Parallel Programming for High-Performance Applications
23 October 2014
Department of Computer Science, Faculty of Sciences

The exam has 9 questions. Your answers should be to the point: address the questions and omit information that is not asked for. The grading system is shown after the last question.

1. Given below is a sequential algorithm that makes a fixed number of sweeps over an N-by-M array and during each sweep updates elements:

```
float G[1:N, 1:M], G2[1:N, 1:M];
for (step = 0; step < NSTEPS; step++) {
  for (i = 3; i < N-1; i++) {
    for (j = 3; j < M-1; j++) {
      G2[i,j] =
         weight * (G[i,j] + G[i-1,j] + G[i-2,j] +  G[i+1,j] + G[i+2,j]
           + G[i,j-1] + G[i,j-2] + G[i,j+1] + G[i, j+2]) / 9;
    }
  }
  G = G2;
}
```

   What is the communication scheme if this algorithm is parallelized by partitioning the array row-wise over P processors (i.e., giving each processor N/P consecutive rows)? Make clear which data the processors will exchange. (You may assume that N is a multiple of P.)

2. (a) What is a "fat tree" topology? What problem of normal (non-fat) tree topologies does it try to solve?

   (b) What is a NUMA architecture? Can it be programmed just like a normal shared-memory machine?

3. Consider a parallel master/slave program in which each slave process gets a large piece of work, executes it, and then terminates. This program has to be extended with the capability to terminate (kill) slave processes prematurely. In the extended program, the master process should be able to send a termination message to all other processes (the slaves). Each slave process should terminate as soon as possible when a termination message arrives (e.g., by invoking the 'exit' system call). The termination messages may arrive at any point in time during the execution of the slaves, making it somewhat difficult to receive and handle such incoming messages.

   (a) How would you implement the receipt of such termination messages if the program is written in SR?

   (b) How would you implement the receipt of such termination messages if the program is written in MPI?

   Your code need not be syntactically correct MPI or SR; however, explain clearly how you benefit from the primitives that MPI and SR provide.

4. A major problem with asynchronous message passing is the possibility of buffer overflows at the receiving machine. Is it also possible to get such buffer overflows with RMI (Remote Method Invocation) in Java? Explain your answer, and take into account that Java also supports multi-threading.

5. (a) What are the most important advantages of HPF (High Performance Fortran) over message-passing systems?

   (b) What is the most important disadvantage of HPF compared to message-passing systems?

6. Parallel search algorithms like IDA* typically use shared transposition tables. One approach to implement such tables on a distributed-memory system is to replicate the tables on all processors, so lookups can be done locally, without any communication. Explain why the Transposition Driven Scheduling (TDS) algorithm still is much more efficient than replicated tables, especially for large numbers of processors. What are the main advantages of TDS over replicated tables?

7. Both branch-and-bound (as used, for example, for the Traveling Salesman Problem) and Barnes-Hut (used for N-body problems) use techniques to cut-off (prune) part of the computations. Parallel branch-and-bound algorithms can sometimes obtain superlinear speedups, because the parallel version may (for certain input problems) perform less work than a sequential algorithm. Can the parallel Barnes-Hut algorithm also obtain superlinear speedups in this way? Explain your answer.

8. Please provide short answers to the following questions:

   (a) Explain how all threads of an NVIDIA GPU grid can be synchronized.

   (b) Give examples of two differences between the shared and global memory in an NVIDIA GPU.

   (c) Explain the difference between shared memory and the L1 cache in an NVIDIA GPU.

   (d) What is memory coalescing? Why is it important? Give an example (using pseudocode) of a GPU kernel that leads to perfect memory coalescing and another example of a kernel that leads to imperfect or no memory coalescing.

   (e) Give an example of an application (and pseudocode) as a use-case for a global atomic operation. Explain what would happen without the atomic operation.

9. (a) What are the best and the worst case performance for a GPU when executing the following *kernel* code? Assume the execution of one operation (any of the lines 1,2,3,4,6,7,9) is T cycles and the code runs in a single block of 32 threads (one warp). Justify your answer.

```
1. i = my_thread_id();
2. for (j=0; j<5; j++) {
3.   if (a[i] % 2 == 0)
4.      a[i] = a[i]+2;
5.   else
6.   if (a[i] % 5 == 0)
7.      a[i] = a[i] * 2;
8.   else
9.      a[i] = a[i] * 4;}
```

(b) Explain the steps required for vectorizing this code for a 4-way SIMD processor and give a pseudocode solution for this vectorization. What is the maximum performance improvement you expect from this vectorization and why?

```
for (i=0; i<N; i++) {
    for (j=0; j<4; j++) {
            a[i] = a[i] * 2;
    }
    b[i]=a[i]-1;
}
```

The instructions you may use are vload, vstore (for loading and storing data from arrays a and b into SIMD vectors), vset (to set scalar values in an SIMD vector) and vsub, vmul (for SIMD arithmetic operations). Their syntax is not relevant for this problem - choose whatever is convenient.

Points

1	2a	2b	3a	3b	4	5a	5b	6	7	8a	8b	8c	8d	8e	9a	9b
10	5	5	5	5	5	5	5	10	10	3	3	3	3	3	5	5

Total: 90 (+ 10 = 100)