KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 1 : JAVA Programming - A Revision

Objective

- To write simple sequential programs using Java.

Lab Activities (see the hints below)

1. Write, compile and run the "`hello, world`" program.
2. Write, compile and run a program which does a vector addition.
3. Write, compile and run a program which does a matrix multiplication.
4. Write, compile and run a program which does the pi computation.

Exercises

1. Write a program that does the following:
   (a) Compute the average of array elements.
   (b) Sum two numbers, stored in a and b.
2. Write a program to find the max number in the given array.

Hints:

1. Vector addition:

$$\begin{aligned}\mathbf{x}+\mathbf{y} &= (x_0, x_1, \ldots, x_{n-1}) + (y_0, y_1, \ldots, y_{n-1}) \\ &= (x_0 + y_0, x_1 + y_1, \ldots, x_{n-1} + y_{n-1}) \\ &= (z_0, z_1, \ldots, z_{n-1}) \\ &= \mathbf{z}\end{aligned}$$

2. Matrix multiplication:

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \vdots & \vdots & & \vdots \\ a_{i0} & a_{i1} & \cdots & a_{i,n-1} \\ \vdots & \vdots & & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_i = a_{i0}x_0 + a_{i1}x_1 + \cdots a_{i,n-1}x_{n-1} \\ \vdots \\ y_{m-1} \end{bmatrix}$$

Matrix-vector multiplication

3. Pi computation:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^n \frac{1}{2n+1} + \cdots\right).$$

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 2: JAVA Multithreading and Parallel Programming

Objective

- To learn about threads creation, tasks and pools in JAVA.

Lab Activities (see the hints below)

1. Write, compile and run the "`hello, world`" program (using Thread class).
   Here is a sample output:

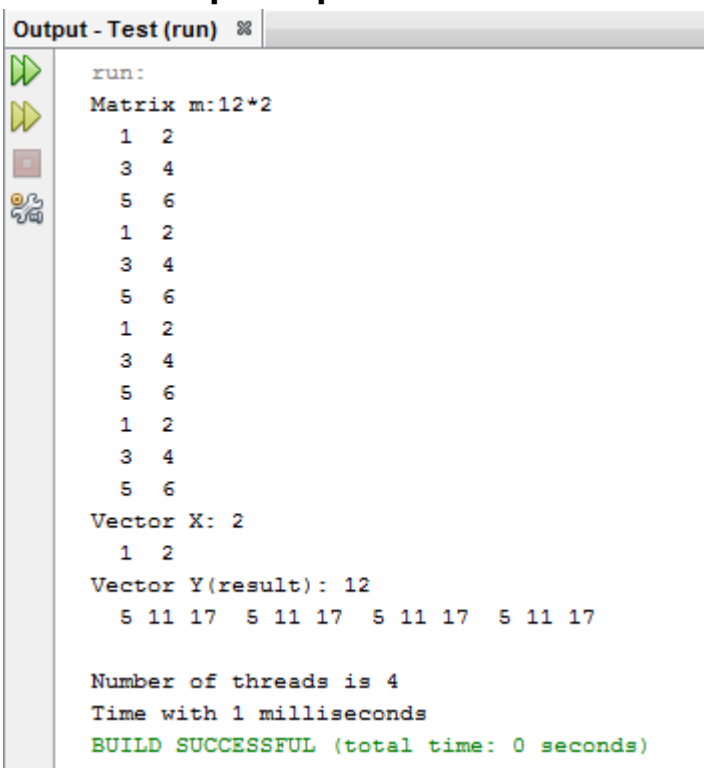
```
Output - Test (run)
run:

Hello from thread 1 out of 2
Hello from thread 0 out of 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Write, compile and run a program which does a vector addition (first, using Thread class and then using Executer).
   Here is a sample output:

```
Output - Test (run)
run:
x:          1  2  3  4  5  6
y:          7  8  9 10 11 12
z(x+y):     8 10 12 14 16 18
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Write, compile and run a program which does a matrix multiplication (using Thread class).

Here is a sample output:



Exercises

1. Write a parallel program that prints 1000 random numbers.
2. Rewrite lab activity (2) so that the input is entered by the user.
3. Rewrite lab activity (3) using Executer.

Hints:

1. Vector addition:

$$\begin{aligned}\mathbf{x}+\mathbf{y} &= (x_0, x_1, \ldots, x_{n-1}) + (y_0, y_1, \ldots, y_{n-1}) \\ &= (x_0 + y_0, x_1 + y_1, \ldots, x_{n-1} + y_{n-1}) \\ &= (z_0, z_1, \ldots, z_{n-1}) \\ &= \mathbf{z}\end{aligned}$$

2. Matrix multiplication:

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \vdots & \vdots & & \vdots \\ a_{i0} & a_{i1} & \cdots & a_{i,n-1} \\ \vdots & \vdots & & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_i = a_{i0}x_0 + a_{i1}x_1 + \cdots a_{i,n-1}x_{n-1} \\ \vdots \\ y_{m-1} \end{bmatrix}$$

Matrix-vector multiplication

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 3: JAVA Programming - Synchronization

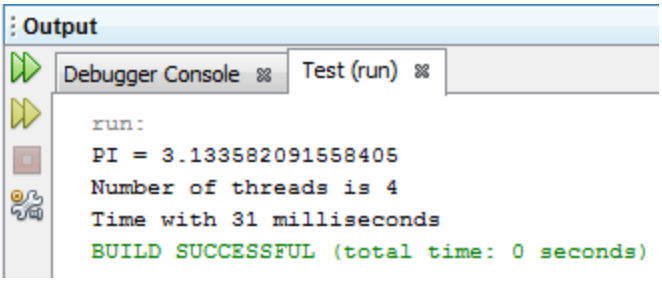
Objective

- To learn about threads cooperation and synchronization using lock mechanism in Java.

