Exam

DVA336 Parallel Systems

14:10–18:30

August 12, 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!

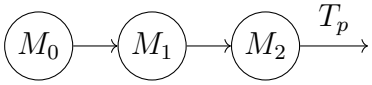
1. (a) [6 p] Explain what the *cache-coherence* mechanism is and how it works.
   (b) [4 p] Explain what the *false-sharing* phenomenon is and how to avoid it.

2. (a) [3 p] Explain how send and get communication work in the data parallel model! Is there any risk of race conditions for any of these operations? If yes, mention which one(s) and explain why!
   (b) [4 p] What is the Divide and Conquer algorithm paradigm? Explain briefly how it works, and why such algorithms can be good parallel algorithms!

3. Answer the following questions about the AVX instructions:
   (a) [2 p] What type of parallelism they implement?
   (b) [3 p] For which kind of computations they are useless? Provide also an example.
   (c) [3 p] How should data be stored in memory to enable/facilitate their usage?

4. (a) [7 p] Give the definition of the *Gustafson's Law* and the *Amdahl's Law* and discuss the differences between them.
   (b) [4 p] Let A be a serial algorithm and assume that two thirds of the work performed by A are perfectly parallelizable. Give an approximation of the scalability achieved by the parallel version of A when running on 150 processors according to the Amdahl's Law and the Gustafson's Law.

5. Let S be a system performing a stream-based computation. S consists of 3 sequential modules $M_{0..2}$ computing, respectively, the functions $f_{0..2}$ having latency $10t$, $45t$, and $5t$ where t is a generic time unit. M_0 produces the input stream.



   In particular, f_1 is a linear composition of three sub-functions:

$$f_1(x) = f_{13}(f_{12}(f_{11}(x)))$$

   Assume that f_{11}, f_{12}, and f_{13} are pure mathematical stateless functions and the latency of each of them is $15t$ and the communication overheads don't affect the performance of S.
   (a) [4 p] What is the inter-departure time T_p of S?
   (b) [4 p] What are the ideal performance of S? Explain and motivate your answer.
   (c) [8 p] Is there any bottleneck in the computation? If yes, provide a parallelization of the bottleneck such that S can achieve its ideal performance and explain your solution by showing the related calculations.

6. (a) [9 p] Describe what the *symmetric* and *asymmetric synchronization* mechanisms are and what is the main difference between them with respect to *mutual exclusion* problem.

7. Consider the following MPI function:

```
void MPI_Foo(int *mask, MPI_Comm comm) {
  int rank, size, root = 0, msg = 1;
  MPI_Comm_rank(comm, &rank);
  MPI_Comm_size(comm, &size);
  if(mask[rank]==0)
    return;
  while(root<size && mask[root]==0)
    root += 1;
  if(root==size)
    return;
  if(rank==root) {
    for(int i=0; i<size; ++i)
      if(i!=root && mask[i]!=0)
        MPI_Recv(&msg, 1, MPI_INT, i, 100, comm, MPI_STATUS_IGNORE);
    for(int i=0; i<size; ++i)
      if(i!=root && mask[i]!=0)
        MPI_Send(&msg, 1, MPI_INT, i, 100, comm);
  } else {
    MPI_Send(&msg, 1, MPI_INT, root, 100, comm);
    MPI_Recv(&msg, 1, MPI_INT, root, 100, comm, MPI_STATUS_IGNORE);
  }
}
```

Assuming that all processes share the same copy of `mask` and such array is properly instantiated (i.e., the number of its entries is at least the size of the `comm` communicator), answer the following questions:

(a) [4 p] What functionality is implemented by the `MPI_Foo` function?

(b) [2 p] What is the complexity of `MPI_Foo`?

8. (a) [13 p] Define a data parallel algorithm that computes an integer from its 2-complement binary representation stored as a data parallel array of zeros and ones! Use the notation for data parallel algorithms that we have used in the lectures on data parallelism and parallel algorithms (not OpenMP or such).

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.

Hint 1: if $b_{n-1} \cdots b_0$ is the binary representation, with b_0 the least significant bit, then the represented integer can be computed as $\sum_{i=0}^{n-1} b_i \cdot 2^i$.

Hint 2: beware that a sequential computation of 2^i takes time $O(i)$.