Exam

DVA336 PARALLEL SYSTEMS

8:10 12:30

January 14, 2019

This exam includes 8 problems, for a total of 80 points.

General information

1. This exam is closed book.
2. A simple non-programmable calculator is permitted.
3. Write your answers in English.
4. Make sure your handwriting is readable.
5. Answer each problem on a separate sheet.
6. Write your identification code on all sheets as required for anonymous exams.
7. All answers must be motivated. If necessary, make suitable assumptions.

Grading

The passing marks are 90% for grade 5, 70% for grade 4, and 50% for grade 3. This corresponds to the passing grades A, C, and E in the ECTS grading system.

Contact information

For questions on problems 1, 3-7, please contact Gabriele Capannini, 021-101458.
For questions on problems 2 and 8, please contact Björn Lisper, 021-151709.

Good luck!

1. (a) [5 p] What the Bernstein's conditions are for?

   (b) [5 p] Analyze the following sequential pseudo-program (where `A`, `B`, and `C` are three integer variables) and state which operations can be performed independently in parallel:

   ```
   A = 10;
   B = A+1;
   A = B;
   C = A+B;
   ```

2. (a) [4 p] The data parallel reduce operation has parallel time complexity $O(\log n)$. Explain why!

   Hint: think about how you could implement reduce using more basic data parallel primitives, taking their parallel complexities into account.

   (b) [5 p] Schwartz' algorithm implements reduce on a fixed number of processors, which typically is much less than the number of elements in the data parallel data structure that is reduced over. Explain how Schwartz' algorithm works. What is the time complexity for using the algorithm to reduce n elements on k processors?

3. (a) [7 p] Write the parallel version of the following program by using AVX (or SSE) instructions when it is possible (syntax of 'SIMD' instructions is not important but make sure to point out the details of the new implementation).

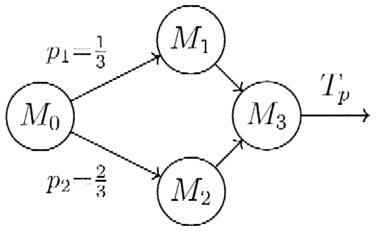
   ```
   #include <stdio.h>
   #include <stdlib.h>

   int main(int argc, char* argv[]) {
     int n = atoi(argv[1]);
     float * A = (float*) malloc(sizeof(float)*n);
     for(int i=0; i<n; ++i)
       A[i] = rand()/(RAND_MAX+1.0f);
     float sum = 0.0f;
     for(int i=0; i<n; ++i)
       sum += A[i];
     printf("avg(A) = %f\n", sum/n);
     free(A);
     return 0;
   }
   ```

   (b) [3 p] What do *AoS* and *SoA* stand for and how are they related to AVX/SSE programming?

4. (a) [6 p] Give the definition of the *Gustafson's Law* and the *Amdahl's Law* and discuss the differences between them.

(b) [4 p] Assume to have a program P of which serial fraction is 25%. Approximate the scalability s of the parallel version of P when running on 1000 processors according to the Amdahl's Law and the Gustafson's Law.

5. Let S be a system of 4 modules $M_{0..3}$ computing, respectively, the functions $f_{0..3}$ having latency $50t$, $20t$, $70t$, and $40t$ where t is a generic time unit. The modules operate according to the 'OR-logic' and M_0 sends an element to M_1 with probability $p_1 = 1/3$ and to M_2 with probability $p_2 = 1 - p_1$:



$$\begin{tikzcd} & M_1 \arrow[dr] & & \\ M_0 \arrow[ur, "p_1=\frac{1}{3}"] \arrow[dr, "p_2=\frac{2}{3}"'] & & M_3 \arrow[r, "T_p"] & {} \\ & M_2 \arrow[ur] & & \end{tikzcd}$$

(a) [5 p] What is the interdeparture time T_p of S?

(b) [5 p] What is the interdeparture time T_p of S if $f_1 = 540$ ms?

6. (a) [7 p] Describe what the *symmetric* and *asymmetric synchronization* mechanisms are and what is the main difference between them with respect to *mutual exclusion* problem.

(b) [3 p] What is a *spinlock*? Why it is important for *blocking locks*?

7. MPI provides a collective function called `MPI_barrier(MPI_Comm comm)` which blocks the calling process until all processes in the communicator `comm` have reached this routine.

(a) [7 p] Implement the barrier function by means of point-to-point functions, namely `MPI_Send()` and `MPI_Recv()` (you can use a simplified syntax where send/receive has only message and destination/source as parameters).

(b) [3 p] Calculate the complexity of your solution as we did in the lecture.

8. (a) [11 p] Define a data parallel algorithm that computes the factorial $n! = n \cdot (n-1) \cdot \ldots \cdot 1$. You can assume that $n \geq 0$.
Use the notation for data parallel algorithms that we have used in the lectures on data parallelism and parallel algorithms (not OpenMP or such). You can assume the existence of a data parallel array with n elements.
What is the parallel time complexity for your algorithm? To get full credits, your solution must have a parallel time complexity that is significantly lower than the sequential complexity for the straightforward sequential solution. (You can assume that multiplication takes constant time, regardless of the size of the operands.)