Lab Activities (see the hints below)

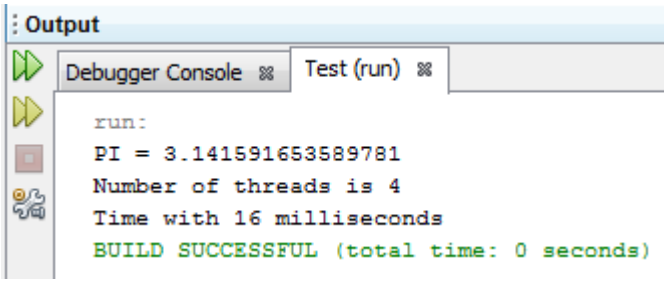
1. Write, compile and run a program which does the pi computation.
   a) without synchronization

Here a sample output:



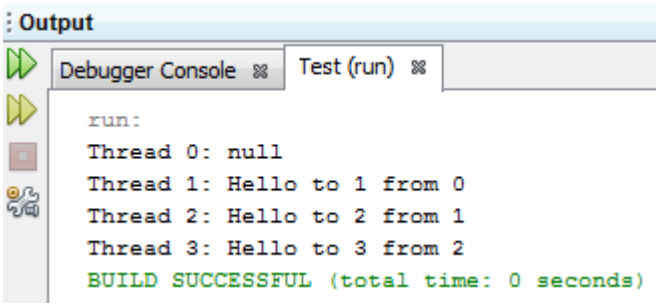
   b) with synchronized keyword
   c) with locks

Here a sample output:



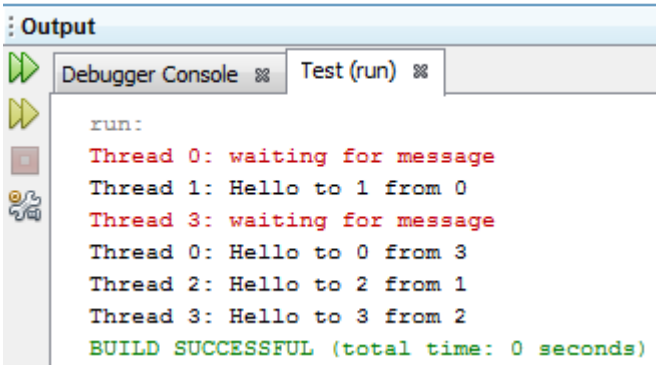
2. Write, compile and run a program for Read-Write (Consumer-Producer) problem (see the hints for more details).
   a) without synchronization

Here a sample output:



   b) using condition

Here a sample output:



Exercises

1. Write a program that compute the factorial of a given number (f(i) = 1*2*…..*(i-1) * i).
2. Write a program that find the max number in the given array.

Hints:

1. Pi computation:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^n \frac{1}{2n+1} + \cdots\right).$$

2. Read-Write (consumer-producer)

Each thread "send" a message to another thread. For example, suppose we have thread count is **t** threads and we want thread 0 to send a message to thread 1, thread 1 to send a message to thread 2, . . . , thread **t-2** to send a message to thread **t-1** and thread **t-1** to send a message to thread 0. After a thread "receives" a message, it can print the message and terminate. In order to implement the message transfer, we can allocate a shared array of strings. Then each thread can allocate storage for the message it is sending.

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 4 : A Revision on C Programming

Objective

- To write sequential programs using C language.

Lab Activities

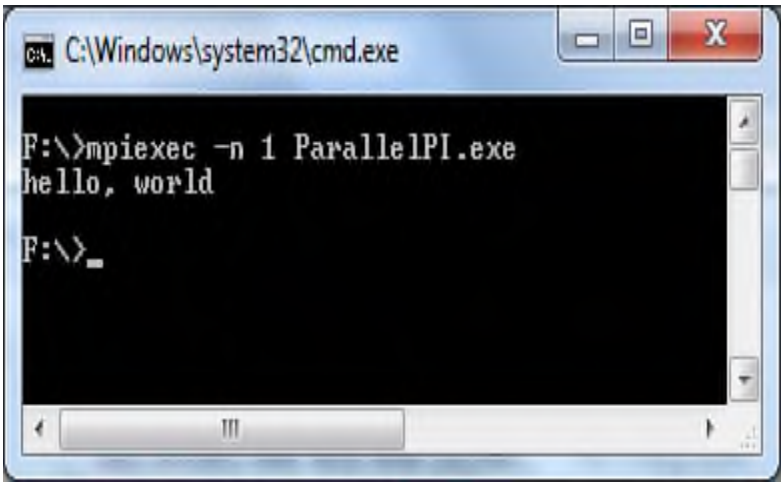
1. Write, compile and run the "`hello, world`" program. Please refer to the example program in section 3.1 of Pacheco.
2. Write, compile and run a program which does a vector addition, i.e. the summation of a list of numbers (Program 3.7 in section 3.4.6).
3. Write, compile and run a program which does a matrix multiplication (section 4.3).
4. Write, compile and run a program which does the pi computation (section 4.4).

