University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**Distributed and Parallel Technologies**			
Coursework Number	**1**			
Deadline	**Time:**	**4:30**	**Date:**	**5/11/2018**
% Contribution to final course mark	**5%**			
Solo or Group ✓	**Solo**	✓	**Group**	
Anticipated Hours	**12**			
Submission Instructions	**On the DPT Moodle page, use the "Coursework Stage 1 Submission Assignment" to upload your Report. It is crucial that your report** **1. follows the specified structure** **2. provides, in appendices, your** **+ Go program and** **+ OpenMP program**			
Please Note: This Coursework cannot be Re-Assessed				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via https://studentltc.dcs.gla.ac.uk/ for all coursework

---

Comparing Parallel and Distributed Programming Models:

Coursework Stage 1: Performance and Programmability

Of Shared-memory Parallel Models

BSc and MSc Students

Purpose

The coursework is in two stages, in this first stage you will develop parallel programming, experimentation, and technology evaluation skills. The parallel architecture is a shared-memory multicore. Given sequential versions of a program, you will develop and measure parallel versions in C with OPENMP and Go. Finally you will compare the performance and programming models of the technologies.

Organisation

The assessed coursework work is to be carried out individually. You must develop and tune the parallel performance of the C+OpenMP and Go programs, make systematic measurements, and prepare a comparative report. It is relatively easy to produce a simple parallelisation of both programs, however *additional marks are available for thoughtful parallel performance tuning*.

The program to be parallelised sums the Euler totient values over sequences of integers. The sequential versions of the program are available from

http://www.dcs.gla.ac.uk/~trinder/Courses/DistrParTech/coursework/totientRange.c

http://www.dcs.gla.ac.uk/~trinder/Courses/DistrParTech/coursework/totientRange.go

Deliverable

The deliverable is report (including sources) with the structure below. Reports not following this structure, or not containing all of the specified results, will be penalised.

The measurements are based on three data sets:

DS1: calculating the sum of totients between 1 and 15000.
DS2: calculating the sum of totients between 1 and 30000.
DS3: calculating the sum of totients between 1 and 60000.

1. You should complete the Go and OpenMP Lab exercises before starting the coursework.
2. You should develop and test on a lab/home machine, and only run performance measurements on a GPG cluster node. You have access to gpgnode-01, gpgnode-02, and gpgnode-03.
3. The measurements you report should be the median (middle) of three executions
4. You may find it useful to write scripts to run the measurements
5. To ensure a fair comparison all measurements should be made on a very lightly loaded GPG cluster node. Check the load on nodes using something like the unix `top` command.
6. Graphs and tables must have appropriate captions, and the axes must have appropriate labels.

Section 1 Comparative Sequential Performance (2 marks)

Complete the table below showing the runtimes of the sequential C and Go programs on DS1 and DS2 on a GPG cluster node. Reflect on the differences.

Data Set	Go Runtimes (s)	C Runtimes (s)
DS1		
DS2		

Table 1: Comparing C and Go Sequential Runtimes

Section 2 Comparative Parallel Performance Measurements (12 marks)

You should measure and record the following results in numbered sections of your report. Runtime measurements should be the middle (median) value of three executions on a GPG cluster node. **N.B.** For comparison purposes the performance of the Go and C+OpenMP systems must be reported on the *same* graph. You may also plot other graphs to show interesting features, or use larger numbers of cores.

Section 2.1 Runtimes.

DS1: execution times for the sequential C and Go programs, and the parallel C+OpenMP and Go programs on 1, 2, 4, 8, 12, 16, 24, 32, 48, 64 threads/goroutines on a GPG Cluster node.

DS2: execution times for the sequential C and Go programs, and the parallel C+OpenMP and Go programs on 1, 2, 4, 8, 12, 16, 24, 32, 48, 64 threads/goroutines on a GPG Cluster node.

DS3: execution times for the parallel C+OpenMP and Go programs on 8, 16, 32, 64, threads/ goroutines on a GPG Cluster node

Section 2.2 Speedups.

Plot three absolute speedup graphs corresponding to the runtime results for DS1, DS2 and DS3 showing the ideal speedup and the speedups for the C+OpenMP and Go programs. Recall that absolute speedup is calculated using the runtime of the sequential program on a single core.

Section 2.3

Complete the table below summarising the sequential performance and the best parallel runtimes of your C+OpenMP and Go programs.

Language	Sequential Runtime (s)	Best Parallel Runtime (s)	Best Speedup	No. Cores
Go				
C+OpenMP				

Table 2: Comparing C+OpenMP and Go Parallel Runtimes and Speedups

Section 2.4

A discussion of the comparative performance of the C+OpenMP and Go programs. **Max 1 A4 page.**

Section 3 Programming Model Comparison (6 marks).

An evaluation of the Go and C+OpenMP parallel programming models for the totient application. You should indicate any challenges you encountered in constructing your programs and the situations where each technology may usefully be applied. The comparison should be based on the TotientRange application, focus on the technology in general, and be supported by technical arguments. **Max 1 A4 page.**

Section 4: Reflection on Programming Models MSc Students Only (5 marks)

An evaluation of the C+OpenMP and Go parallel programming models in general. This section should discuss the advantages and disadvantages of the programming models. It should discuss issues related to the parallel performance, programmability and usability of the models. It should address issues of portability, in principle, discussing which kind of architectures are best targeted with each programming model. This discussion needs to be general, but can draw on the experience you gained in using the models on the Totient application. **Max 1 A4 page.**

Appendix A C+OpenMP TotientRange Program (5 marks)

A listing of your C+OpenMP TotientRange program, clearly labelled with the author's name. Also include a paragraph, and possibly diagram(s), identifying the parallel paradigm used, and performance tuning approaches used.

Appendix B Go TotientRange Program (5 marks)

A listing of your Go TotientRange program, clearly labelled with the author's name. Also include a paragraph, and possibly diagram(s), identifying the parallel paradigm used and performance tuning approaches used.