THE UNIVERSITY OF SYDNEY

School of Information Technologies
A/Prof Uwe Roehm

COMP5349: Cloud Computing **1.Sem./2017**

Assignment 1: Hadoop MapReduce Programming

Group Work: 15% **13.04.2017**

Introduction

In this assignment you have to write a series of Hadoop jobs to analyze a data set from `flickr.com`. The analysis is structured into three separate tasks which build on each other. You will practice basic Hadoop programming features and also observe the ease of use of the Hadoop framework. The data set is the same as those you have used in the labs.

Input Data Set Description

The main input data can be found in the directory `/share/photo` in our Hadoop cluster. The directory has several text files all with names like `n0*.txt`. All of them have the same format, with each line representing a photo record. These data are downloaded from Flickr website.

Each photo record contains the following information delimited by `tab`:

```
photo-id  owner  tags  date-taken place-id  accuracy
```

An additional input file `place.txt` can be found in the directory `/share`. This file stores place data. Each line contains a place record with the following `tab` information delimited:

```
place-id  woeid  latitude  longitude  place-name  place-type-id  place-url
```

Both `place-id` (A Flickr place identifier) and `woeid` (Where On Earth identifier) can uniquely identify a place on earth. The photo record only contains `place-id` information. Details about a place are stored in `place.txt`. Places can be specified at different levels, indicated by `place-type-id`. The valid `place-type-ids` in decreasing specificity order are:

- **22**: neighbourhood (roughly equivalent to place of interest)
- **7**: locality (roughly equivalent to city)
- **8**: region (roughly equivalent to state/province)
- **12**: country
- **29**: continent

`place-name` and `place-url` are alternative ways of naming a place. They both contain information about the parent locations of a place. For instance, a neighbourhood level (with `place-type-id` 22) place "Coogee" has a `place-url` "/Australia/NSW/Sydney/Coogee" and a `place-name` "Coogee, Sydney, NSW, AU, Australia".

Note that the data set may not have a consistent way of using acronyms in `place-name` and `place-url`. For instance, it may use either "NSW" or "New South Wales" to refer to the same region level place. You do not have to handle the mapping of acronyms to full names in this assignment. It is OK for you to treat them as different places.

The size of all input files is about 10GB. Carefully design the map/reduce tasks to minimize the number of scans of the original data set.

Analysis Task Descriptions

We are interested in getting some simple summary information of the data set. In particular, we want to find out:

1. Number of photos taken per **locality**.
   In the first task, we are interested in an explorative analysis of the Flikr data set. The data set consists of photos taken by users in different locations. Note that photos can be geotagged at various neighbourhood levels that belong to the same locality level. For instance, the locality level place name "Sydney, NSW, Australia" has 500+ neighbourhood level geotags associated with it such as "Coogee, Sydney, NSW, AU, Australia", "Circular Quay, Sydney, NSW, AU, Australia". You need to count all photos geotagged at locality level and neighbourhood levels belonging to the same locality. The output should be alphabetically sorted by locality name.

   The output file should have the following format, ordered by localityName:

   ```
   localityName \t numberOfPhotos
   ```

2. The top 50 **locality** level places based on the number of photos taken in this locality.
   In the second task, you shall extend your solution from Task 1 to order the output by descending order of number of photos per locality (localities with high number of photos first, then localities with a smaller number of photos taken there). Only return the top 50 localities with regard to number of photos taken at that locality.

   The output file should have the same format than in Task 1, but with a different order (by numberOfPhotos) and only the top-50 results.

3. **Ten most popular tags** for each of the top-50 localities.
   In the third task, for each top locality level place, we are also interested to find the most popular tags used for that place. The popularity of a tag is measured by the frequency of tags assigned to photos taken in this place. Note that it is common for the actual place name, the name of the parent place, and/or the year to be contained in the top tag list. You need to filter those out and output the remaining top 10 tags. For instance, the most popular tags assigned to "`London, UK`" include `london`, `uk` and `2008`, all of these should be filtered.

   The output file should have the following format (ordered descending by numberOfPhotos, with the tags per locality ordered by descending frequency):

   ```
   localityName \t numberOfPhotos \t [(tag1,freq1) (tag2,freq2) ... (tag10,freq10)]
   ```

Special Coding Requirements

1. It is possible to implement the above tasks using a sequential program. A sequential program would require loading all data in memory and organize them as a Hashmap or Treemap structure. You are asked to avoid using this type of approach either for the whole data set or for the partial data set. Instead, you should follow the divide-and-conquer principles of MapReduce framework to achieve most of the functions. Some tasks will require more than one jobs to produce the final result.

2. Always test your code using a small data set before applying it to the large one.

3. If you notice that majority of the execution time is spent on reduce phase, try to increase your number of reducers.

4. Do not copy the entire input data to your HDFS home directory or Linux home directory

Deliverable

There are three deliverables: **source code**, a brief **program design documentation** (up to 3 pages), and a **self-reflection survey**. All are due on Week 7 Friday 28th of April. There will be extra points allocated to be able to run your submission on the Azure platform. This has to be demoed to tutors by Week 8. Please submit the source code and a soft copy of design documentation as a zip or tar file in ELearning. The documentation should contain the following sections:

- **Job Design Documentation**
  In your document, describe the MapReduce jobs you use to implement Tasks 1 to 3. For each job, briefly describe the map and reduce functions. If you use a user-defined combiner, partitioner or key-type, please describe those too.

- Include as appendix the HDFS location of your final output files from various executions.

Self-Reflection Survey: As part of the submission, you are also asked to fill in a self-reflection survey, one per each team member. This survey will ask you to assess your experience with Hadoop MapReduce.
Azure Demo: A few points of the marking scheme (in the range of 10% to 20%) will be given to any submission which can be demoed successfully running on Windows Azure. Teams who are interested in getting these points should test their solutions early on Azure, and then have to demo their MapReduce programs by Week 8 to their tutor.