Exercises

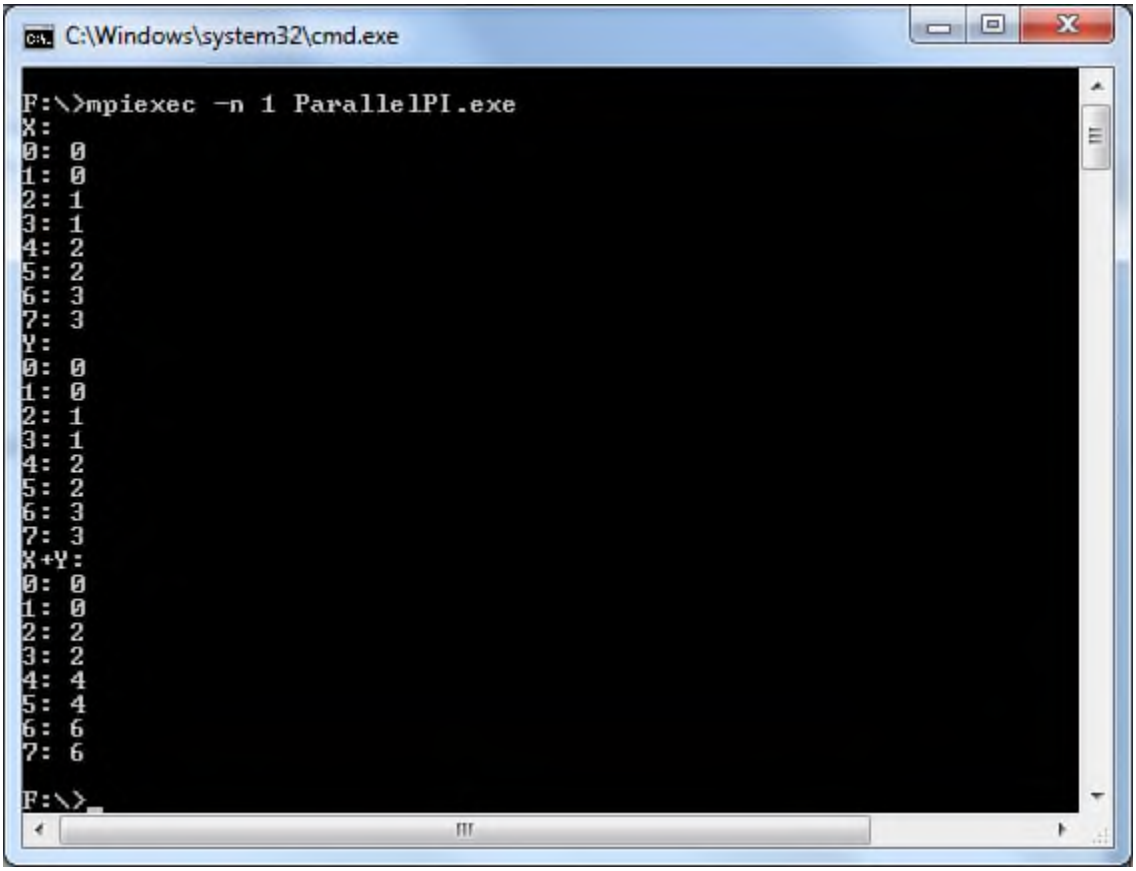
1. Write a program that does the following:
   (a) Compute the average of array elements.
   (b) Sum two numbers, stored in a and b.

2. Write trapezoidal rule program discussed in section 3.2.1.

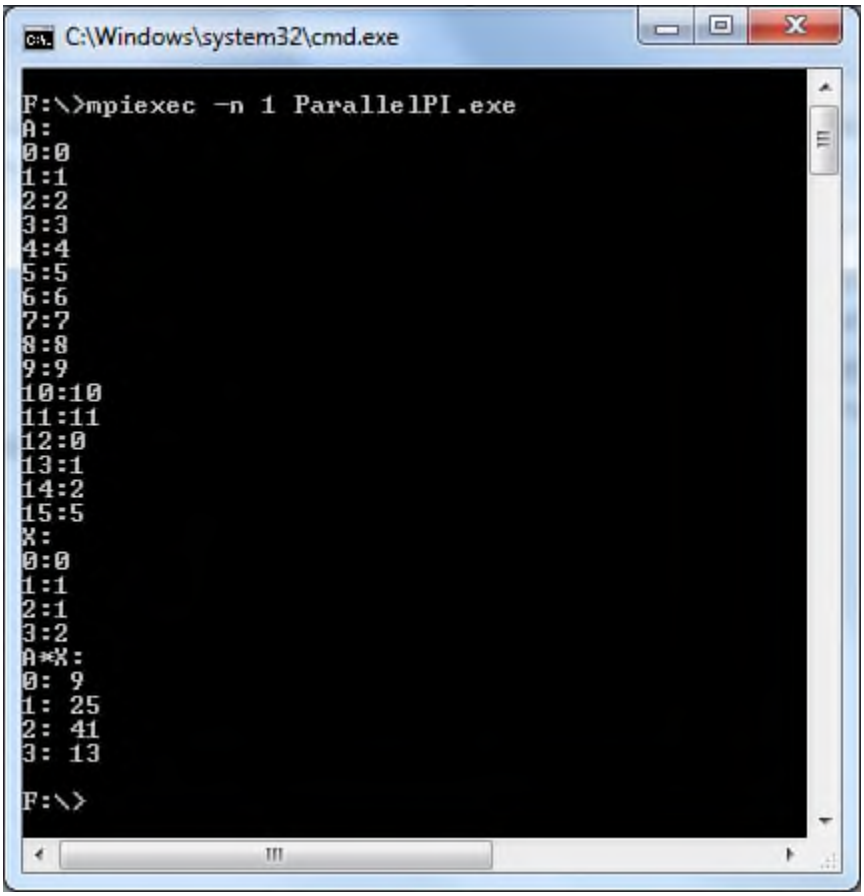
A sample for activity 1:



