# COMP60011 - LAB 2

Understanding java.util.concurrent & implications of Ahmdal's law

The marking of Parts I, II & III will require a written report. This report, a document (only PDF, ODF, or Word 2003), will be emailed with subject "COMP60011 Lab 2 Group X" to comp60011@googlemail.com before 9:00am on Monday 30th Nov.

Part I – Implications of Ahmdal's Law for the design of multi-core systems.
At http://www.cs.wisc.edu/multifacet/amdahl/ you will find a tool that draws speedup curves based on the paper http://dx.doi.org/10.1109/MC.2008.209

QI.1 Explain what the terms "symmetric", "asymmetric" and "dynamic" mean.
QI.2 Explain the basic assumptions made for the calculations.
QI.3 Plot the graphs using 128 BCEs, 256 BCEs and 512 BCEs with fraction of parallelism values 0.99, 0.9, 0.8 and 0.7. Explain the results obtained.
QI.4 A related "law" is Gustafson's Law. Explain Gustafson's law and provide the original reference where it was described.
QI.5 Consider the mergesort algorithm investigated in Lab 1. What would be the speedup curve based on the drawing tool? Justify how you selected the values to draw the curve.
QI.6 If you were to improve the mathematical formulae and considering mergesort, describe what you would like to improve.

Part II – Reentrant Read-Write Locks
(java.util.concurrent.locks.ReentrantReadWriteLock)

QII.1 Describe what is a Reentrant Read-Write Lock.
QII.2 Describe the basic data structure used to implement ReentrantReadWriteLock (see http://hg.openjdk.java.net/jdk7/tl/jdk/file/cc5db1a62f70/src/share/classes/java/util/concurrent/locks/ReentrantReadWriteLock.java).
QIII.3 What is the extra functionality provided by this class when compared with the standard Java lock (i.e. the lock associated with every Java object)? Why is this extra functionality needed?

Part III – Concurrent Data Structures in java.util.concurrent
(source code available at http://hg.openjdk.java.net/jdk7/tl/jdk/file/cc5db1a62f70/src/share/classes/java/util/concurrent/)

QIII.1 Describe how the ConcurrentHashMap class is implemented.
QIII.2 Describe how the CopyOnWriteArrayList class is implemented.
QIII.3 Select one of the data structures available in the package (different from ConcurrentHashMap and CopyOnWriteArrayList) and write a Java test program that uses such data structure. Describe the program, describe the performance metric that will be used and report performance for 1, 2, 4, 6, 7, 8, 12 and 16 threads.