COMP60011 - LAB 3

Understanding coarse-grain and fine-grain synchronization & using Simics

The marking of Parts I & II will require a written report. This report, a document (only PDF, ODF, or Word 2003), will be emailed with subject "COMP60011 Lab 3 Group X" to comp60011@googlemail.com before 9:00am on Monday 7th Dec.

The objective of this lab is to understand coarse-grain and fine-grain synchronization, and experiment with a hardware simulator (Simics and GEMS). The benchmark rand-array (described in the paper http://doi.acm.org/10.1145/1168857.1168900 (Hybrid Transactional Memory, ASPLOS 2006) may be useful for Part I where we will design a parallel program. Note that Simics will only run on the Nehalem systems.

Part I –**Design a Parallel Route Finder program.**

Description of the program and algorithm. The parallel route finder program reads as command line parameters the number of threads, a seed (integer value), number of routes and size of a two dimensional square array. It creates a two-dimensional square array of the appropriate size and is initialised to empty. It creates a set of the appropriate size of randomly generated correct start and destination pairs: start point (si,sj) & destination point (di,dj) (correct implies that the start point and destination point are different and are within the array bounds of the two-dimensional array; use the seed in the constructor of the random number object). In parallel, the specified number of threads find the correct routes (a correct route is a set of contiguous array elements that connect the source and destination by means only of horizontal and vertical lines, and any point part of a route can only belong to one route). Each thread will use a best-first search algorithm, where best is defined as following a straight line as much as possible. Finally, the program executes a sequential check of the routes found and reports the execution time without the initialisation or sequential check.

QI.1 Describe a coarse-grain parallelization of Parallel Route Finder. Only describe the parallel section of the program.

QI.2 Describe a fine-grain parallelization of Parallel Route Finder. Elaborate on the risks of deadlock and how they are avoided. Only describe the parallel section of the program.

QI.3 How would Transactional Memory change the design of the program?
Hint: Look at benchmark rand-array

Part II – Using a hardware simulator

Instructions for QII.1 Using a Perfect Memory Model

You will need to create a directory for example "simics_area" in your home directory and execute a simics script.

mkdir simics_area
cd simics_area
/usr/virtutech/simics-3.0.31/bin/workspace-setup

After this you would be able to start the simics simulator itself.

We will execute now a virtual x86 hardware:

./simics targets/x86-440bx/tango-common.simics

This will load simics and you will need to press "c" to allow the OS to be loaded into the virtual hardware. In the virtual hardware you can login using "root" and "simics". Experiment with running different commands. Once you are comfortable with the environment exit Simics (pressing "Ctrl-c" "q" on the simics console).

QII.1 Simics is "a full system simulator". Report what it means and contrasts with "an instruction set simulator". Also compare "an execution driven simulation" with "a trace driven simulation". What is the difference between a "full system simulator" and hypervisor or virtual machine monitor. What means using a "perfect memory model"?

Instructions for QII.2 Using a sophisticated cache model - GEMS

We have executed scripts for running one Transactional Memory microbenchmark (btree) on a simulated Hardware Transactional Memory system (LogTM). Simics provides the basic simulation engine and GEMS provides the memory model. You can obtain the files from http://www.cs.manchester.ac.uk/apt/COMP60011/lab3_gems.zip.

For example, btree-TM-priv-alloc-20pct-4p-1t-default-EagerCD_EagerVM_Base_NoPred-H3_2048_4_Parallel-Perfect_-Perfect_-10000-11517-11517.stats contains the statistics from running btree on 4 cores. There are three files, 1 core, 4 cores and 8 cores.

Look at the results and discuss with the demonstrators the contents of the results.

QII.2.A Which information is the most useful?

Hint for important information:

```
Basic:
        g_NUM_PROCESSORS
        Ruby_cycles or Virtual_time_in_seconds
        transactions_started
        transactions_ended
        xact_aborts

More detailed ones:
        instructions_per_transaction
```

```
cycles_per_transaction
misses_per_transaction
 XACT_BREAKDOWN_NON_TRANS_CYCLES
 XACT_BREAKDOWN_TRANS_CYCLES:
 XACT_BREAKDOWN_GOOD_TRANS_CYCLES:
 XACT_BREAKDOWN_ABORTING_CYCLES:
 XACT_BREAKDOWN_COMMITING_CYCLES:
 XACT_BREAKDOWN_BACKOFF_CYCLES:
 XACT_BREAKDOWN_BARRIER_CYCLES:
 XACT_BREAKDOWN_STALL_CYCLES:
```

QII.2 Comment on the results provided for GEMS.

The answer to this question should provide the performance graphs and should select not more than 2 performance metrics. Describe which performance metric you have selected and why it provides interesting information.