CS6220 BDS

# Homework Assignment 4
## (Programming Category)

Student Name: Pedro Henrique Ribeiro Pinto

Student Session: cs6220-A

In this 4th programming assignment, you are given **five** types of programming problems. The first and the fourth problem are designed for those of you who are interested in performance of big data systems (Key-value or Graph Processing). The second one is about frequent pattern mining. The 3rd problem is designed for you to gain some hand-on experience with deep learning framework through accuracy tuning or comparison and you can reuse the models trained in your previous HWs. The 5th (last) one is about hand-on experience with GAN.

For the problems that have more than one options, you only need to choose one option (subproblem) for the 4th assignment.

For those of you who wish to choose a programming option from previous assignments, you may choose any option that you have not done in HW1 ~ HW3, provided that you need to provide a summary of the option you did in the previous 3 assignments.

This is the last assignment for the course.

**Post Date**: Sunday of Week 10 (Oct. 20)
**Due Date**: Midnight on Friday of Week 11 (Nov. 1 with no penalty grace period until 9am on Saturday of Nov. 2)

## Problem 1. Hand-on Experience with a Key-Value Store System

**Problem 1.1 Performance Measurement of your favorite Key-Value Store**

- Here are some example Key-Value stores, such as Redis (http://redis.io), Hbase (http://hbase.apache.org), RocksDB (http://rocksdb.org), LevelDB (http://leveldb.org), Mongo DB (mongodb.com) and so forth. You can also choose any one of the Key-Value stores of your interest as long as it is an open source implementation.

- You are asked to use some workload benchmarks, such as YCSB (https://github.com/brianfrankcooper/YCSB), to run four sets of experiments: Bulk Loading of dataset to the key-value store, Read only or read most workloads, write only or write most workload, and read and write mixed workloads. You may also use your own key-value datasets.

- You are asked to conduct performance evaluation of one key-value system by using 4-5 different scales of datasets, such that you can compare the key-value

system performance when the dataset size is increasing from, say 100, 1000, 2000, 4000, 6000.

- **Deliverable**: You can choose to conduct performance evaluation of one key-value system by using two different datasets or conduct performance comparison of two competing key-value systems on at least one dataset. Your deliverable should include the performance measurement results, the experience with the installation, running and measurement. You can provide your deliverable in ppt or word or pdf.

**Problem 1.2 Performance Comparison of two Key-Value Systems**

1. You are asked to download and install 2-3 key-value stores,Here are some example Key-Value stores, such as Redis (http://redis.io), Hbase (http://hbase.apache.org), RocksDB (http://rocksdb.org), LevelDB (http://leveldb.org), and so forth. You can also choose the Key-Value stores of your interest, which have an open source implementation.

2. You are asked to download and install two key-value stores. You can populate the key-value store you created using a workload benchmark, such as YCSB (https://github.com/brianfrankcooper/YCSB).

3. You are asked to conduct performance comparison study with four sets of experiments: Bulk Loading of dataset to the key-value store, Read only or read most workloads, write only or write most workload, and read and write mixed workloads. For this set of comparisons, you are asked to use the same dataset of at least 1000 records. You can also choose any one of the Key-Value stores of your interest or choose some other one that you are familiar with and has an open source implementation.

4. **Deliverable**: Your deliverable should include the performance measurement results, the experience with the installation, running and measurement, and your analysis and discussion on the results. You can provide your deliverable in ppt or word or pdf.

## Problem 2. Hand-on Experience with Frequent Pattern Mining Algorithms

### Problem 2.1 Implementing and Evaluating a Frequent Pattern Mining Algorithm

Your task for this problem is to implement and evaluate the Apriori-based algorithm for frequent itemsets mining, originally proposed by Agrawal et al. for frequent itemsets mining (see http://rakesh.agrawal-family.com/papers/vldb94apriori.pdf ). You can find the pseudocode and its related procedures from course note or download the open source Apriori code you can find. You may use any programming language that you are familiar with.

