Answers to Example Exam in4049TU

Exercise 1

1a. NUMA stands for cache coherent Non-Uniform Memory Architecture. This is an architecture where the processors have local memories and local caches. The union of local memories is behaving like a shared memory, but the access time to this shared memory is dependent on whether the data is in the local memory of the processor accessing the data or in a remote memory. The local caches in the system are kept in a coherent state, i.e. a write operation on a data element in the cache of one processor is also done on possible copies of this data element in other caches.

1.b The following levels of parallelism can be distinguished:

- bit-level parallelism, e.g. all bits in a data word can be operated on simultaneously
- instruction level parallelism; a number of instructions are executed simultaneously
- multiple functional units; a number of functional units can operate in parallel
- multiple processors

1c. An SIMD processor consists of a large number of "small" processors and a control processor. The control processor generates the instruction stream. Each processor executes the instruction coming from the control processor. Processors can be selectively switched off for certain instructions as to facilitate conditional execution.

1d. The following three interconnection topologies can be mentioned:

- Mesh. advantage: regular interconnection structure with a constant number of links per processor. Disadvantage: relatively large diameter $O(n)$.
- Hypercube. advantage: small diameter $O(\log n)$. Disadvantage: number of links per processor grows if dimensionality of the hypercube increases. Not so regular interconnection structure in 2D or 3D.
- Tree. Advantage: constant number of links per processor. Small diameter $O(\log n)$. Disadvantage: low bandwidth in higher layers of the tree.

Exercise 2

2a. The following phases have to be passed:

- Decomposition in tasks
- Assignment of tasks to processes or threads
- Addition of communication and synchronization points (orchestration)
- Assignment of processes to processors (mapping)

2b. A barrier is a synchronization operation that forces processors to wait at a certain point in the code execution until all processors have reached the same point.

2c. This is caused by the overhead in setting up communication between two programs. Usually, the OS must allocate buffer space and the communication path has to be set up. This overhead is usually a constant factor that is virtually independent of the number of bits that has to be sent across.

2d. This must be done through a shared variable that can be locked. This shared variable can represent a TRUE or FALSE situation. Each operation on this shared variable must be atomic, i.e. only one processor at a time can change the value.

Exercise 3

3a. With the Jacobi process, the new approximation of the result $\underline{x}$ on the k-th iteration, $\underline{x}_k$, only depends on the vector $\underline{x}_{k-1}$. The calculations within an iteration are therefore fully parallel. With the Gauss-Seidel process, $\underline{x}_k$ depends both on $\underline{x}_k$ and $\underline{x}_{k-1}$. Therefore, the calculations within an iteration are mutually dependent. A way to obtain some parallelization is the application of red-black ordering. With red-black ordering the calculations within an iteration are split in two sub-iteration spaces. The calculations within each sub-iteration space can then be done in parallel.

3b. The Jacobi, Red-Black SOR end the CG methods are parallelized by using the domain decomposition method. The numerical grid is partitioned in p parts and each processor is assigned a part of the grid. The calculations on all parts of the grid are then to be done in parallel by the p processors, with an exchange of the border values after each time step.

3c. Suppose the 2-D grid is partitioned in $\sqrt{p} \times \sqrt{p}$ blocks. Then each block has $((n/\sqrt{p}) \times (n/\sqrt{p}))$ grid points. The maximum number of grid points on the edges of a block (with 4 neighbors) is $4(n/\sqrt{p} - 1)$. The computation time of a block is linearly dependent on the number of grid points in a block, $T_{comp} = (n/p) \cdot t_{fl} \cdot c$, where t_{fl} is the time it takes for a floating-point operation and c is a constant. There are maximally 4 communications of length $(n/\sqrt{p} - 1)$. Hence, the communication time is $T_{comm} = 4(\alpha + ((n/\sqrt{p}) - 1)\beta)$, where α and β are the startup time and the speed of the communication channel. The ratio is the given by

$$\frac{T_{comp}}{T_{comm}} = \frac{n^2 \cdot t_{fl} \cdot c}{4(\alpha + ((n/\sqrt{p}) - 1)\beta} = O(n/\sqrt{p})$$

3d. The two parallel FFT algorithms have the same parallel time complexity. The difference is in the data distribution and the computation and communication phases. With a block distribution a group of (m/p) joint data is assigned to the same processor. With a cyclic distribution the data is assigned one after the other to the next processor. Furthermore, with the block distribution algorithm communication only takes place in the first log(p) steps and there is no communication in the remaining log(m/p) steps. With the cyclic distribution algorithm there is no communication in the first log(m/p) steps and all communication takes place in the

last $\log(p)$ steps.

Exercise 4

4a. We want a balanced distribution of the workload to the processors with a minimum of communication. The first objective asks for a partitioning of the graph in a number of subgraphs that are equal in size. The second objective asks for a minimum number of connections between the subgraphs.

4b. No. The recursive spectral method usually gives a more optimal partitioning of a graph, but often costs much more computation time than the Kernighan-Lin algoritm. The choice is dependent on the type of applications.

4c. The most time consuming part are the calculations of the forces that the particles have on each other. The calculation time of these forces is $O(N^2)$ on a system with N particles.

4d. No. The calculations of the forces in the Barnes-Hut algorithm are reduced to $O(N\log(N))$. The other steps, such as the building of the quadtree or the octtree also have a time complexity of $O(N\log(N))$. So in an efficient parallel implementation of the Barnes-Hut algoritm also the other steps must be parallelized.