CPS216 Data-Intensive Computing Systems - Fall 2011
Assignment 6 (Written)

- Total points = 100. Due date: Monday, Oct. 24, 2011 (5.00 PM).
- Submission: In class, or email solutions in pdf or plain text to the TA. You can also drop off the solutions at the TA's office.
- Do not forget to indicate your name on your submission.
- State all assumptions. For questions where descriptive solutions are required, you will be graded both on the correctness and clarity of your reasoning.

Question 1 **Points 20**

This question is based on the following Pig Latin query:

```
clicks = LOAD 'clicks' AS (userid, pageid, linkid, viewedat);
byuser = GROUP clicks BY userid;
result = FOREACH byuser {
     fltrd = FILTER clicks BY viewedat IS NOT NULL;
     uniqPages = DISTINCT fltrd.pageid;
     uniqLinks = DISTINCT fltrd.linkid;
     GENERATE group, COUNT(uniqPages), COUNT(uniqLinks);
};
```

(I) Draw a valid logical plan, physical plan, and MapReduce plan for this query.

(II) Give a valid set of example bags for each of the variables in this Pig Latin query. Note that the variables in this Pig Latin query are: clicks, byuser, fltrd, uniqPages, uniqLinks, and result. See Figure 4 in the "Pig Latin: A Not-So-Foreign Language for Data Processing" paper on how to present your answer.

(III) Is it correct to move the FILTER statement to before the GROUP statement? That is, will the following Pig Latin query produce the same final result as the above query:

```
clicks = LOAD 'clicks' AS (userid, pageid, linkid, viewedat);
fltrd = FILTER clicks BY viewedat IS NOT NULL;
byuser = GROUP fltrd BY userid;
result = FOREACH byuser {
     uniqPages = DISTINCT fltrd.pageid;
     uniqLinks = DISTINCT fltrd.linkid;
     GENERATE group, COUNT(uniqPages), COUNT(uniqLinks);
};
```

Question 2 **Points 20**

This question is based on the following Pig Latin query:

```
urls = LOAD 'dataset' AS (url:chararray, category:chararray, pagerank:double);
groups = GROUP urls BY category;
bigGroups = FILTER groups BY COUNT(urls) > 50000000;
result = FOREACH bigGroups GENERATE group, get_url_with_max_pagerank(urls);
STORE result INTO 'myOutput';
```

We are given a dataset of urls. Each url is a tuple containing three attributes: "url", "category", and "pagerank". The above Pig Latin query outputs, for each sufficiently large category of urls, the url as well as pagerank for the url with the maximum value of pagerank in that category. get_url_with_max_pagerank() is a user-defined-function (UDF) that takes a bag of urls as input, and returns the url and pagerank for the url with the maximum value of pagerank in that bag.

The dataset of urls is heavily skewed with the values of the attribute "category" having a *power law distribution.*[1] That is, there are some categories that contain a very large number of urls. The largest category contains 100 Million urls. Most other categories contain few urls. The average size of a url tuple is 50 bytes.

You are asked to answer the following questions:

(I) Draw a valid MapReduce plan for this query.

(II) Give an estimate of the maximum amount of memory that a map or reduce task in the MapReduce plan you proposed may need for this query. Explain the reasoning behind your estimate for full credit.

(III) If you run this MapReduce plan on the m1.small nodes on Amazon EC2 using the harness that we provided, what behavior would you expect? Explain your reasoning for full credit.

(IV) Suppose you observe that the MapReduce plan keeps crashing due to insufficient memory. You decide to rewrite the query as one or more Java MapReduce programs in a way that your Map and Reduce tasks will need as little memory as possible to run to completion. Briefly explain how you will achieve this. You only need to explain the main ideas. There is no need to write the actual MapReduce program(s).

Question 3 **Points 20**

Figure 7 in the "Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience" paper shows an innovative idea in Pig. Here, Pig is using one single MapReduce job to process two very different Group operations. One Group operation groups on the "pageid" attribute and the other Group operation groups on the "linkid" attribute. Explain in your own words how these two group operations can be processed using a single MapReduce job instead of two separate MapReduce jobs.

Question 4 **Points 20**

This question is taken from the following Pig Latin query given at
http://developer.yahoo.com/blogs/hadoop/posts/2010/01/comparing_pig_latin_and_sql_fo

```
Users                = LOAD 'users' AS (name, age, ipaddr);
Clicks               = LOAD 'clicks' AS (user, url, value);
ValuableClicks       = FILTER Clicks BY value > 0;
UserClicks           = JOIN Users BY name, ValuableClicks BY user;
Geoinfo              = LOAD 'geoinfo' AS (ipaddr, dma);
UserGeo              = JOIN UserClicks BY ipaddr, Geoinfo BY ipaddr;
ByDMA                = GROUP UserGeo BY dma;
ValuableClicksPerDMA = FOREACH ByDMA GENERATE group, COUNT(UserGeo);
STORE ValuableClicksPerDMA INTO 'ValuableClicksPerDMA';
```

[1] If you want to learn more about the power law distribution, see http://en.wikipedia.org/wiki/Power_law

Draw a valid logical plan, physical plan, and MapReduce plan for this query.

Question 5 **Points 20**

This question is also taken from a Pig Latin query given at
http://developer.yahoo.com/blogs/hadoop/posts/2010/01/comparing_pig_latin_and_sql_fo

```
Users         = LOAD 'users' AS (name, age, gender, zip);
Purchases     = LOAD 'purchases' AS (user, purchase_price);
UserPurchases = JOIN Users BY name, Purchases BY user;
GeoGroup      = GROUP UserPurchases BY zip;
GeoPurchase   = FOREACH GeoGroup GENERATE group, SUM(UserPurchases.purchase_price) AS sum;
ValuableGeos  = FILTER GeoPurchase BY sum > 1000000;
STORE ValuableGeos INTO 'byzip';
DemoGroup     = GROUP UserPurchases BY (age, gender);
DemoPurchases = FOREACH DemoGroup GENERATE group, SUM(UserPurchases.purchase_price) AS sum;
ValuableDemos = FILTER DemoPurchases BY sum > 100000000;
STORE ValuableDemos INTO 'byagegender';
```

Draw a valid logical plan, physical plan, and an efficient MapReduce plan for this query. (*Hint: note the web site's claim that this query can be done using two MapReduce jobs.)*