COSC 6397
Big Data Analytics

Discussion of 1st homework

Edgar Gabriel
Spring 2015

Document

- Problem description
- Resource description
  - Has to describe the cluster, not just one node
  - Put in information that is useful, (e.g. memory, disk, number of nodes) and
- Submitting output files
- Number of measurements performed and values presented
  - Min: typically used if:
    - other users/events can influence the result
    - evaluating an algorithms
  - Max: should be used if caching could influence performance and that is not desired
  - average: somewhere inbetween

UNIVERSITYof HOUSTON
Connecting individual data points

- Implies a value for intermediate points
- definitely wrong for 1.5 reducers
- use dots/bars in these cases

Part 1

Mapper:
- parse line
- Check for question (e.g. `PostIdType==2`)
- Retrieve `AnswerCount`

```java
  Text key = new Text("1")
  context.write(key, new IntWritable(answerCount))
```

Reducer

```java
  for (IntWritable v : values) {
    sum += v.get();
    count++;
  }
  context.write(key, new FloatWritable(sum/count));
```
Part 3

• Mapper
  - Parse line
  - Check for answer (PostIdType == 2 )
  - Retrieve OwnerUserId
    Text key = new Text (owneruserid);
    context.write (key, new IntWritable(1));
  
• Reducer
  int count = 0;
  for (IntWritable v : values) {
    count++;
  }
  context.write(key, new IntWritable(count));

Part 2

• Matching of an accepted answer and an answer ‘entry’ requires extra step
• Step 1: Mapper
  - If question (PostIdType==1 ) {
    • Identify AcceptedAnswerId
    • context.write (acceptedanswerid, 0 );
  - If answer (PostIdType == 2 ) {
    • Identify Score
    • context.write (id, score);
• Step 1: reducer
  - Since key’s with same content will be sent to the same reduce instance, all iterables that have two entries are an accepted answer, all with one entry are not accepted answer
  
  ```java
  int count = 0;
  int score = 0;
  for (IntWritable v : values) {
    score += v.get();
    count++;
  }
  if (count == 2)
    context.write(new Text("A"), new IntWritable(score));
  else
    context.write(new Text("U"), new IntWritable(score));
  ```

• Step 2:
  - mapper just parses the intermediary input file to separate key and value again
  - reducer calculates the average of all values assigned to the key A and key U separately

• Alternative:
  - for questions: use id
  - for answers: user parent_id
  - fewer different key-value pairs
  - more work for reducer
  - reducer code slightly more complicated
Some remarks:
- intermediary directory ideally provided as input argument or automatically appended and removed again
- Using fixed string/value as key
  -> remember: intermediary key-value pairs sorted/shuffled such that all values with the same key end up at the same reducer!
  -> if key is a fixed string/value number of reducers invoked is always 1
  -> e.g. in part 1, does not make sense to have more than 1 reducer

Chainmapper

The ChainMapper class allows to use multiple Mapper classes within a single Map task
Mapper classes are invoked in such that the output the first becomes mapper is the input of the second etc.

```java
Configuration conf1 = new Configuration(false);
ChainMapper.addMapper(job, Mapper1.class,
                        IntWritable.class, Text.class, Text.class,
                        IntWritable.class, conf1);

Configuration conf2 = new Configuration(false);
ChainMapper.addMapper(job, Mapper2.class, Text.class,
                        IntWritable.class, Text.class, IntWritable.class,
                        conf2);
```
Problem with ChainMapper

- **Mapper 1:**
  - if question: add to a global external structure (e.g. hashmap, array, ...)
  - if answer: `context.write (id, score);
- **Mapper 2:**
  - input is the `context.write` from the 1st Mapper
  - check whether id is in the external global structure
  - if yes -> `context.write` as an accepted answer
  - if no -> `context.write` an a not accepted answer
- **Reducer**
  - calculate average for all accepted and not accepted answers

```java
public class hw1part2 {
    public static ArrayList<String> acceptans = new ArrayList<String>();

    public static class Mapper1 extends Mapper<...> {
        void map(LongWritable key, Text value, Context con) {
            if (question)
                acceptans.add(acceptedanswerid)
            if (answer)
                con.write(...);
        }
    }

    public static class Mapper2 extends Mapper<...> {
        void map(IntWritable key, IntWritable value, Context con) {
            if (acceptans.contains(id))
                con.write(new Text("A", score);
            else
                con.write(new Text("U", score);
        }
    }
}
```

UNIVERSITYof HOUSTON
Problem with ChainMapper

- Cluster is distributed memory architecture
  - different jvm’s do not share memory!
  - global list/hashmap etc contains only data from mappers that were running on that node
  - if a question and the accepted answer are assigned to different nodes, you will not find it

Problem with chainmappers

Hadoop input file splits into multiple pieces

large input file → input part1 rows:0-1000 → ... → input part n rows:99000-10000
input part1
rows:0-1000

assigned to shark01

```java
jvm started for Mapper on shark01
ArrayList<String> acceptans = new ArrayList<String>();

setup()
for ( each line in input part 1) {
    map(current_line)
}
cleanup()

e.g. inside map() {
    acceptans.add ("3")
}
```

input part n
rows:99000-10000

assigned to shark18

```java
jvm started for Mapper on shark01
ArrayList<String> acceptans = new ArrayList<String>();

setup()
for ( each line in input part 1) {
    map(current_line)
}
cleanup()

e.g. inside map() {
    acceptans.add ("99025")
}
```