UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in IN3200 — High-Performance Computing and Numerical Projects

Day of examination: June 11th-18th, 2020

This problem set consists of 4 pages.

Appendices: None

Permitted aids: Any

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

This home exam accounts for 60% of the final grade

The first and second home exams account for 20% each. The final grade will be pass or no pass. **Each student should independently answer the following questions by her/himself.** All the answers should be contained in one PDF file. Please also read the additional info given at Inspera.

Problem 1 (weight 10%)

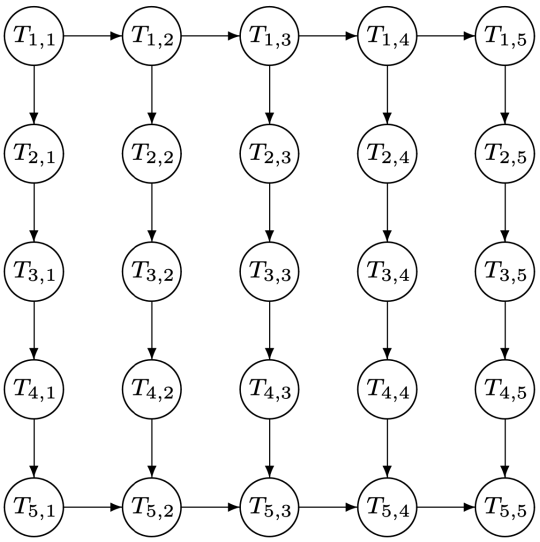

Figure 1: The dependency relationship between 25 tasks.

Figure 1 shows the dependency relationship between 25 tasks: $T_{1,1}, T_{1,2}, \ldots, T_{5,5}$. Each directed edge in the graph connects a pair of "source" and "destination" tasks.

(Continued on page 2.)

A destination task cannot be started until all its source tasks are carried out. All the 25 tasks are equally time-consuming, requiring one hour of a worker. (We also assume that it is not possible to let two or more workers collaborate on one task for a faster execution.)

1a (weight 5%)

Explain in detail how many hours minimum do three workers need to finish all the 25 tasks.

1b (weight 5%)

What is the maximum speedup that can be achieved? How many workers are needed to achieve the maximum speedup? Please explain your answers.

Problem 2 (weight 10%)

How would you parallelize the following code segment using OpenMP directives? Please provide sufficient explanations. Also, what are your opinions about the speedup that can be achieved as the number of threads is increased?

```
int i,j, sqrt_N;

char *array = malloc(N);  // N is a predefined very large integer
array[0] = array[1] = 0;
for (i=2; i<N; i++)
  array[i] = 1;

sqrt_N = (int)(sqrt(N));  // square root of N
for (i=2; i<=sqrt_N; i++) {
  if (array[i]) {
    for (j=i*i; j<N; j+=i)
      array[j] = 0;
  }
}

free (array);
```

Problem 3 (weight 15%)

Consider the following function

```
void sweep (int N, double **table1, int n, double **mask, double**table2)
{
  int i,j,ii,jj;
```

(Continued on page 3.)

```
  double temp;
  for (i=0; i<=N-n; i++)
    for (j=0; j<=N-n; j++) {
       temp = 0.0;
       for (ii=0; ii<n; ii++)
         for (jj=0; jj<n; jj++)
           temp += table1[i+ii][j+jj]*mask[ii][jj];
       table2[i][j] = temp;
    }
}
```

All the three 2D arrays `table1`, `mask` and `table2` are allocated beforehand, where `table1` is of dimension $N \times N$, `mask` of dimension $n \times n$, and `table2` of dimension $(N-n+1) \times (N-n+1)$. It can be assumed that N is much larger than n. (For example, N is at least 10000 whereas n is at most 10.)

3a (weight 5%)

Show how OpenMP directives can be used to parallelize the function `sweep`. Please provide necessary explanations.

3b (weight 10%)

Given a computer with two Intel Xeon 24-core processors of model 8168 (`https://ark.intel.com/content/www/us/en/ark/products/120504/intel-xeon-platinum-8168-processor-33m-cache-2-70-ghz.html`). How would you estimate the theoretical minimum computing time needed by the function `sweep` when the values of N and n are known? Please elaborate your reasoning.

Problem 4 (weight 20%)

We aim to design a *simplistic* simulator for the day-to-day spread of a virus among a large number of people. (**Note:** There is no need to write a complete program. Code segments and/or high-level pseudo code, with clear explanations, are sufficient.) The following are some assumptions to be used by this very simple simulator:

- The interactions among the people remain the *same* every day. (That is, if two people meet each other today, then they will also meet tomorrow, the day after tomorrow and so on; unless one of them is so ill to have to stay at home.)
- The *fixed* person-person daily interactions are described by a graph, where the people are represented by vertices, and an edge between a pair of vertices means that the corresponding two people have a long enough daily interaction that bears the risk of transmitting the virus in-between (if one of them is ill).
- No person is initially immune to the virus.
- Once infected, a person *cannot* immediately infect other people on the same day of the infection. Starting from the next day until day T (an input parameter to the simulator), the infected person becomes ill and has the potential of infecting other

(Continued on page 4.)

healthy, non-immune people with whom she/he interacts. After day T, the infected person is so ill that he/she will stay at home and thus without the possibility of infecting more people.

- An ill person will become healthy after staying at home for, say, X days. A recovered person is assumed to be immune to the virus, will thus not be infected again.
- If a healthy, non-immune person interacts with an ill person (who has been infected for at least one day), the probability of being infected is f every day until the ill person starts to stay at home. (The value of f is a prescribed percentage, as an input parameter to the simulator.)

4a (weight 5%)

Please describe in detail the data structure you will use to store the person-person interaction graph, and the healthy/ill/immune states of all the people. Please motivate your design by efficiency considerations if applicable.

4b (weight 10%)

Please sketch a function that can be used to evolve the healthy/ill/immue states of all the people by one day. (For example, the inputs can include the person-person interaction graph and the healthy/ill/immune states of all the people from the previous day. The output can be the healthy/ill/immune states of all the people for the present day.) Please motivate your implementation sketch by efficiency considerations if applicable.

Hint: It is standard practice to use a random number generator (such as the standard function `rand`) to simulate whether a healthy, non-immune person is to be infected by an ill person, while statisfying the prescribed probability f.

4c (weight 5%)

In order to study different scenarios and collect statistics, we will need to run a large number of simulations, which differ by the T value, and/or the f value, and/or who and how many the initially infected people are. Please present your high-level ideas about how parallel computing can be applied to efficiently run the large number of simulations. (There is no need to show the programming details.)

Problem 5 (weight 5%)

Please present your thoughts (in your own words) about the advantages/disadvantages of MPI programming versus OpenMP programming for a computer with multiple sockets of multicore CPUs (that is, a shared-memory system with NUMA architecture).