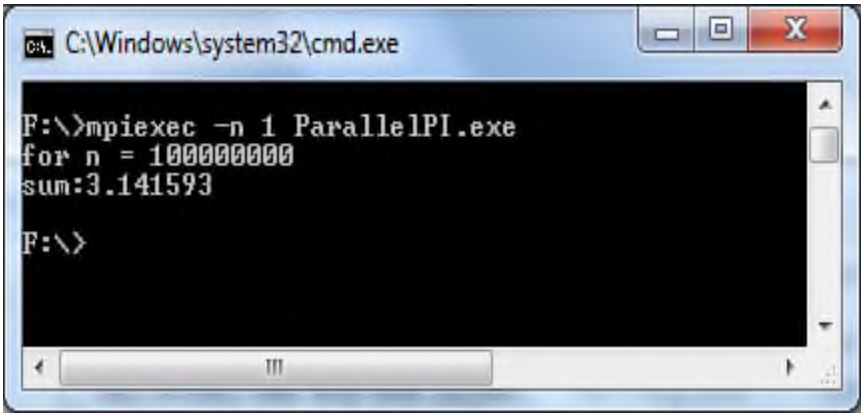
A sample for activity 2:



A sample for activity 3:



A sample for activity 4:



Note:

All provided samples were written using VS2017 which can't run the MPI programs directly by IDE, so "cmd.exe" was used to execute and run the programs, with VS2010 there is no need to use the command, the programs can be run directly by IDE.

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 5: Introduction to Message Passing Interface (MPI)

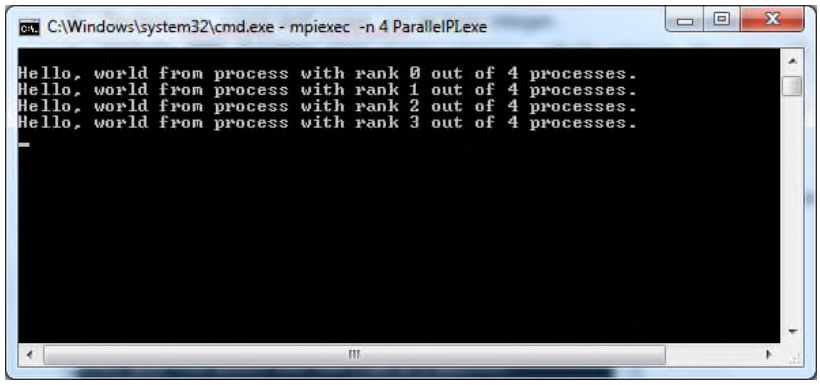
Objective

- To learn the basics of MPI program.
- To learn the MPI point-to-point communication.

Lab Activities

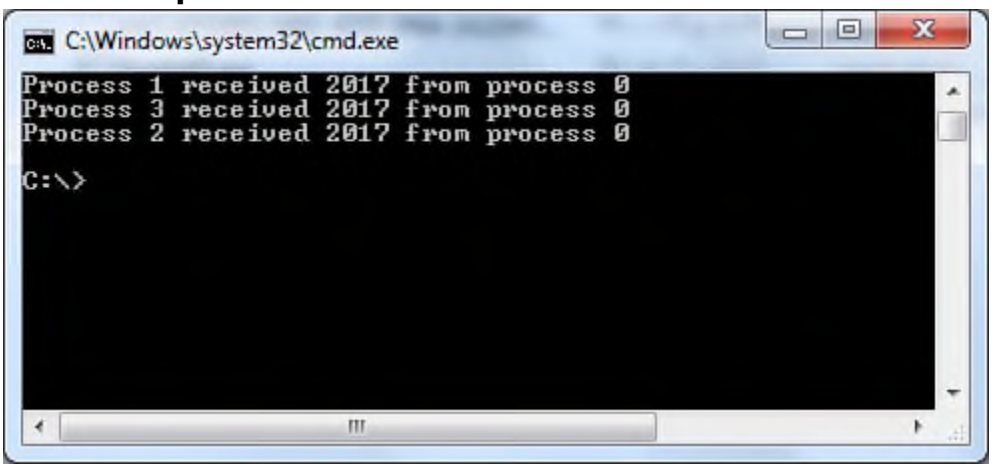
1. Using the "`hello, world`" program in the previous lab, please do the following:
   a. Include the MPI header file `mpi.h` at the top of your program.
   b. Include the MPI function, `MPI_Init()`, which initializes the program with the MPI environment and `MPI_Finalise()`, which does the cleaning up of the environment before the program ends.
   c. Edit the "`hello, world`" to be "`Hello, world from process with rank %d out of %d processes.`", `my_rank, comm_sz);`
   d. Declare `my_rank` and `comm_sz` as type integer.
   e. Include MPI function `MPI_Comm_size()` which returns the size of the communicator (i.e. total number of processes).
   f. Include MPI function `MPI_Comm_rank()` that tells the rank or id of the process.

2. Compile and run the "`MPI-hello, world`" program with 1 process, 2, 4 and 8 processes. Write down the output, note the differences and explain.
   Here a sample with 4 processes:



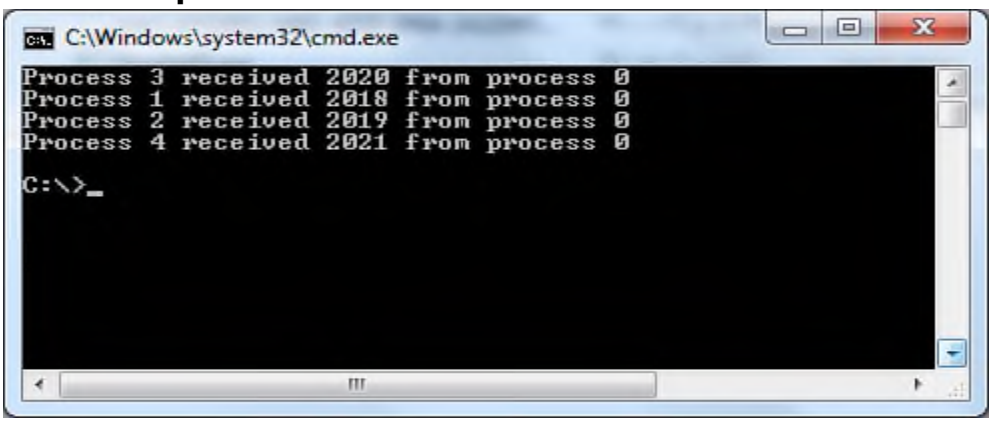
3. Write, compile and run an MPI program which does the following point-to-point communication.
    - Process 0 sends the value 2017 to process 1, 2 and 3.
    - The receiving process prints a message upon getting the value.

