



## Laboratory Exercise 1

### 1 Outcome

This is a group laboratory exercise. You are expected to familiarise yourself, learn and understand the use of the following:

1. Linux (Ubuntu) and programming within this environment. You have a choice of:
  - the use the Computer Laboratory resources in the Chamber of Mines Building but boot into Linux.
  - Ubuntu installed on your personal laptop by dual booting if your main environment is Windows.
  - the Windows Subsystem for Linux (WSL2) running the Ubuntu flavour.
  - a virtual machine/computer running Linux within Windows, if you do not like dual booting.
2. The use of one or more of the following editors – vi/vim/gvim, gedit, atom, or VSCode.
3. How to run and compile programs with your favourite compiler(s)/interpreter(s) with an industry standard build-tool. The list of approved compiler(s), interpreter(s), and build-tools are listed in the software guide document on the course home page (Sakai).
4. How to use the gcc compiler in a Unix environment to compile and run a sample program successfully that uses the PThread library, e.g., adding `-lpthread`
5. 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 code below for a `hello_world.c` example.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "omp.h"
4
5 int main()
6 {
7     #pragma omp parallel
8     {
9         printf("hello_world\n");
10    }
11 }
12 /*
13 To compile
14 % gcc -fopenmp -lgomp hello_world.c -o hello_world
15 To run use:
16 export OMP_NUM_THREADS=8 # <number of threads to use>
17 % ./hello_world
18 */
```

6. How to write pseudo-code for relevant modules of your programs. The  $\LaTeX$  package `Algorithm2e.sty` is recommended for your pseudo-code writing.

7. How to use `git` and setup a group project account on GitHub.
8. Your reports must follow the “Blue Book” guidelines and are regarded as lab-reports not project reports (i.e. no abstract or conclusion etc.). Your entire project directory will be cloned for marking and assessment. Report submission is through Sakai whereas source-code is submitted through the course Github Organisation.
9. Your source-code must include a README explaining which build-tool is used and how to compile and run your program. Make sure to follow the software guide document.

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, and big data concepts. 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.**

## 2 Work Schedule

### 2.1 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.

### 2.2 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.

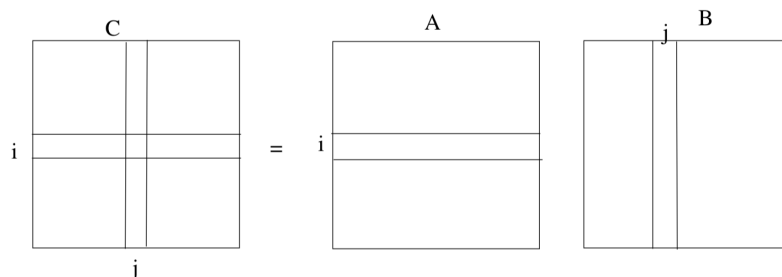
$$C_{ij} = \sum_k A_{ik} \times B_{kj}$$


Figure 1: 2-Dimensional Matrix Multiplication

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 two 2-dimensional matrices that can be computed with the traditional approach of rows and columns.

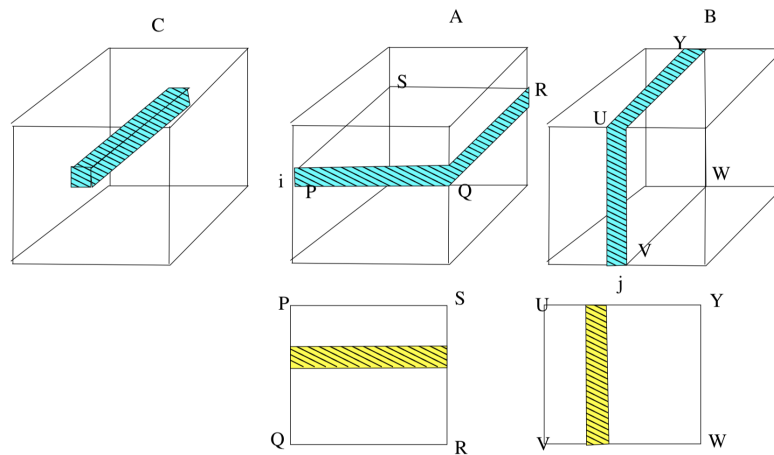


Figure 2: 3-Dimensional Array Multiplication

You are required to write two procedures using two parallelisation methods (PThreads and OpenMP); one for 2-dimensional matrices and the other for 3-dimensional arrays. Let's call these `rank2TensorMultPThread()`, `rank3TensorMultPThread()`, `rank2TensorMultOpenMP()` and `rank3TensorMultOpenMP()`. Set the arrays bounds to be  $N$ , for  $N = 10, 20$ , and  $30$ . `rank2TensorMultPThread()` computes the 2D matrix multiplication of two square matrices using PThreads as the parallelisation method. `rank2TensorMultOpenMP()` computes the same result but uses OpenMP to manage the parallelisation. `rank3TensorMultPThread()` and `rank3TensorMultOpenMP()` are the three-dimensional analogues of the rank 2 procedures that use the above-mentioned 3D matrix multiplication scheme. 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  $N = 10, 20$ , and  $30$ .

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 the range  $[0, 20]$  and the bounds are  $N$ . For each value of  $N$  ( $N = 10, 20, 30$ ) and for both two and three dimensions, use your program to generate appropriate random matrices and then compute the result  $C$  using your already implemented procedures. Use the Unix command `time` to record the execution time for all combinations.

### 3 Deliverables

- A lab-report analysing the performance of the algorithms as  $N$  increases and the number of dimensions changes. Max 2 pages and one report per group.
  - Provide a short description of your algorithm to the problem of 3-D array multiplication and also the pseudo-code of your routines.
- Your code for marking in your initialised group `git` repository in the Github Organisation.
  - Ensure your repository has an appropriate README describing the build process.