EXAM: Parallel Programming

June 10, 2016, 14:00 – 17:00

The duration of this exam is 3 hours. The number of (itemized) questions is 6 with a total of 18 items. Behind each item the number of points to be earned is depicted between square brackets (total number of points is 100). This exam is "**open book**", which means you can use your own notes and textbook. The use of electronic devices is not allowed. Give arguments with each answer: an answer without arguments will not be graded [0].

Question 1 [15]

1. Describe the difference between a message passing platform and a shared address space platform. [3]
2. Explain which problems need to be solved when building a shared address space platform as a physical parallel processor architecture (Shared Memory Machine), in which each processor has its own cache. Describe also possible solutions to these problems. [6]
3. Suppose we want to build a shared address space platform using a multistage interconnection network to connect the processors to a set of memory banks. Describe how main memory can be divided into these banks and how memory addressing can be used to route the data in this interconnection network (assume an Omega network was used for the interconnection network). [6]

Question 2 [20]

1. Describe how Minimum Spanning Tree and Travelling Salesman Problem are related. [3]
2. Describe how Prim's algorithm to solve Minimum Spanning Tree is very similar to Dijkstra's algorithm to compute Single Source Shortest Path problem. Explain why both algorithms are in fact very similar. [5]
3. Explain that Boruvka's algorithm for Minimum Spanning Tree has the advantage over Prim's algorithm that this algorithm is intrinsically parallel. [3]
4. When actually parallelizing Boruvka's algorithm on a Distributed Memory Computer this advantage can easily disappear. Explain why. [9]

Question 3 [20]

1. Dense Matrix times Vector Multiplication can be rather easily parallelized by loop blocking techniques resulting in the parallel execution of multiple level 2 BLAS routine. Describe how this works. [3]
2. Explain that for Sparse Matrix times Vector Multiplication the solution as described in 1 leads to load-balancing problems. Describe how this problem can be solved. [3]
3. Describe step by step how this load balancing problem for Sparse Matrix Multiplication can be solved automatically within **tUPL** by the application of transformations (among which: loop blocking, materialization, and orthogonalization) starting from the very simple initial specification:

```
forelem ( t; t ε T )
{    Value_C[t.i]+= Value_A[t.i,t.j]*Value_B[t.j]}
```

   [14]

Question 4 [15]

1. Describe how dense BLAS2/3 routines can be employed when solving ***Ax=b*** for a Sparse Matrix ***A*** by Block LU factorization. [3]
2. Describe why Block LU factorization does not work very well for Sparse Matrices ***A***. [3]
3. In general LU factorization for solving linear systems of equations can lead to numerical instabilities. Describe that this problem is the same for both dense matrices and sparse matrices, but that in practice, when implementing pivoting strategies, this problem is much harder to solve for sparse matrices compared to dense matrices. [9]

Question 5 [15]

1. Explain how the Bitonic Sort Network works. [6]
2. Show that each stage in the Bitonic Sort Network can be implemented as a Perfect Shuffle. Demonstrate this for the Bitonic Sort Network with 8 inputs and 8 outputs. [9]

Question 6 [15]

1. Explain in detail why in **tUPL** the programmer does not have to bother about data structures. [3]
2. Describe the relationship between Data Flow Computing and **tUPL**. [6]
3. Explain why or why not **tUPL** might lead to an efficient implementation platform for parallel programming whereas Data Flow computing was never successfully/efficiently implemented. [6]