    Here a sample:



4. Modify the above program by having process 0 sends the value of `year` iteratively, starting from 2018 to 2021, to 4 different processes (i.e. 2018 to process 1, 2019 to process 2 etc).

    Here a sample:



Exercises

1. Please answer the following questions:
    (i) Name the header file that you need to run MPI programs. Briefly explain the content of the header file.
    (ii) What are the two MPI functions that must be included in every MPI program. What are they for?

(iii) The rank of a process can be identified using which MPI function?
(iv) Name the default communicator in MPI.
(v) Are `MPI_Send()` and `MPI_Recv()` a blocking or non-blocking operations?
(vi) Name the different variations of send and receive operations.

2. Compile and run Program 3.1 on page 85 with 1, 2 and 4 processes. Explain what the program does.

3. Modify the program so that it does the reverse, that is process 0 sends the greeting to the rest of the processes and each prints the greeting.

Note:

All provided samples were written using VS2017 which can't run the MPI programs directly by IDE, so "cmd.exe" was used to execute and run the programs, with VS2010 there is no need to use the command, the programs can be run directly by IDE.

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

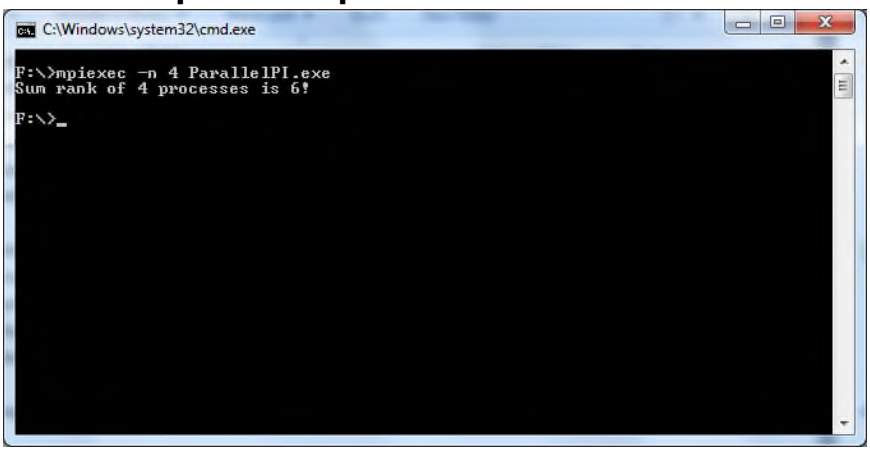
LAB 6: MPI – Collective Communication

Objective

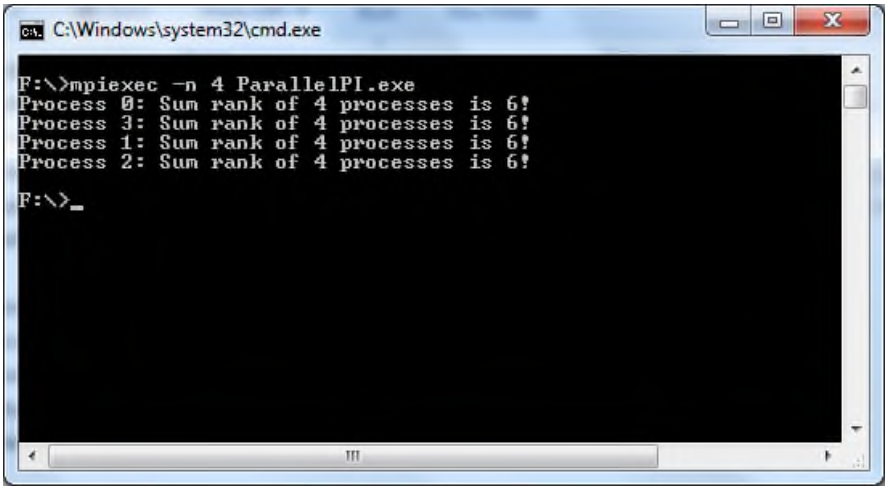
- To learn the MPI collective communication.

Lab Activities

1. Use `MPI_Reduce()` to sum up the process rank of all the processes. The final result is in process 0.
   Here a sample with 4 processes:

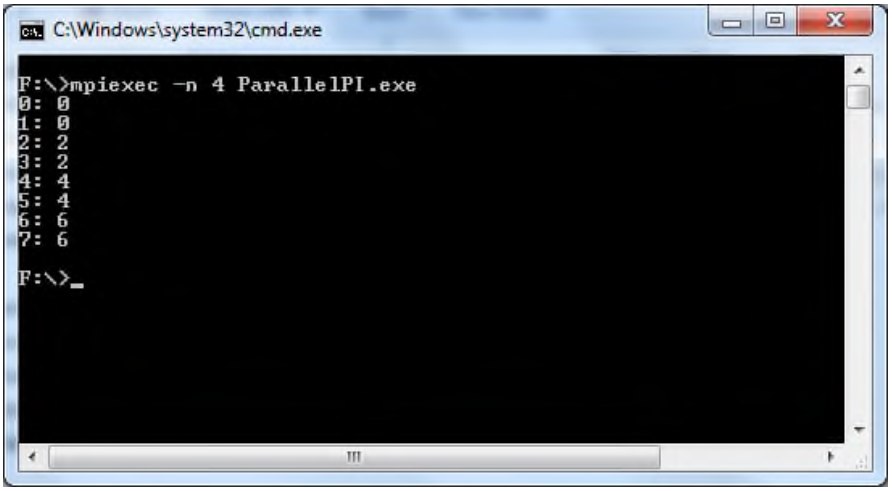


2. In the same program, use `MPI_Bcast()` to send the result of the summation from process 0 to all other processes.
   Here a sample with 4 processes:



3. Use only `MPI_Allreduce()` to do operation in (1) and (2) above.
4. Modify the sequential vector summation of Lab 1 using `MPI_Scatter()` and `MPI_Gather()`.

