Project 2 – Csci 244 Fall 2019 - Operating Systems

K-nearest neighbor with multi-threading in C++

Goal

The goal of this project is to monitor the evolution of the characteristics of processes over time.

Rationale

The k-nearest neighbor (knn) algorithm is a density based and memory based pattern recognition technique to retrieve the label of a test example corresponding to the label of the closest example from a dataset that contains labelled examples. The algorithm can be easily found online (cite the reference of the paper you have checked in the short report).

We consider a 2 datasets: DBTrain and DBTest

DBTrain is of size Ntrain X D and DBTest is of size Ntest X D where Ntrain, Ntest, and D are the number of examples in the Training dataset, the number of examples in the Test dataset, the dimension of each example, respectively. D corresponds to the number of features. If you have an image of size 10 x 10, D=100.

The task

The goal is to implement the algorithm by considering specific values of NTrain, NTest, and D that are small enough to run on your computer and large enough to provide relevant results.

The content for DBTrain and DBTest can be generated randomly, or it can be obtained through some state of the art database such as MNIST that you can download at this address: http://yann.lecun.com/exdb/mnist/ (Ntrain=60000, Ntest=10000, and D=28*28.)

The goal is to implement knn by using multithreading to cut DBTrain into Nthread equal parts.

For the implementation, you will have to consider 3 types of implementations: Pthreads, thread <thread>, and asynch <future>. In the code, use a #DEFINE TYPE_IMPLEMENTATION to determine what implementation is used, by considering directives of precompilation.

For running the code, we want to estimate the running time by manipulating the following conditions:

- The type of implementation (3 conditions)
- The number of threads (1-8)
- The priority of the threads (select 3 different priorities) – For the best implementation only.

To obtain reliable measurements, you need to have multiple examples, i.e. multiple runs and then estimate the average running time with its standard deviation to report the results.

It will lead to:

- A figure or table with: 3*8 measurements. (implementation x #threads)
- A figure or table with: 3*8 measurements. (priority x #threads)

Deliverables

- The code (2)
  - It has comments. (0.4)
  - It is elegant. (0.4)
  - It compiles. (0.4)
  - It runs. (0.4)
  - It displays appropriate messages in the console to determine what is happening. (0.4)
- Source (7)
  - Function to load real data (BONUS) +1
  - Function to generate random number (1)
  - Function for the knn algorithm (2)
  - The multithreaded aspect for
    - PThread (1)
    - Thread (1)
    - Asynch (1)
  - Function to analyze the time (0.5)
  - Function to run the simulations (0.5)
- Short report (6)
  1. Short introduction about the goal of the project
  2. Brief description of the methods
  3. A table/figure for Implementation vs. #Thread
  4. A table/figure for Priority vs. # Thread
  5. Discussion about the results with some statistical analysis
  6. Personal reflection about the obtained results
- Demo (1)