UiT

THE ARCTIC UNIVERSITY OF NORWAY

# EXAMINATION QUESTION PAPER

| | |
|---|---|
| Exam in: | INF-3201 Parallel Programming |
| Date: | Wednesday, November 30, 2016 |
| Time: | Kl 09:00 - 13:00 |
| Place: | Adm.bygget, Aud. max |
| Approved aids: | Computer required for exam |
| Type of sheets (squares/lines): | WiseFlow |
| Number of pages incl. cover page: | 3 |
| Contact person during the exam: | John Markus Bjørndalen |
| Phone: | 90148307 |

**NB! It is not allowed to submit scratch paper along with the answer sheets. If you do submit scratch paper, it will not be evaluated.**

**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, but please indicate important information that must be passed through function calls.

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

Good luck!

## 1. Terms (15%)

a) Assume that we have a program where 10% of the code must be executed sequentially. What is the maximum potential *speedup* of the program (ignoring communication overhead) if we execute it on 32 CPU's? You only need an approximation if you show the use in the correct formula.

b) What is *Amdahls law*?

c) A term related to speedup is *efficiency*. Explain the term in the context of running a parallel program on a cluster.

d) What is *superlinear speedup*? Why can this happen?

## 2) Load Balancing (10%)

Give a short explanation of the following:

a) Load balancing.

b) Static load balancing.

c) Dynamic load balancing.

d) What separates applications (or algorithms) that benefit from static load balancing vs. ones that benefit from dynamic load balancing?

## 3) 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?

## 4) Parallelisation strategies (10%)

Give a short description of the following:

a) Embarrassingly parallel computations

b) Divide and conquer

c) Pipelined computations

## 5) 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 optimisation, we’re after a rough outline/template and how the GPU and the CPU cooperate.

A suggestion for a simple GPU program would be to add two vectors.

## 6) Pipelines, CSP and MPI (30%)

We are going to make a simple pipelined program with $P$ processes. Each process adds a constant to a received partial sum and passes it to the next process. The first process is the *source* and the last process is the *sink*. For simplicity, just assume that the source has a list of numbers of length $N$ and that the last process collects the finished numbers in a result list.

When writing the programs, you can use simple pseudocode. You don’t need to remember all of the parameters or exact names for API calls. You can also use a CSP-based language (PyCSP, JCSP, Go or pseudocode similar them) instead of syntax from CSP theory.

a) Write pseudocode for a simple CSP program implementing the program.

b) Write pseudocode for a simple MPI program implementing the program.

It often makes sense for the source and the sink to be the same process (the process with the problem is often the one using the solution). This changes the configuration from a line to a circle.

c) This can easily cause deadlocks. Why?

d) CSP has a mechanism we can use to avoid deadlocks in this situation. What is it and how does it work? Modify your program to avoid deadlocks using this mechanism.

e) MPI provides multiple options for avoiding deadlocks with the circle configuration. Describe one and show how you can change your program to use it.