   Here a sample with 4 processes:



5. Repeat number (4) using `MPI_Allgather()`.

Exercises

1. Modify the program in (5) so that process 0 reads the values into the vector, does the summation and print the final results.
2. Examine the effect of multiple calls to `MPI_Reduce ()` as presented in Table 3.3 section 3.4.3 of Pacheco.

Note:

All provided samples were written using VS2017 which can't run the MPI programs directly by IDE, so "cmd.exe" was used to execute and run the programs, with VS2010 there is no need to use the command, the programs can be run directly by IDE.

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 7: MPI - Performance Evaluation

Objective

- To write simple parallel applications using MPI.
- To evaluate the performance of parallel applications.

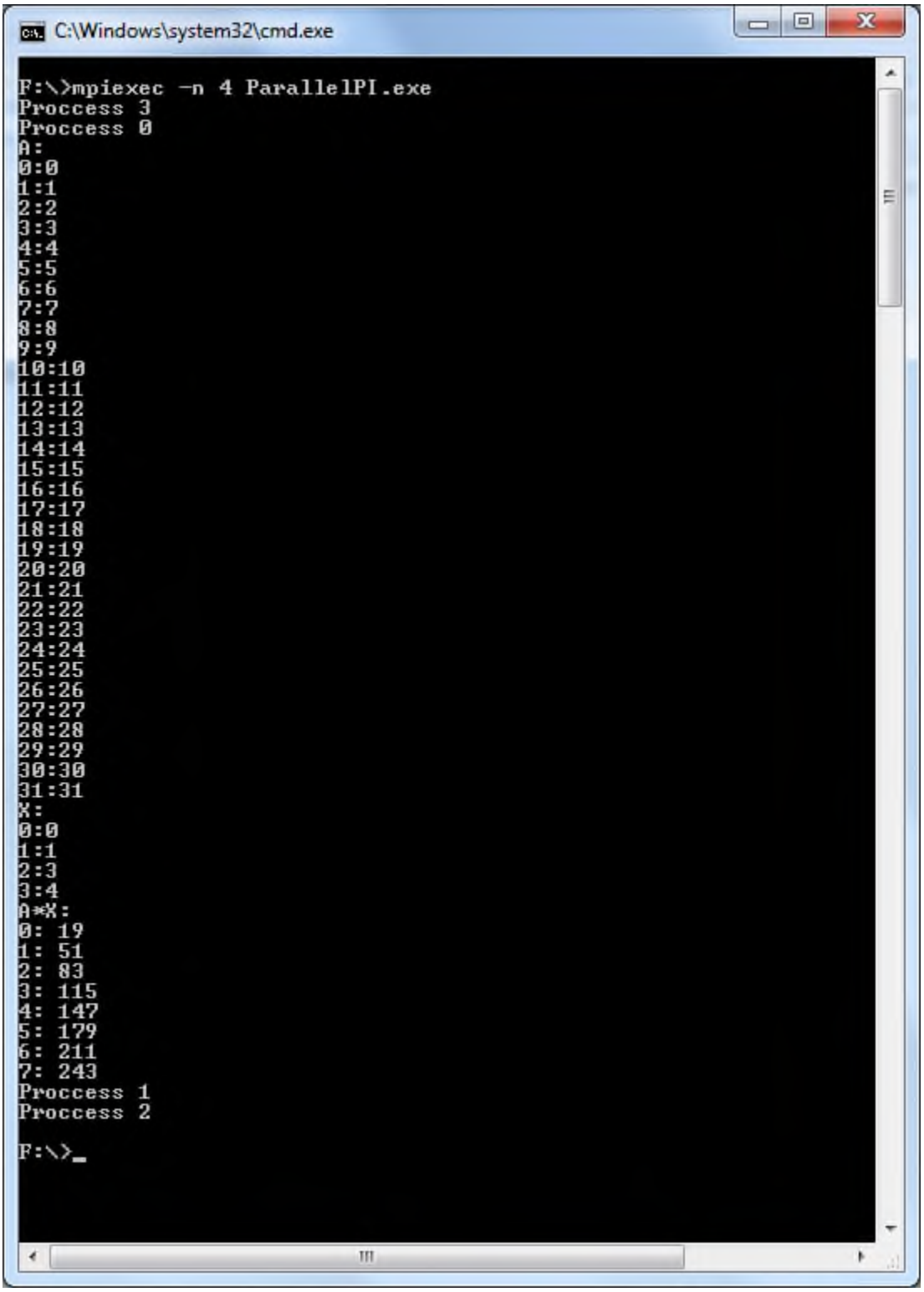
Lab Activities

1. Modify your sequential matrix multiplication in Lab 1, into parallel matrix multiplication.
2. Insert the necessary code to time the parallel matrix multiplication (please refer to section 3.6.1).
3. Get the sequential run-time and parallel run-time for 2, 4, 8 and 16 processes for varying matrix sizes as in Table 3.5 (Pacheco).
4. Calculate the speed up and the efficiency of the parallel program in (3) and plot a graph for both.