The program should be executable with 3 parameters: the name of the input dataset file, the threshold of minimum support count, and the name of the output file. The minimum

support count should be an integer. An itemset is frequent if its support count is larger or equal to this threshold. The program should output a file that contains all the frequent itemsets together with their support. The output file should have the following format: each line contains a single frequent itemset as a list of items separated by whitespace. At the end of the line, its support is printed between a pair of parenthesis. For example: A B C (5) represents an itemset containing items A, B and C with a support count of 5.

You are encouraged to use existing or your own optimization techniques for the Apriori algorithm. If you do, explain and discuss the techniques you have used and/or provide the appropriate references in the report.

Test your implementation on a dataset. You can choose one of the frequent itemset mining datasets at http://fimi.ua.ac.be/data/, or choose from WEAK, R or Mahout website. Similarly, there are a few online repositories for frequent pattern mining implementations, such as FIMI repository (http://fimi.ua.ac.be/src/) or WEKA, R, Mahout. You can extend/modify the source code of your choice to add the optimization.

**Deliverable.**
Write a brief report in PDF or word to present your results on the test dataset and other datasets if you have experimented with. Explain and discuss the results. If the algorithmic optimizations are used in your implementation or incorporated into an existing open source algorithm you used for evaluation and comparison, discuss the experiences and lessons you have learned from the implementation.

Your submission should be a zip or tar file that contains the PDF or word report as well as the program deliverables, including your source files, the executable, a readme file explaining how to compile/run your program, the output file for the test dataset, and the setting of your minimum support for the results.

**Problem 2.2 Comparing two different frequent pattern mining algorithms**

Your task for this assignment is to compare and evaluate two different implementations of the same algorithm or two different algorithms for frequent pattern mining.

There are a few online repositories for frequent pattern mining implementations, including WEKA, R, Mahout and FIMI repository (http://fimi.ua.ac.be/src/). You can study them and make your selection on the algorithms.

You are asked to choose two algorithms from open source packages, such as FIMI repository (http://fimi.ua.ac.be/src/) or WEKA, R, Spark MLib, Hadoop Mahout, and run the two algorithms on two datasets of different sizes and compare their performance and illustrate the comparison results. You can find the datasets from frequent itemset mining datasets at http://fimi.ua.ac.be/data/. The two algorithms can be two alternative implementations of the Apriori algorithm or comparing Apriori algorithm with FP-Growth algorithm. The comparison results performed on two datasets of different sizes should allow you to comment on the scalability of the algorithms.

**Deliverable:**

Write a brief report in PDF or word to present your results on the test dataset and other datasets if you have experimented with. Explain and discuss the results.

Your submission should be a zip or tar file that contains the PDF or word report as well as the program deliverables, including your source files, the executable, a readme file explaining how to compile/run your program, the output file for the test dataset, and the setting of your minimum support for the results.

## Problem 3. Deep Learning Framework: Accuracy Tuning or Comparison

## Option 3.1 Accuracy Tuning of a deep learning framework

1. You are asked to choose one of your favorite deep learning frameworks, such as TensorFlow, Caffe, Torch, Theano, AlexNet, Deeplearn4j, Theano or the deep learning framework of your choice, as long as it has an open source implementation.

2. You are asked to use some benchmark datasets, such as MINIST, CIFAR10, CIFAR100, ImageNet1K, to generate your training dataset and your testing (validation) dataset. You should choose at least 1000 images with K (>=2) classes from the above datasets or your own preferred dataset with 8:2 or 9:1 ratio for training and testing.

3. You are asked to conduct accuracy evaluation study by tuning at least three hyperparameters, such as

   (i) The training dataset size;
   (ii) The neuron size;
   (iii) The number of weight filters;
   (iv) The number of mini-batches over the training dataset;
   (v)The number of training epochs;
   (vi) The # of convolutional layers;
   (vi) The Learning Rate Policy;
   (vii) The accuracy of classification with varying # of hidden layers but the fixed # of training epochs.

