TDT4200 – Assignment 4

Stian Jensen

Problem 1

a) Explain the difference between MPI, OpenMP and Pthreads.

MPI is based around sending messages between processes.

OpenMP uses shared memory.

Pthreads is a library for creating and managing threads.

b) Explain the difference between shared and distributed memory systems.

In shared memory systems, all processes have access to all the same memory, and can communicate simply by writing to and reading from the same memory.

In distributed memory, processes can't access the same data, and need to explicitly send information to each other in order to be able to collaborate.

c) Mention and explain benefits and drawbacks of shared memory relative to distributed memory.

A benefit of using shared memory as opposed to distributed memory, is that processes don't need to do any message passing, and avoids the overhead that implies.

Distributed memory can be easier to program, especially it makes it easier to avoid race conditions in the code.

Problem 2

a) Mention and explain the different OpenMP schedules

There are three types of scheduling:

static, dynamic and guided.

Static scheduling means that all threads are allocated a number of iterations before they start executing.

In dynamic scheduling a smaller number of iterations are assigned to the threads to start with, and when a thread is finished, it gets allocated a new set of iterations from the ones that are left.

Guided scheduling is similar to dynamic in that not all iterations are allocated at the start. However, threads get a large contiguous chunk to start with, and the chunk size gradually decreases as the program runs, down to a set limit.

b) Consider the following functions, whose execution time depends on their argument as follows:

I: **128t**
II: The last thread will run with 896 <= i < 1024, and therefore be slowest.

The execution time will be t(896+897+···+1022+1023) = t * (1023*1024 - 895*896) / 2 = **122816t**

III: The longest running thread will be the one that's running from i = 900 to i = 999.
That will probably be executed by the same thread which runs from i = 100 to i = 109

Total execution time is therefore: t(199*200 - 99*100) / 2 + t(999*1000-899*900)/2 = **109900t**

IV:

c)

Total serial execution time will be 1000*t + 1000*999/2*t = 500500*t.

This means that the best workload distribution will be for each thread to approximate 500500/8 = 62562*t in execution time.

Problem 3

a) What is a race condition?

Race conditions may occur if some code is dependent on running in a specific sequence. When the code – due to different parts of a program run in parallel – does not execute in the “correct” order a bug occurs.

b) What is a deadlock?

A deadlock might occur when a process A has requested a lock for a resource, and is trying to request another lock for a second resource. If that second resource has already been locked by process B, process A will have to wait for it. Normally this is fine, but if process B needs a lock for a resource that process A already has locked before process B will release its own lock, they will both be waiting for each other. That is a deadlock.

c)

I: Race condition. Threads should lock a mutex for the j variable.
II: May deadlock
III: No deadlock or race condition
IV: No deadlock or race condition

Problem 4

a) For each of the alternatives, determine how many additions must be performed

A: 31
B: 31
C: 31
D: 31

b) For each of the alternatives, determine the maximum number of threads which can operate in parallel.

A: 2
B: 4
C: 8
D: 16

c) Determine which alternative is best to use if you have from 2 to 16 cores available. Show your work.

With 2 cores, option A is the best. With 4 cores, option B is better since option A only has work for 2 simultaneous cores.

Similarly, with 16 cores option D is best.