[Recitation 3]

Paahuni Khandelwal
Email: cs435@cs.colostate.edu

13th September, 2019
Issue: NameNode is in Safemode?

What is the reason for name node to get in safe mode?

**SAFEMODE**
- State when NameNode is under maintenance.
- No operations such as reading/writing into HDFS allowed - No modifications allowed
- HDFS in read-only mode
- Data Blocks - Smallest unit of data (128 MB)
- When certain % of data blocks are unavailable

**Job of NameNode -**
- Load the filesystem namespace from last saved FsImage
- Edits log files
- Merging log files into FsImage
- Wait for data nodes (workers) to send information on the blocks stored by them.
- During safe mode, these block reports are collected
- As name node start it enters into safe mode
- Leaving the safe mode happens when the Datanodes reports the most blocks are available

Leaving SAFEMODE
$HADOOP_HOME/bin/hdfs dfsadmin -safemode leave
Programming Assignment 1

• This will be worth 7 points (7% of your total grades)
• Submission due on 30\textsuperscript{Th} September by 5 pm
• \textbf{Dataset:} Summary of 1.5 million Wikipedia articles (~1 GB)

Title\_of\_Article-1<====>DocumentID<====>Summary\_of\_Article-1
NEWLINE
NEWLINE
Title\_of\_Article-2<====>DocumentID<====>Summary\_of\_Article-2
NEWLINE
NEWLINE
Pre-processing of Dataset

- Consider only alphabetic and numeric text
- Convert upper cases to lower cases
- Example (Original → processed → toLowerCase)
  - D.I.A. → DIA → dia
  - South* → South → south
  - (U-S%A) → USA → usa
  - D.C., → DC → dc
  - 9.8 → 98 → 98
Preprocessing done in map function

Night Hunter<====>4264696<====>The Night Hunter is a 1996 horror film directed by Rick Jacob

• Removing non-alphanumeric characters

```java
public void map(key, value){
    Check if value is not empty{
        use StringTokenizer to split value on <====>
        extract appropriate value from tokenized value (2nd value of array)
        iterate though each token in summary of document {
            Perform preprocessing on each word
            ........
            Emit output
        }
    }
}}
```
Preprocessing Code

```java
if (!value.toString().isEmpty()) {
    StringTokenizer token = new StringTokenizer(value.toString().split("<====>")[2]);
    while (token.hasMoreTokens()) {
        String out = token.nextToken().replaceAll("[^A-Za-z0-9]","").toLowerCase();
        ....
        ....
    }
}

Night Hunter<====>4264696<====>The Night Hunter is a 1996 horror film directed by Rick Jacob
```
Profile 1

- Must be sorted in alphabetical order (ascending)
- First 500 unigrams in the dataset
- Eliminate duplicates
- For designing you can use combiner to eliminate local duplicates
Profile 1 Solution

- Similar to Word count program (without counting frequency of words)
- Mapper function emits (key, value) pair which is (word, null) or (word, 1)
- Reducer will emit key from mapper function i.e. (key, null) as result
  
  ```java
  Context.write(key, NullWritable.get())
  ```

- Use of reducer?
  - For sorting in ascending alphabetical order (output from mapper)
- By default, only 1 reducer is set for MapReduce job
- How to set the number of reducer more than 1?
  - Inside driver code, set `job.setNumReducerTask(n)`
- The Mapper outputs are sorted and partitioned per Reducer
- The total number of partitions is the same as the number of reduce tasks for the job
- Now, we get sorted output per reducer, is it globally sorted?
No!

To perform global sorting, control which keys go to which Reducer by implementing a custom Partitioner.
Custom Partitioner

- Use Partitioner to get globally sorted output
- Defined in driver class
- `Job.setPartitionerClass(UnigramPartitioner.class)`

```java
public static class UnigramPartitioner extends Partitioner<Text, NullWritable> {
    @Override
    public int getPartition(Text key, NullWritable value, int numReduceTasks) {
        if (numReduceTasks == n) {
            Character partitionKey = key.toString().toLowerCase().charAt(0);
            else if (partitionKey > 'g' && partitionKey <= 'n') {
                return 0;
            } else if (partitionKey >= 'o' && partitionKey <= 't') {
                return 2;
            } else if (partitionKey >= 'o' && partitionKey <= 't') {
                return 2;
            } else {
                return n;
            }
        }
    }
}
```
Solution A: Top N for Profile 1

- Using Top-N design pattern
- A TreeMap sorts on the key. The default ordering is ascending
- Then, if there are more than n records in TreeMap, the first element (lowest value) can be removed.
- After all the records have been processed, the top n records in the TreeMap are output to the reducers in the cleanup method
- Inside Mapper:
  - Initialize TreeMap<Text,Integer>
  - Store and update first 500 unigrams in the TreeMap
  - Use cleanup() method to send Local 500 unigrams from each mapper (Use nullWritable as key)
- Inside Reducer:
  - Initialize TreeMap<Text,Integer>
  - Store and update TreeMap to get global first 500 unigrams
Solution B: Top N for Profile 1

- Instead of passing local 500 unigrams from mapper, filter global 500 unigrams at reducer-side
- Not as efficient as previous solution
- Inside **Mapper**:
  - Emit key, value pair as <unigram, 1> or <unigram, null>
- Inside **Reducer**:
  - Initialize TreeMap<Text, Integer>
  - Store and update TreeMap to get global first 500 unigrams
- Make sure you set number of reducer is set to 1.
PROFILE 2

- A list of top 500 unigrams and their frequencies within each article
- Each article has unique integral Document ID
- Output should be grouped by Document ID
- Output files will correspond to the number of reducer used
- Top 500 unigrams per document
- Total output records $\leq 500 \times \text{(no. of unique documents)}$

<table>
<thead>
<tr>
<th>Document ID</th>
<th>Term</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>453673</td>
<td>biological</td>
<td>3</td>
</tr>
<tr>
<td>453673</td>
<td>people</td>
<td>2</td>
</tr>
</tbody>
</table>

...
Profile 2 Solution

- Composite Keys
- For Profile 2, use may use **CompositeKey** concept in MapReduce
- Output of Mapper:
  - \((\{\text{DocumentID}, \text{unigram}\}, 1)\)
- Output of Reducer will be in form:
  - \((\{\text{DocumentID}, \text{unigram}\}, \text{frequency})\)
- Use TopK concept to output based on frequency
Profile 3

- A list of top 500 unigrams and their frequencies in the corpus
- List should be sorted from most frequent unigrams to least frequent ones
- Solution to generate Profile 3 is combination of Profile 1 and Profile 2
- We will list unigram with its total occurrence in the complete dataset
Profile 3 Solution

○ Count frequency in reducer as we did in WordCount program in PA0
○ Sorting is based on the value of (key, value) pair output from reducer
○ You can use second MapReduce job
○ To perform sorting, reverse key-value pair being sent by first map reduce job
Next Recitation

- Will be on 20th September (Friday) in CS130 from 4 to 5pm.
- Going through solutions for Profile 2 and Profile 3
- Wrapping up PA1
- Comparator class for Profile 2 and Profile 3
Questions?