   Note that you are asked to measure accuracy by varying each metric (e.g., varying the number of epochs for at least 3~5 different settings on X-axis and measure/display accuracy measure on Y-axis.

4. You can use any open source tools for tuning hyperparameters, such as GridSearch in Scikit learn, GTDLBench and LRBench. Cross validation is recommended (though optional).

5. **Deliverable**: You are asked to report the accuracy evaluation of the deep learning framework of your choice on at least one dataset in tabular format or excel plots. Your deliverable should include the accuracy measurement results, the experience with this HW4, and your analysis and discussion. You can provide your deliverable in ppt or word or pdf.

## Option 3.2. Accuracy Comparison of two deep learning frameworks

1. You are asked to download and install two deep learning frameworks, one is TensorFlow, and the other one can be either Caffe, Torch, Deeplearn4j, Theano or any other deep learning framework of your interest, as long as both have an open source implementation.

2. This option is an advanced version of the option 3.1. On top of the option 3.1, you are asked to conduct accuracy comparison study for two models trained on the same dataset but using two different software frameworks, such as TensorFlow and PyTorch.

3. You are encouraged to use cross validation in your study.

4. **Deliverable**: You are asked to report the accuracy and training/testing time comparison of the two deep learning frameworks on at least one dataset in tabular format or excel plots. Your deliverable should include the performance measurement results, the experience with the installation, running and measurement, and your analysis and discussion. You can provide your deliverable in ppt or word or pdf.

## Problem 4. Distributed Graph Processing Systems

- You are encouraged to download a distributed graph processing system and run at least one graph computation algorithm such as PageRank or SSSP (single source shortest path) or CC (connected component) on two graph datasets.

- Example Distributed Graph Processing Systems you can download and play with include: Apache Giraph from http://giraph.apache.org; Apache Hama from https://hama.apache.org ; GPS from http://infolab.stanford.edu/gps/ ; distributed GraphLab (PowerGraph) from https://github.com/dato-code/PowerGraph, GraphX from Spark.

- Example graph datasets can be found at several websites, including:
  https://snap.stanford.edu/data/
  http://law.di.unimi.it/datasets.php
  http://www.eecs.wsu.edu/mgd/gdb.html

- All the above distributed graph processing systems are running on top of Hadoop File System (HDFS) and their vertex centric API will include example code of PageRank, SSSP or CC, which you can use directly for your experiments.

- **Deliverable**:
  - You can choose to conduct performance evaluation of one big data system by using two different datasets or
  - You can choose to conduct performance comparison of two competing big data systems on at least one dataset.

- Your deliverable should include the performance measurement results, the experience with the installation, running and measurement. You can provide your deliverable in ppt or word or pdf.

## Problem 5. Hand on experience with Generative Adversarial Network

In 2020 Fall semester, we could not include the lecture on GAN in the schedule.

This programming homework is included here for those students who have learned the concept of **generative adversarial network** (**GAN**), which is easy to understand, and would like to get hand-on experience in this HW to solve a learning task.

You are free to choose a training set of your interest, and use the GAN technique to generate new data with the same statistics as the training set.

For example, a GAN trained on photographs or paintings can generate new photographs or paintings that look at least superficially authentic to human observers, because the GAN generated photos or paintings have many realistic characteristics. You are free to choose other types of learning tasks, such as unsupervised learning, semi-supervised learning, fully supervised learning, or reinforcement learning.

Deliverable.

1. You are asked to provide the URL where you download your GAN installation.
2. You are asked to provide the detailed description of your dataset with 5 examples of your dataset. Also provide the URL of your dataset.
3. Define the problem that you wish to use GAN to solve.
4. Describe how you use GA in solving your problem, the procedural steps and why your approach is good.
5. Provide 2-3 measurements results to show the training effectiveness and the prediction results.
6. Discuss the lessons learned from this homework.