THE UNIVERSITY OF SYDNEY

School of Information Technologies
A/Prof Uwe Roehm

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

Week 5: MapReduce Programming: Join

06.04.2017

Question 1: Run Sample Hadoop Join Programs

Run the following commands on a cluster node to download `join-week5.tar`:

```
wget http://web.it.usyd.edu.au/~comp5349/2016/code/join-week5.tar
tar xvf join-week5.tar
cd week5
```

File `join-week5.tar` contains source code of several jobs. The ant build file `build.xml` will build an executable jar file `join.jar`. You may inspect the source code in Eclipse or use any command line editor.

The first job takes the `place.txt` as input and finds all places in a given country. The country name can be supplied as command line argument and passed to the mapper. This job is implemented by two classes: `PlaceFilterMapper` and `PlaceFilterDriver`. Both are in `join` package. It shows how to pass command line arguments to a mapper. It also shows how to initialize a mapper object using the `setup` method. This is a map only job. In the driver class, we set the number of reducer to 0. Use the following command to run the job:

```
hadoop jar join.jar join.PlaceFilterDriver /share/place.txt china-place China
```

As usual, the first argument specifies the input file; the second argument specifies the output directory. The third argument specifies a country name. If you do not supply the country name, the default one is "Australia". You will notice that the output file is named `part-m-00000`. The letter "m" indicates that the output is generated by a map task.

The second job is an example of replicated join. It takes the output of the place filter job and replicated it to all mappers. The mapper stores the replicated data in its memory and joins it with the input photo data file to produce records containing `photo-id`, `taken-time` and `place-name`. This job is implemented by two classes: `ReplicateJoinMapper` and `ReplicateJionDriver`. Both are in join package. The small place file is added in a `DistributedCache` in the driver class. All mappers then read the file from the DistributedCache during setup and store the content in a hash table. Run the job using this command:

```
hadoop jar join.jar join.ReplicateJoinDriver china-place/part-m-00000
/share/photo/n05.txt china-photo
```

Again, this is a mapper only job. Three mappers will be started because the input file n05.txt has three blocks. Each generates a result named from `part-m-00000` to `part-m-00002`.

A third driver is provided in `join` package to chain the above two jobs. This is implemented by `JobChainDriver` class. It submits the "place filter" job, waits for it to complete and stores the output in a temporary folder. It then submits the "join" job and writes the final result in the given folder. The temporary folder is then deleted. Run the chained jobs using the following command:

```
hadoop jar join.jar join.JobChainDriver /share/place.txt /share/photo/n01.txt
china-photo-chain China
```

Question 2: The Python Version

The Python version of the tutorial code (credit to Andrian Yang) can be downloaded using the following command:

```
wget http://web.it.usyd.edu.au/~comp5349/2016/code/py_join_week5.zip
unzip py_join_week5.zip
cd py_join_week5
```

As usual, a shell script is provided to demonstrate how to set various properties to submit the python job.

Question 3: Reduce Side Join

Package `reducesidejoin` (in python) contains sample code of reduce side join we discuss in the lecture. If you haven't already downloaded the Python code, run the job using the following commands and then inspect the reduce side join files:

```
wget http://web.it.usyd.edu.au/~comp5349/2016/code/py_join_week5.zip
unzip py_join_week5.zip
cd py_join_week5
```

Please pay special attention to how common data are passed to mapper script in the shell script.

```
hadoop jar join.jar reducesidejoin.JoinDriver -D mapreduce.job.reduces=5
/share/photo/n06.txt /share/place.txt n06-place
```

The argument `-D mapreduce.job.reduces=5` specifies the number of reducers you want to use. This is a more flexible option than hard code it in the driver class.

Question 4: Write Your Own Code

a) The Problem

Now write your own program to find users that have taken photos in two or more given countries. For instance, your program should be able to find out which users have taken photos in Australia and Germany and how many photos they have taken in each country. The input and output of your program would look like:

Input data:

photo data (use any of the `n0x.txt` in `/share/photo`)
place data (`/share/place.txt`)
countryNames(supplied as command line arguments)

Output data:

```
owner countryName1:photoNumber countryName2:photoNumber ...
```

b) Design Jobs and Tasks

The problem involves filtering relevant place-ids from a place data set; joining the result with a photo data set using place-id as the join key; and filtering the join results based on associated place information. We can reuse the distributed cache based join for the join phase and add a reducer to filter the join result.

Job 1: filtering relevant place-ids

`Mapper` (input: `place.txt`)

```
Map function: (lineOffset, place.txt line) -> (place-id, countryName)
```

Job2: join the filtered place data with photo data

`Mapper` (input: `n0x.txt`, distributed cache: filtered place data from job1)

The map function associates owner with country name

```
Map function: (lineOffset, n0x.txt line)-> (owner, countryName)
```

`Reducer`

The reduce function counts the number of times a countryNname appears in a owner's list. It only emits records when all country names are present

```
Reduce function: (owner, (countryName, countryName,...)) ->
(owner, countryName: count \t countryName:count)
```

c) Implementation

Job 1 is similar to the `PlaceFilter` job except that you need to handle more than one country name as command line arguments and the output is slightly different. Job2's mapper is similar to the mapper of `ReplicateJoin` job, the only difference is the output. You will need to write your own reducer for Job2.

The design in is only a guideline. You may use other ways to solve this problem.