Assignment Project

Marks: This assignment is worth 100 marks and 25% of all marks for the unit.

Due Date: The assignment is due in **Week 11, Mom 18/May/2015, 2PM**.

Submission: (i) The assignment submission must be made through Moodle portal for this unit **by the due date** unless advised otherwise by the lecturer.
(ii) The submission of this assignment must be in the form of a single GZIP, TAR, or ZIP file. Proprietary Winzip format will not be accepted.

Extensions: *No extensions will be given.*

Lateness: Late penalty of up to 5% marks deduction/day including weekends.

Authorship: This assignment is an individual assignment and the final submission must be identifiably your own work. Breaches of this requirement will result in an assignment not being accepted for assessment and may result in disciplinary action.

Cover Sheet: A completed individual assignment covered sheet is required with the submission.

General instructions

- All MPI programs must compile with Open MPI using the GNU C compiler (gcc).
- All OpenCL programs must compile with AMDAPPSDK-2.9-x using GNU C compiler (gcc)
- Programs using more than one file for compilation and execution must include a Makefile.
- All programs must be accompanied with a set of clear and concise step-by-step instructions for compiling and executing the programs.
- The number of processors required to test the MPI program(s) must be specified (e.g. –np 2). *Lack of clarity in the instructions may adversely affect the assessment.*

Assignment Submission Instructions

The deliverable folders (sub-directories) must be compressed into a single archive using either standard ZIP, TAR, or GZIP utilities. The archive must preserve the folder information. Submit the archive file with the appropriate extension (.zip or .gz) at the Moodle website for the unit.

Part A (25 marks)

Write short MPI programs that demonstrate the use of the following functions.

MPI_Send, MPI_ISend, MPI_Bcast, MPI_Comm_rank, MPI_Abort (5x5= 25 marks)

Part A Deliverables will comprise a sub-directory with,

1. Five appropriately named source code files (e.g. MPISend_demo.c) OR ***preferably*** a single file with all the five functions.

2. A README file in PDF format that contains compilation and execution instructions.

Marking Criteria		Marks
1.	The program compiles successfully without any warning(s) and correctly implements the function.	2
2.	Code is well structured, commented, and easy to understand.	2
3.	The compilation and execution instructions are accurate and clearly stated.	1
Total marks for this part:		5x5=25
Note: Part A (items 1 to 3) will be marked during the in-lab demo. You must however include the code and README files for this part within the Moodle submission.		

Part B (65 Marks)

Select ONLY ONE, of the following project options for this part of the assignment:

Project option 1: Bigdata processing with OpenCL or OpenMP

Evolutionary techniques provide an efficient path to finding best-fit solutions to complex scheduling problem. In such problems, addition of even a few extra variables can lead to exponential increases in processing time. Within large data sets the problem often become intractable for conventional computers as there is a corresponding increase in number of variables. Furthermore, faster turn around time is at time necessary for online services such as a recommender system.

This project provides the *C* code of a standard evolutionary technique known as *genetic algorithms* (GA), which is purely sequential i.e. written for a single CPU/core (Moodle Assignment Project file: Simple_GA.zip).

The aim of this project will be to convert this sequential code into an OpenCL or OpenMP[1] based parallel GA code, which can efficiently utilise a multicore CPU/GPU architecture.

The *C* code is self-explanatory and requires no in-depth knowledge of GA for parallelisation with OpenCL or OpenMP. However background information on the GA and its implementation is being provided to further assist with the understanding of the program design and its working principles (Moodle Assignment Project files: GA Coding Primer.pdf).

[1] The alternative to program with OpenMP is mainly being provided for students who may have some difficulty in accessing an OpenCL supported computer.

[2] The vessel movement is instantaneous and can be to a non-contagious location. In other words

Marking Criteria B project option 1		Marks
1.	The program compiles successfully without any warning(s) and correctly translates the GA code into an equivalent OpenCL or OpenMP code. It is accompanied with complete instructions for compiling and running the code in a README file (or a *Makefile*).	10
2.	Code is well structured, commented, and easy to understand.	5
3	The written report fully describes (1) the program structure, (2) the OpenCL or OpenMP based parallel GA scheme, and (3) Performance metrics that include a comparison of the execution time for the parallel GA code vis-à-vis the original GA code (provided with this assignment). The choice of other metrics will be at the discretion of each student.	20+20+10=50
Total marks for this part:		65
Notes: Item 1 will be marked during the in-lab demo. Items 2 and 3 are to be included within the Moodle submission.		

