EXAM: Parallel Programming I

December 20, 2017, 14:00 – 17:00

The duration of this exam is 3 hours. The number of (itemized) questions is 6 with a total of 16 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 [20]

1. Explain why a Non-Uniform-Memory-Access-Architecture (NUMA) is easier to realize with a distributed memory architecture than a Uniform-Memory-Access-Architecture. **[5]**
2. If a NUMA architecture is implemented on a distributed memory architecture for which each CPU has (multiple levels of) cache, then explain how cache coherence protocols can be used to ensure an effective use of these caches. **[5]**
3. Describe a possible implementation of these cache coherence protocols on such a NUMA architecture. **[10]**

Question 2 [15]

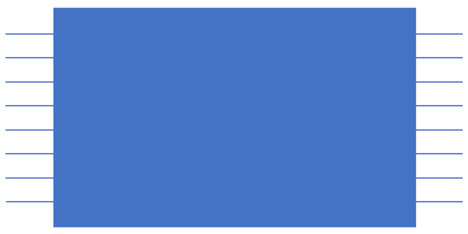
1. Explain the advantages/disadvantages of multi stage interconnection networks compared to single stage interconnection networks. **[5]**
2. Explain the differences/correspondences between an N-cube interconnection network and a Hypercube interconnection network. **[10]**

Question 3 [10]

1. Explain (exactly) how block LU factorization for solving dense systems of linear equations works together with the forward substitution and backward solve steps. A mere repetition of the course material is not sufficient! **[5]**
2. Explain which problems arise when implementing block LU factorization for a sparse system of linear equations. **[5]**

Question 4 [15]

1. Draw a Batcher sorting network for 4 numbers (4 inputs and 4 outputs). **[5]**
2. Given a Batcher sorting network for 8 numbers, describe how a Batcher sorting network can be constructed for 16 numbers by using the 8 number Batcher sorting network as "building blocks". Draw the resulting network and use in your drawing



as a black box to denote the 8 number Batcher sorting network. **[10]**

Question 5 [25]

Consider the following code fragment which is representative for a solution process (5-point stencil) to solve partial differential equations on a **N** by **N** grid:

```
for (it = 1; it<=niter; it++) {

   for (i = 2; i <= N-1; i++) {

      for (j = 2; j <= N-1; j++) {

         A[j][i] = (B[j][i] + B[j-1][i] + B[j+1][i] +
                    B[j][i-1] + B[j][i+1]) / 5

      }
   }
   /* Assign B to A: A[j][i] = B[j][i] */
}
```

1. Explain how you would parallelize the execution of this loop structure on a distributed memory architecture. **[5]**
2. Give the implementation of this parallelized code with MPI send/receive commands and primitives. **[5]**
3. Explain which performance issues/bottlenecks you would have when running this code on an actual architecture (say DAS 4). **[5]**
4. In which way would you resolve these bottlenecks. **[10]**

Question 6 [15]

1. Consider the following **tUPL** specification for sorting:

   ```
   whilelem ( t; t ε T )
       {
           if ( X[t.i] > X[t.j] )
               swap ( X[t.i], X[t.j] )
       }
   ```

   Describe a step by step execution of this `whilelem` loop structure which results in a sorted list in increasing order, using as a tuple reservoir: `T = {<4,1>, <99,2>, <1,99>}` and initial values: `X[4]=14, X[1]=17, X[99]=11, X[2]=15`. **[5]**

2. Explain in your own words how chaining works in **`tUPL`** to optimize the execution of **`whilelem`** loops. **[5]**
3. Identify the chains and give for each chain the tuple order for which the sorting specification above can be optimized. Also, derive the number of computational steps needed for the optimized version. **[5]**