

In Assignment 1, you have designed an efficient parallel algorithm to solve the **all pairwise computation** problem on shared-memory computing platforms and implemented the algorithm using Pthreads. In **Assignment 2**, you are asked to design an efficient parallel algorithm to solve the same problem on a **distributed-memory machine**. In this distributed-memory machine, there are a number of processing nodes and each processing node consists of a number of cores with shared memory. Thus, in your algorithm design, you need to consider data/tasks distribution and necessary communication among the processing nodes; you also need to consider how you will assign tasks to different cores in each processing node. After you have designed your parallel algorithm, you will implement it using **MPI** (for data communication between processing nodes) and **Pthreads** (for thread management and synchronization in each processing node). The all pairwise computation problem is presented again below.

All Pairwise Computation is defined as performing computation (e.g., correlations) between every pair of the elements in a given dataset. In this assignment, we consider a set of N sequences, each being of length M , which are stored in a two-dimensional (2D) matrix of size N by M (i.e., N rows and M columns). We then calculate a dot product for every possible pair of sequences (row vectors) in the matrix.

A dot product of two sequences, or vectors is defined as

$$x \cdot y = \sum_{i=0}^{M-1} x_i y_i$$

If we allow a vector to pair with itself, then there are $N(N+1)/2$ such pairs in total for N vectors. Assume N is equal to 5 for example. We then have the following 15 vector pairs:

(0, 0)(0, 1)(0, 2)(0, 3)(0, 4)

(1, 1)(1, 2)(1, 3)(1, 4)

(2, 2)(2, 3)(2, 4)

(3, 3)(3, 4)

(4, 4)

In the above dot product formula, (i, j) denotes the pair of sequences i and j . After the dot product computation for pair (i, j) , we will have a single value as the output. Let a_i and a_j be two sequences, namely the i^{th} and j^{th} row vectors in the 2D matrix, and $c_{ij} = a_i \cdot a_j$. We can store the computational results in a one-dimensional (1D) array of size $N(N+1)/2$ in a row major order:

c_{00}	c_{01}	c_{02}	c_{03}	c_{04}	c_{11}	c_{12}	c_{13}	c_{14}	c_{22}	c_{23}	c_{24}	c_{33}	c_{34}	c_{44}
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

In your parallel algorithm design, you **must** consider how to balance the workload, minimize communication overhead, and use the loop unrolling technique to improve the performance.

In your parallel algorithm implementation:

- Your program needs to ask for N (number of sequences) M (sequence length) and T (number of threads in each processing node) as user defined parameters (and block size B if necessary);
- The results must be placed in a 1D array in the row major order as shown in the above example; and
- After the parallel computation, your main program must conduct a self-check for correctness; i.e., perform a sequential computation using the same data set and then compare the result of this computation with the parallel computation result.

Your implementation must be in C. It must use MPI and Pthreads as described earlier in this assignment description.

You **must** write a **report**. The report must be concise, clear (3-6 A4 pages) and contain the following sections:

1. Problem definition and requirements
2. Parallel algorithm design and implementation
3. Testing
4. Discussion
5. Known issues in program
6. Manual (e.g. how to run the program, input and output)

Your assignment will be marked on the efficiency of your algorithm, program logic and readability, accuracy of results, and quality of your report.

You **MUST** attempt this assignment **individually**.

Submission Requirements

1. Your submission must be made by **11:59pm on Friday, 27 May, 2022 (Sydney time)**.
2. Create a **tar or zip file** that contains **your report**, *makefile* and **source files** (e.g., **.c** and **.h files**). **DO NOT INCLUDE ANY OBJECT OR BINARY FILES.**
3. Submit only one **.tar** or **.zip file**.

Failure to follow these submission requirements may lead to loss of marks.