Important Note: Students selecting this project must ensure that their personal computer can support OpenCL by compiling and executing clDeviceQuery.cpp available within opencl-platform-test-ex.zip http://moodle.vle.monash.edu/mod/resource/view.php?id=2275074 if planning to work from home. Also to note, that this unit does not guarantee access to lab PCs outside the timetabled lab hours. It is entirely the students' reposiblity to ensure they have satisfactory access to an OpenCL supported platform.

Project option 2: Distributed Event Modelling with MPI

Your naval fleet patrols an area comprising **1100 distinct** locations. Each vessel can occupy any one of these locations randomly[2] at the end of sampling interval[3]. For a vessel to be able to launch a strike, other vessels must accompany it. The rules for launching a strike are summarised as follows:

1. At least **three** vessels must share the same location, at a given point in time, for a strike to be counted.

2. The fleet may generate more than one strike at an instant of time. It will however depend on the number of locations with **three** or more vessels present at that instant of time.

[2] The vessel movement is instantaneous and can be to a non-contagious location. In other words there is no constraint as to how a vessel moves from one location to another (except for the location being randomly assigned).

[3] Sampling interval is to be selected by the programmer and should ideally be a small fraction of 60 seconds i.e. the overall runtime period stipulated for calculating the total number of strikes.

3. There is no limit on the number of vessels in the fleet. The objective is to achieve the highest possible strike rate. It may however be pointed out that increases in fleet size will increase the probability of satisfying Rule no. 1 (above), but doing so will also slow the program owing to higher inter-process communication overheads.

Assume that a set of MPI processes represents the fleet and each MPI process represents a naval vessel.

Marking Criteria B project option 2		Marks
1.	The program compiles successfully without any warning(s) and correctly implements the launch rules. It is accompanied with complete instructions for compiling and running the code in a README file (or a *Makefile*).	10
2.	Code is well structured, commented, and provisions for efficiency.	5
3	The written report fully describes (1) the program structure and the provisions made their in for efficiency. (2) the inter-process communication scheme, and (3) Performance metrics that include the average number of launches generated by the program over a minute. The program must be repeated at least three times to estimate the average launch number and other metrics. The choice of other metrics will be at the discretion of each student.	20+20+10=50
Total marks for this part:		65
Note: Item 1 will be marked during the in-lab demo. Items 2 and 3 are to be included within the Moodle submission.		

Part B Deliverables will comprise a sub-directory with,

1. Appropriately named source code file(s) (e.g. GA.c or Fleet_Sim.c).
2. A README file in PDF format that contains compilation and execution instructions.
3. A written report file in PDF format (1000 word limit applies).

Part C: Content-addressable search engine (10 Marks)

Problem Statement: Searching for images and other multi-media on the Internet require a distributed associative memory scheme. Hierarchical Graph Neuron is a fast distributed associative memory technique that is well suited for distributed systems. Details of the Hierarchical Graph Neuron Technique can be obtained by legally downloading the following paper from Monash Library's website:

Title: A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition

Persistent Link for Downloading the Paper PDF:
http://ieeexplore.ieee.org.ezproxy.lib.monash.edu.au/servlet/opac?punumber=72

This paper appears in: Neural Networks, IEEE Transactions on
Issue Date : Feb. 2008, **Volume :** 19 , Issue:2 , **On page(s):** 212 - 229 **ISSN :** 1045-9227, **INSPEC Accession Number:** 9794258
Digital Object Identifier : 10.1109/TNN.2007.905857 , **Date of Current Version :** 07 February 2008

Task: Propose how the HGN associative memory technique may be parallelised over a group of processors. (10 marks, 500 word limit applies).

Hint: Proposal to parallelise an equation-solving algorithm may be seen, in the opening paragraphs of Section 5, of the following paper.

http://users.monash.edu.au/~asadk/JournalPapers/parcgfem.pdf

Marking Guide:
Identification of parallelism within the HGN scheme (5 marks). A flow-chart for the parallel HGN (5 marks).

Part C Deliverable will be the report to be included in your Moodle submission.