ELEN4020A: Data Intensive Computing
Laboratory Exercise No 1

To Be Completed By: 11:59Hrs February 25th, 2019

Outcome

This is a group assignment. We expect to familiarise yourself, learn and understand the use of the following:

i.) The Linux (Ubuntu 18.04.X/16.04.X) programming environment. You have a choice of:

- the use the D-Lab resources but boot into Linux,
- Ubuntu 16.04/18.08 (installed on your personal laptop by dual booting if your main environment is Windows 10)
- Cygwin within Windows 10, if you do not like dual booting.

ii.) The use of one or more of the following editors – vi/vim/gvim, gedit, atom or emacs;

iii.) How to run and compile programs with your favourite compilers; a choice of one or more of gcc-6.4.0/7.3/8.2, python-3.6.X

iv.) How to use the gcc ompiler in a Unix environment to compile and run a sample program successfully that uses the PThread library, e.g., adding ”-lpthread”

v.) How to use the gcc compiler in a Unix environment to compile and run a sample program successfully that uses the OpenMP library, i.e. adding ”-lopenmp -lgomp.” See the sample text below for a “hellow_world.c” code.

```
#include <stdlib.h>
#include <stdio.h>
#include "omp.h"
int main() {
#pragma omp parallel
    {
        printf("hello world \n");
    }
}
/*
    To compile
    % gcc -fopenmp -lgomp hello_world.cc -o hello_world

    To run use:
    export OMP_NUM_THREADS=8 # <number of threads to use>
    % ./hello_world
*/
```

vi.) How to write a pseudo-code for relevant modules of your programs. We recommend the use of *Algorithm2e.sty* for your pseudo-code writing [1].

vii.) How to use GIT and setup a group project account on GitHub.

viii.) **Please include a documentation branch for your reports** in GitHub. Your reports are expected to be written in **LaTeX**. Do not play the trick of writing your documents in MSWord and then checking in only the pdf files of the documentation. I'll be *cloning* your entire project directory for marking.

Please keep in mind that this laboratory exercise and the subsequent ones are intended to prepare you for your projects — mainly writing software/applications that will involve parallel execution, parallel I/O, big data and machine learning concepts. You will be expected to follow-up this exercise here with learning to use other interesting tools: HTML, CSS, Javascript, XML, JSon, SQL (BerkeleyDB, SQLite, PostgreSQL, MariaDB, etc), Jupyter, R, Julia, etc.

Please ensure that you save copies of your laboratory assignments on a **pen-drive** or **flash-drive** as backups **PLEASE CHECK OFTEN THAT YOUR PEN DRIVES ARE VIRUS FREE, and have the most up-to-date backups.**

Work Schedule

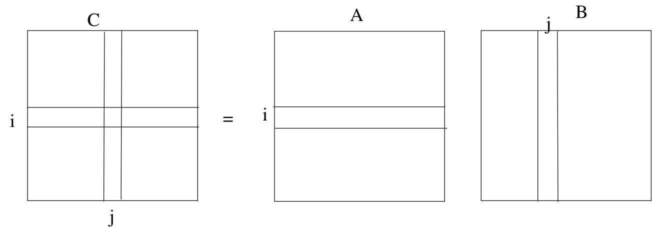
Unix Tutorial for Beginners

If you are not already familiar with Unix then consult any Unix tutorial site and drill yourself on how to use this environment. Familiarise yourself with basic commands such as: "ls/ll, cd, mkdir, pwd, cat, less, etc."

Problem Description: 3-D by 3-D Array Multiplication also referred to as rank 3 Tensor Contraction

The final goal is to develop an algorithm that computes the array multiplications $C = A \times B$ where C, A, and B are 3-dimensional arrays. Such a problem has its origin in machine/deep-learning. The multiplication of arrays A and B when these are 2-dimensional (i.e., matrices) use the well known matrix multiplication algorithm.

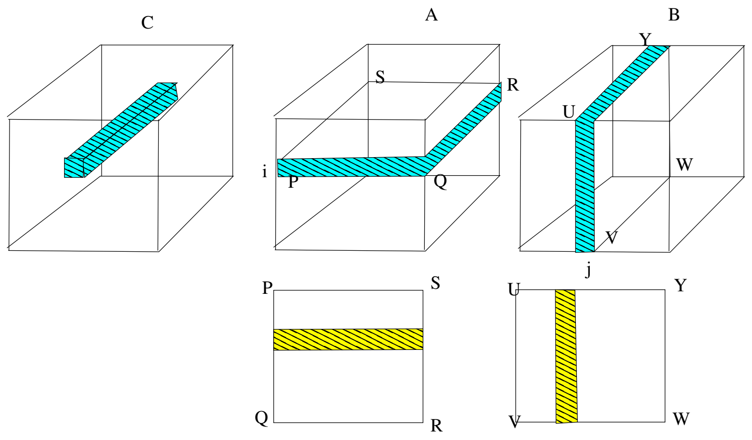
Figure 1: 2-Dimensional Matrix Multiplication



$$c_{ij} = \sum_k a_{ik} \times b_{kj}$$

In computing the multiplication of a 3-D array, first consider the row of a array that should be multiplied by a column of matrix be as a row-plane and and column-plane. These two, for a row index i in A and a column-index j in B give you now 2 2-dimensional matrices that can be computed with the traditional approach of rows and columns.

Figure 2: 3-Dimensional Array Multiplication



You are required to write 4 procedures; 2 for 2-dimensional matrices and the other two for 3-dimensional arrays. Lets call these rank2TensorAdd(), rank2TensorMult(), rank3TensorAdd() and rank3TensorMult(). Set the arrays bounds to be N, for N = 10 and then 20. The procedures rank2TensorAdd() and rank3TensorAdd() simply compute the respective result C, using element by element additions. The rank2TensorMult() computes the multiplication of 2 2-dimensional matrices. The rank3TensorMult() computes the multiplication of 2 3-dimensional arrays based on the computational model illustrated in the diagram above. Choose a strategy for passing the parameters and arguments to the procedures/functions that you code. Do not forget to include a method of reporting an error condition. For example your 3-dimensional arrays will be something like A[N][N][N], B[N][N][N] and the result of your computation will be stored in C[N][N][N] for N = 10, 20.

You are then to write a main program, that dynamically generates, in turn, the elements of the input arrays where each element is a random number in [0, 20]. For each pair of arrays generated, first for 2-dimensional matrices, then for 3-dimensional arrays, it calls the procedures to compute the appropriate C result matrix.

The Deliverable

- Submit your codes, for marking in your group's initialised Git repository on GitHub.
- Provide a short description of your algorithm to the problem of 3-D array multiplication and also the pseudo-code of your routines.
- Write a short set of instructions on how to access your GitHub repository and send this to Sakai. This should be submitted under the name/id of your lead member of your group.

References

[1] LaTeX/Algorithms, " LaTeX Wikibooks"