 853 883 966 978
26 104 143 320 569 620 798
7 185 214 350 529 658 682 782 809 849 883 947 970 979
227 390
71 192 208 272 279 280 300 333 496 529 530 597 618 674 675 720 855 914 932
```

Với mỗi dòng ghi nhận một giao dịch (một tập các sản phẩm được mua cùng nhau).

Yêu cầu: Lưu dữ liệu vào HDFS và viết 01 chương trình MapReduce có các chức năng:

a) Thống kê sự xuất hiện đồng thời của từng cặp sản phẩm: count (A, B) = số giao dịch chứa đồng thời A và B.

- Mã nguồn:

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Bai3a {

  public static class ProductPairsMapper extends Mapper<Object, Text, Text, IntWritable>
{
    private Text pair = new Text();
    private final static IntWritable one = new IntWritable(1);
```

```
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      String[] products = value.toString().split(" ");
      List<String> productPairs = generatePairs(products);

      for (String productPair : productPairs) {
        pair.set(productPair);
        context.write(pair, one);
      }
    }

    private List<String> generatePairs(String[] products) {
      List<String> pairs = new ArrayList<>();

      for (int i = 0; i < products.length; i++) {
        for (int j = i + 1; j < products.length; j++) {
          pairs.add(""+products[i] + " - " + products[j]);
        }
      }
      return pairs;
    }
  }

  public static class ProductPairsReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
      int sum = 0;

      for (IntWritable val : values) {
        sum += val.get();
      }

      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "ProductPairs");

    job.setJarByClass(Bai3a.class);
```

```
        job.setMapperClass(ProductPairsMapper.class);
        job.setCombinerClass(ProductPairsReducer.class);
        job.setReducerClass(ProductPairsReducer.class);

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

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

b) Xác suất mua sản phẩm B khi đã mua sản phẩm A (xác suất có điều kiện - conditional probability): prob (B | A) = count (A, B) / count (A).

c) Thống kê sự xuất hiện đồng thời của từng bộ ba sản phẩm: count (A, B, C) = số giao dịch chứa đồng thời A, B và C.

d) Xác suất mua sản phẩm A khi đã mua sản phẩm B và C (xác suất có điều kiện - conditional probability): prob (A | B, C) = count (A, B, C) / count (B, C).

4. **Bài tập 4:** Download dữ liệu thông tin về giá cả một số mặt hàng thiết yếu theo thị trường TP.HCM tại: Thông tin giá cả thị trường | Cổng dữ liệu mở Thành Phố Hồ Chí Minh (hochiminhcity.gov.vn).

   Yêu cầu: Lưu dữ liệu vào HDFS và viết 01 chương trình MapReduce có các chức năng:

   a) Thống kê theo ngày cập nhật, số lượng mặt hàng có giá cao hơn một ngưỡng giá nhập vào.

   - Mã nguồn:

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Bai4a {

    public static class PriceCountMapper extends Mapper<LongWritable, Text, Text,
LongWritable> {
        private static final double THRESHOLD_PRICE = 100000.0;

        @Override
        protected void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {
            String[] fields = value.toString().split(",");
            if (fields.length == 4) {
            String date = fields[3].split(" ")[0];
            try {
                double price = Double.parseDouble(fields[1]);
                if (price > THRESHOLD_PRICE) {
                    context.write(new Text(date), new LongWritable(1));
                }
            } catch (NumberFormatException e) {

            }
        }
        }
    }

    public static class PriceCountReducer extends Reducer<Text, LongWritable, Text,
LongWritable> {
        @Override
        protected void reduce(Text key, Iterable<LongWritable> values, Context context)
                throws IOException, InterruptedException {
            long count = 0;
            for (LongWritable value : values) {
                count += value.get();
            }
            context.write(key, new LongWritable(count));
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: PriceCountMapReduce <input_path> <output_path>");
            System.exit(1);
        }
```

```
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Price Count");

        job.setJarByClass(Bai4a.class);
        job.setMapperClass(PriceCountMapper.class);
        job.setCombinerClass(PriceCountReducer.class);
        job.setReducerClass(PriceCountReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

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

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

b) Giá bán trung bình của từng mặt hàng.

- Mã nguồn:

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Bai4b {

    public static class PriceAverageMapper extends Mapper<LongWritable, Text, Text,
DoubleWritable> {
        @Override
        protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            String[] fields = value.toString().split(",");
            if (fields.length == 4) {
```

```
                String itemName = fields[0];
                try {
                    double price = Double.parseDouble(fields[1]);
                    context.write(new Text(itemName), new DoubleWritable(price));
                } catch (NumberFormatException e) {
                    // Xử lý lỗi nếu dữ liệu không hợp lệ
                }
            }
        }
    }

    public static class PriceAverageReducer extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
        @Override
        protected void reduce(Text key, Iterable<DoubleWritable> values, Context
context)
                throws IOException, InterruptedException {
            double sum = 0.0;
            int count = 0;
            for (DoubleWritable value : values) {
                sum += value.get();
                count++;
            }
            double average = sum / count;
            context.write(key, new DoubleWritable(average));
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: PriceAverageMapReduce <input_path>
<output_path>");
            System.exit(1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Bai4b");

        job.setJarByClass(Bai4b.class);
        job.setMapperClass(PriceAverageMapper.class);
        job.setReducerClass(PriceAverageReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);
```

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

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

c) Với mỗi mặt hàng, cho biết giá bán cao nhất và thấp nhất.

- Mã nguồn:

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Bai4c {

    public static class PriceMapper extends Mapper<Object, Text, Text, Text> {
        private Text product = new Text();
        private Text priceInfo = new Text();

        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            String[] fields = value.toString().split(",");
            if (fields.length == 4) {
                product.set(fields[0]);
                priceInfo.set(fields[1]);
                context.write(product, priceInfo);
            }
        }
    }

    public static class PriceReducer extends Reducer<Text, Text, Text, Text> {
        private Text maxPrice = new Text();
        private Text minPrice = new Text();
```

```
        public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
            double max = Double.MIN_VALUE;
            double min = Double.MAX_VALUE;

            for (Text value : values) {
                double price = Double.parseDouble(value.toString());
                if (price > max) {
                    max = price;
                }
                if (price < min) {
                    min = price;
                }
            }

            maxPrice.set("Max Price: " + max);
            minPrice.set("Min Price: " + min);

            context.write(key, new Text(maxPrice.toString() + ", " +
minPrice.toString()));
        }
    }

    public static void main(String[] args) throws Exception {
        Job job = Job.getInstance();
        job.setJarByClass(Bai4c.class);
        job.setMapperClass(PriceMapper.class);
        job.setReducerClass(PriceReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

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

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```