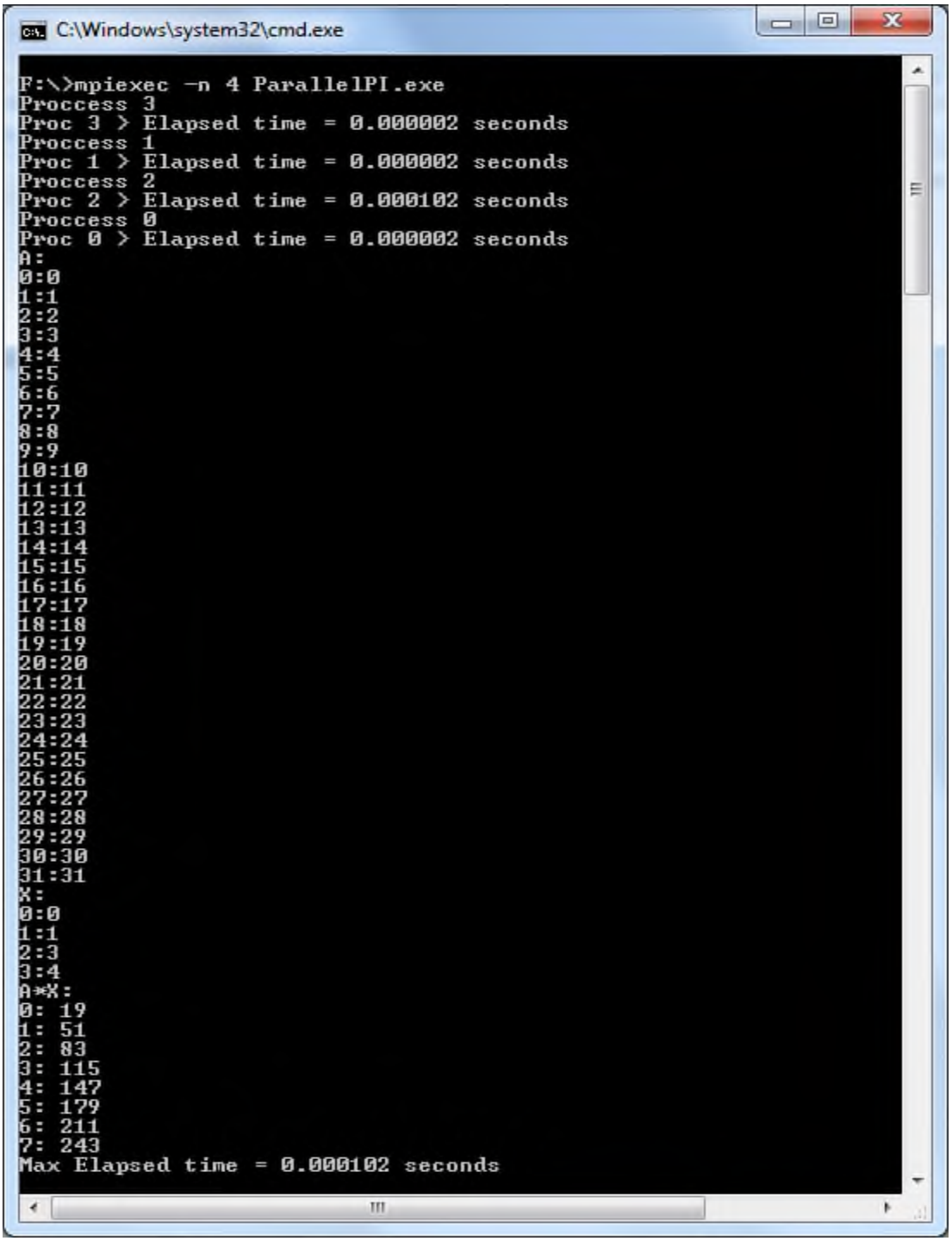
Exercises

1. Obtain a speed up and efficiency for the following application using varying processes and data size:
    a. parallel trapezoid program (section 3.2.2).
    b. parallel pi computation (section 4.4).

A sample with 4 processes for Lab Activity 1:



A sample with 4 processes for Lab Activity 2:



Note:

All provided samples were written using VS2017 which can't run the MPI programs directly by IDE, so "cmd.exe" was used to execute and run the programs, with VS2010 there is no need to use the command, the programs can be run directly by IDE.

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

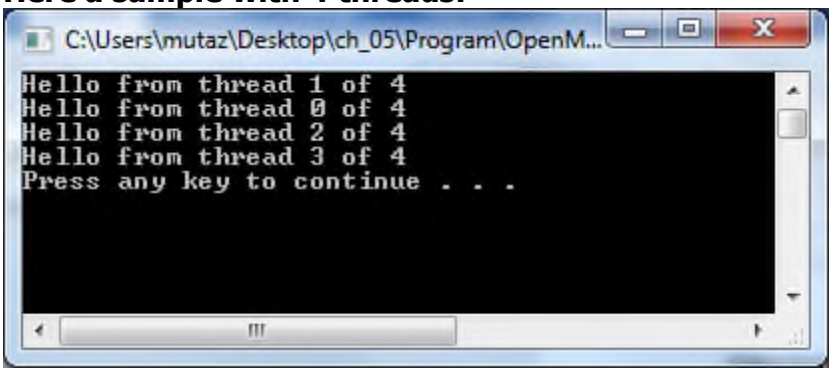
LAB 8: Introduction to OpenMP

Objective

- To learn how to parallelise program using OpenMP directives such as `parallel` and `parallel for`.
- To learn how to ensure data consistency in OpenMP programs using `critical`.
- To learn collective operation directive such as `reduction`.

