UiT

THE ARCTIC
UNIVERSITY
OF NORWAY

EXAMINATION PAPER

Exam in: INF-3201 Parallel Programming
Date: Wednesday December 3, 2014
Time: 09:00 - 13:00
Place: Åsgårdveien 9

Approved aids: None

The exam contains 3 pages, including this cover page.

Contact person: John Markus Bjørndalen.
Phone: 90148307

NB! It is not allowed to submit scratch paper along with the answer sheets

UNIVERSITY OF TROMSØ

NB: for questions that ask for pseudocode, you can, if you prefer to do so, use languages such as Python, Java, C and Go instead. There is no requirement for remembering exact parameter lists or exact names of API calls.

You are free to answer in Norwegian or English, or a combination of the two.

Good luck!

1) Message passing (15%)

a) Give a short explanation of what blocking vs. non-blocking communication is in message passing systems and show how the two can be used. You can use MPI as an example (or MPI-like pseudocode).

b) What is the difference between asynchronous and synchronous operations?

c) When does a locally blocking send routine in MPI behave as a synchronous routine?

2) Collective/group operations (15%)

a) Explain the term group (or collective) operations. A short description is enough.

Explain the following operations:

b) Broadcast

c) Reduce

d) Gather

e) Scatter

3) GPGPU computing with CUDA (20%)

a) What is a “Kernel”?

b) What are “thread blocks”, “warps” and “grids”? How are they related?

c) Outline how a program that uses a GPU looks and works. The description should be high level and can use pseudocode. Assume that the program has some data that the GPU should use and that the GPU code should produce some resulting data for the CPU. Don’t worry about performance optimisations, we’re after a rough outline/template and how the GPU and the CPU cooperate.

4) Parallelisation strategies (15%)

Give a short description of the following:

a) Embarrassingly parallel computations

b) Divide and conquer

c) Pipelined computations

5) Parallel execution (15%)

Given the following code:

```
for (i=2; i<6; i++) {
    x = i - 2 * i + i * i;
    a[i] = a[x];
}
```

a) Can you run the iterations in parallel without causing problems? Explain why.

Hint: Bernstein's conditions are useful for analysing the problem.

6) Barriers and concurrency (20%)

a) What is a barrier? What purpose does it have?

The following is a Python program with a barrier of size 2. By calling `run_sync`, we can start `nthr` threads that each try to call `b.wait() nsyncs` times.

```
import threading
import sys
import time

b = threading.Barrier(2)

def syncthr(thid, nsyncs):
    for i in range(nsyncs):
        print(thid, "waiting for barrier round", i)
        b.wait()
        time.sleep(1)

def run_sync(nthr, nsyncs):
    # make 'nthr' worker threads, telling each to sync 'nsyncs' times
    workers = [threading.Thread(target=syncthr, args=(i, nsyncs)) for i in range(nthr)]

    # start the threads
    for w in workers:
        w.start()
    # wait for every thread to complete
    for w in workers:
        w.join()
```

Which of the following calls will complete? Explain why. It may be easier to draw the execution of each thread on a timeline to help solve the problem.

b) `run_sync(2, 1)`

c) `run_sync(3, 1)`

d) `run_sync(3, 2)`