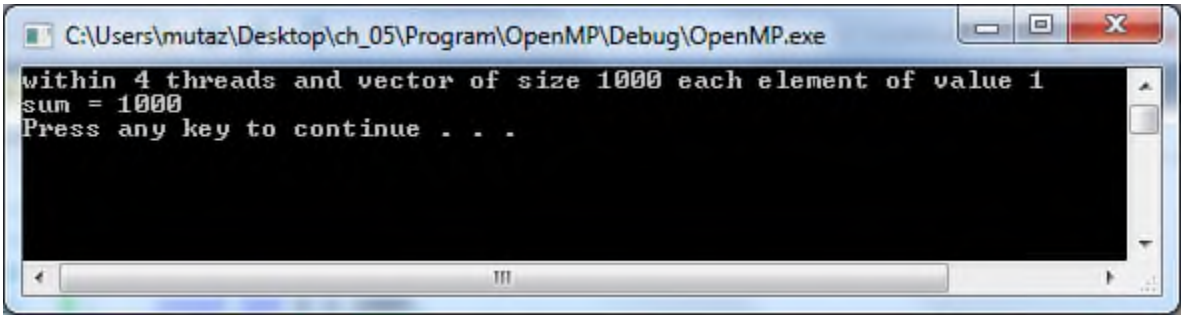
Lab Activities

1. Using the sequential "`hello, world`" program in Lab 5, make the necessary changes so that the program runs in parallel using OpenMP `parallel` directive.
   Here a sample with 4 threads:

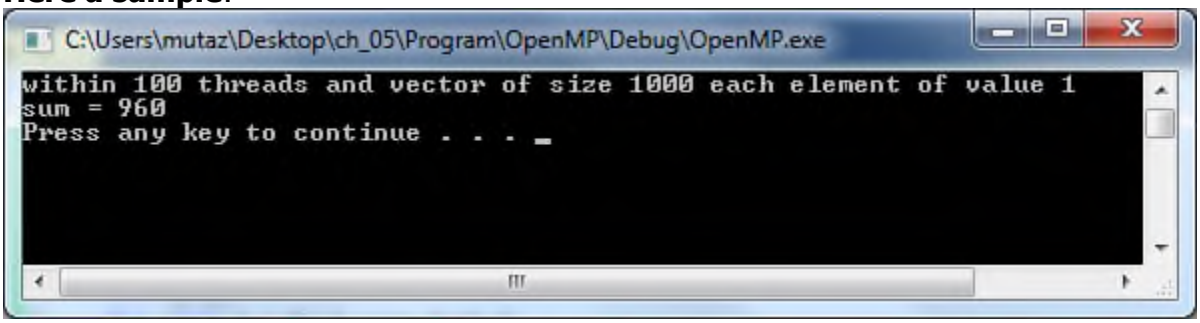


2. Compile and run the "`OpenMP-hello, world`" program with varying number of threads (e.g. 1, 2, 4 and 8 thread). Observe the output. Is the ordering of the output as expected, i.e. the first thread prints the output, followed by the second etc.

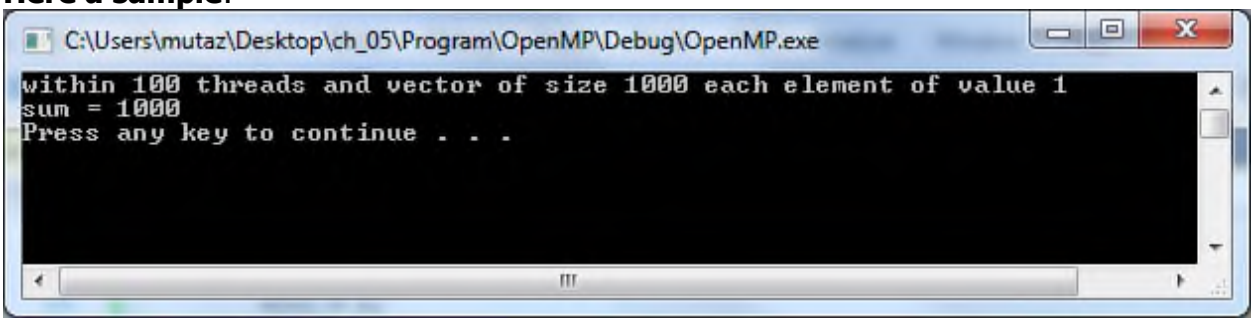
3. Modify your sequential vector summation program into a parallel program.
   Here a sample:



4. Run your program using 1000 numbers (with 10, and 100 threads). Observe the output. Do you always get the expected results.
   Here a sample:



5. Use `omp critical` to solve the data inconsistency problem in (4).
   Here a sample:



6. Modify the program in (5) by using `reduction`.

7. Modify the program (6) using `parallel for`.

Exercises

1. Compare and contrast between OpenMP, Java and Pthread "`hello, world`" programs. For Pthread refer to page 154 of Pacheco.

2. Include error checking code in your "`hello, world`" program (in Lab Activity A), which avoid errors if your compiler does not support OpenMP. Compile and Run the program.

3. Write a program with the master thread prints the following environment information:
   - The number of processors available
   - The number of threads being used
   - The maximum number of threads available
   - If you are in a parallel region

4. Write a program which will only do the summation of numbers between array a and b.

KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science & Engineering

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة–كلية علوم وهندسة الحاسب الآلي

CS424 Introduction to Parallel Computing Academic Year 2019-2020

LAB 9: Advanced OpenMP Programming

Objectives

- To learn how to use `schedule` clause and the different types of scheduling.
- To analyse the performance of OpenMP programs.

Lab Activities

1. Use your parallel summation to examine the effects of cyclic schedule with different chunk sizes. What can you say about default and cyclic schedule, and the effect of chunk sizes (see section 5.7).

2. Modify your sequential Pi program to an OpenMP-Pi (see section 5.5.4).

3. Obtain the speed up and efficiency of your pi program using 2, 4 and 8 threads. Plot the graphs for speed up and efficiency.

4. What can you say on the performance of OpenMP-Pi compare to MPI-Pi.

Exercises

1. Write, compile and run the following programs using OpenMP and analyse their performance (using 2, 4, 8 threads).
   (a) The trapezoid rule program (section 5.4)
   (b) Parallel matrix multiplication (section 5.9).